

CICS Transaction Server for z/OS



Web-Services mit CICS verwenden

Version 5 Release 5

CICS Transaction Server for z/OS



Web-Services mit CICS verwenden

Version 5 Release 5

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen“ auf Seite 737 gelesen werden.

Inhaltsverzeichnis

Informationen zu dieser PDF v

Kapitel 1. CICS und Web-Services 1

CICS- und SOAP-Web-Services	4
SOAP-Knoten	5
SOAP-Nachrichten und die Anwendungsdatenstruktur	9
WSDL und die Anwendungsdatenstruktur	12
WSDL und Nachrichtenaustauschmuster	14
In der Web-Service-Bindungsdatei	16
Externe Standards	16
SOAP-Architektur und -Nachrichtenformat	16
Verwendung von SOAP-Web-Services planen	28
CICS- und JSON-Web-Services	29
Konzepte von JSON-Web-Services	30
Konzepte REST-konformer JSON-Web-Services	33
Verwendung von JSON-Web-Services planen	35
CICS und z/OS Connect	38
Funktionalität von z/OS Connect for CICS	43

Kapitel 2. Web-Services in CICS konfigurieren 45

CICS-System für Web-Services konfigurieren	45
CICS-Ressourcen für Web-Services	45
CICS für die Verwendung des WebSphere MQ-Transports verwenden	50
Interoperabilität zwischen dem Web-Service-Assistenten und WSRR	57
Web-Service-Infrastruktur erstellen	59
Web-Service-Infrastruktur	59
CICS-Infrastruktur für SOAP-Service-Provider erstellen	70
CICS-Infrastruktur für SOAP-Service-Requester erstellen	72
CICS-Infrastruktur für JSON-Service-Provider erstellen	74
CICS-Infrastruktur für einen Nicht-Java-JSON-Service-Provider erstellen	75
z/OS Connect for CICS konfigurieren	77
Pipelinekonfigurationsdateien	95
Anwendungshandler	149
Nachrichtenhandler	151
SOAP-Nachrichtenhandler	159
In der Pipeline verwendete Container	165
Laufzeitverarbeitung für Web-Services	198
Unterstützung für Web-Service-Transaktionen	207
Unterstützung für MTOM/XOP-Optimierung von Binärdaten	216
Unterstützung für Web-Service-Adressierung	226
Unterstützung für SAML	248

Kapitel 3. Web-Services entwickeln 249

JSON-Web-Service erstellen	249
CICS JSON-Assistent	249

JSON-Service-Provider-Anwendung unter Verwendung des JSON-Assistenten erstellen	293
Einschränkungen von JSON-Web-Services	309
Anwendungsdaten und XML zuordnen und umsetzen	311
CICS-XML-Assistent	312
Zuordnungen aus Sprachstrukturen generieren	426
Zuordnungen aus einem XML-Schema erstellen	429
Anwendungsdaten in XML umsetzen	430
XML in Anwendungsdaten umsetzen	431
Anwendungsdaten und JSON zuordnen und umsetzen	433
CICS JSON-Assistent	434
Zuordnungen aus einer Sprachstruktur generieren	512
Zuordnungen aus einem JSON-Schema generieren	513
Anwendungsdaten durch Verknüpfung mit DFHJSON in JSON umsetzen	515
Anwendungsdaten mit dem API-Befehl TRANSFORM DATATOJSON in JSON umsetzen	516
JSON durch Verknüpfung mit DFHJSON in Anwendungsdaten umsetzen	517
JSON mithilfe des API-Befehls TRANSFORM JSOINTODATA in Anwendungsdaten umsetzen	518
JSON-Web-Service-Clientanwendung erstellen	519
SOAP-Web-Service erstellen	520
CICS-Web-Service-Assistent	521
Web-Service-Provider mit dem Web-Service-Assistenten erstellen	617
Service-Provider-Anwendung aus einer Web-Service-Beschreibung erstellen	617
Service-Provider-Anwendung aus einer Datenstruktur erstellen	620
Kanalbeschreibungsdokument erstellen	623
Generierte Web-Service-Beschreibungsdokumente anpassen	625
SOAP-Fehler senden	627
Web-Service-Requester mit dem Web-Service-Assistenten erstellen	629
Web-Service mithilfe von Tools erstellen	631
Eigene XML-fähige Web-Service-Anwendungen erstellen	632
XML-fähige Service-Provider-Anwendungen erstellen	632
XML-fähige Service-Requester-Anwendungen erstellen	634
Java mit Web-Services verwenden	636
Web-Service im Java-Providermodus auf einem Axis2-JVM-Server implementieren	636
Java-Web-Service erstellen, der XML generiert und parst	638
Java-Web-Service mit COBOL-Schnittstelle erstellen	639
JAX-WS-Web-Service im Requestermodus implementieren	639

Java-Web-Service im Providermodus auf einem Liberty-JVM-Server implementieren	640
SOAP-Nachrichten überprüfen	641
Ungültige und nicht initialisierte von Anwendun- gen bereitgestellte Daten verarbeiten	642
Beispiel 1: Tolerierung von Dezimalfeldern	643

Kapitel 4. Unterstützung für den

Schutz von Web-Services 647

Voraussetzungen für Web Services Security	648
Sicherheit von SOAP-Web-Services planen.	649
Optionen zum Sichern von SOAP-Nachrichten	650
Authentifizierung mit einem Sicherheitstokenser- vice	652
Vertrauenswürdige Clientschnittstelle	654
SOAP-Nachrichten unterzeichnen	654
Signaturalgorithmen	655
Beispiel einer signierten SOAP-Nachricht	655
CICS-Unterstützung für verschlüsselte SOAP- Nachrichten	656
Verschlüsselungsalgorithmen	657
Beispiel einer verschlüsselten SOAP-Nachricht	657
RACF für Web Services Security konfigurieren	658
Web-Services im Providermodus für die Weitergabe von Identitäten konfigurieren	660
Pipeline für Web Services Security konfigurieren	663
Angepassten Sicherheitshandler schreiben	666
Vertrauenswürdigen Client aus einem Nachrichten- handler aufrufen	667
Sicherheit für z/OS Connect	669
Berechtigungen für z/OS Connect-Services und -APIs konfigurieren	669

Kapitel 5. Fehlerbehebung bei Web-

Services 673

Fehlerbehebung bei SOAP-Web-Services	673
Implementierungsfehler diagnostizieren	673
Laufzeitfehler des Service-Providers diagnosti- zieren	676
Laufzeitfehler des Service-Requesters diagnosti- zieren	677
MTOM/XOP-Fehler diagnostizieren	679

Datenkonvertierungsfehler diagnostizieren	681
Fehlerbehebung bei JSON-Web-Services	683
Fehlerbehebung bei JSON-Implementierungspro- blemen	684
Fehlerbehebung für den JSON-Assistenten	685
Fehlerbehebung bei JSON-Anforderungen	686
An den Client zurückgegebene Fehlerantworten	687
Rückgabecodes des JSON-Assistenten	688

Anhang A. Container für verknüpfbare Schnittstelle des JSON-Umsetzungs-

programms 691

Container DFHJSON-JJSON	691
Container DFHJSON-DATA	691
Container DFHJSON-TRANSFRM	692
Container DFHJSON-JVMSEVR	692
Container DFHJSON-ERROR	692
Container DFHJSON-ERRORMSG	693

Anhang B. Web-Service-Beispiele. . . 695

Beispielanwendung des CICS-Katalogmanagers	695
Basisanwendung	696
Basisanwendung installieren und konfigurieren	698
Beispielanwendung mit der BMS-Schnittstelle ausführen	704
Web-Service-Unterstützung für die Beispielan- wendung.	706
Web-Client konfigurieren	718
Web-Service-aktivierte Anwendung ausführen	719
Beispielanwendung implementieren	720
Komponenten der Basisanwendung	725
Dateistrukturen und -definitionen	733
JSON-Beispiele	735
HTTP GET-Beispielanforderung mithilfe einer Abfragezeichenfolge	735
HTTP-Beispielanforderung mit einem JSON- Hauptteil.	735

Bemerkungen 737

Index 743

Informationen zu dieser PDF

In dieser PDF wird beschrieben, wie Sie Web-Services in CICS verwenden. Sie richtet sich an Systemprogrammierer, die für die Konfiguration von CICS zur Unterstützung von Web-Services verantwortlich sind, sowie an Anwendungsentwickler, die für Anwendungen verantwortlich sind, die in einer Web-Service-Umgebung implementiert werden. Vor CICS TS V5.4 hatte diese PDF den Namen "Web Services Guide".

Details zu den Begriffen und Notationen in diesem Handbuch finden Sie unter Conventions and terminology used in the CICS documentation im IBM Knowledge Center.

Datum dieser PDF

Diese PDF wurde am 14. Dezember 2018 erstellt.

Kapitel 1. CICS und Web-Services

CICS stellt Unterstützung für Web-Services bereit.

Was ist ein Web-Service?

Ein Web-Service hat eine Schnittstelle, in der die Implementierungsdetails ausgeblendet werden, sodass er unabhängig von der Hardware- oder Softwareplattform, auf der implementiert ist, und unabhängig von der Programmiersprache, in der er geschrieben ist, verwendet werden kann. Diese Unabhängigkeit fördert flexibel verbundene, komponentenorientierte, technologieübergreifende Implementierungen von Web-Services. Web-Services können alleine oder zusammen mit anderen Web-Services verwendet werden, um eine komplexe Aggregation oder eine Geschäftstransaktion auszuführen.

Von CICS unterstützte Web-Services

CICS unterstützt zwei Web-Service-Protokolle, die SOAP- und die JavaScript Object Notation-Protokolle (JSON). Diese beiden Protokolle unterscheiden sich in ihren Merkmalen und Vorteilen.

Web-Service-Terminologie

Extensible Markup Language (XML)

Ein Standard für Dokumenten-Markup, der eine generische Syntax verwendet, um Daten mit einfachen, lesbaren Tags zu formatieren. Der Standard wird vom World Wide Web Consortium (W3C) unterstützt.

Ursprünglicher SOAP-Absender

Der SOAP-Absender, der eine SOAP-Nachricht ab dem Ausgangspunkt eines SOAP-Nachrichtenpfads erstellt.

JavaScript Object Notation (JSON)

Ein einfaches Datenaustauschformat, das auf der Objekt-Literal-Notation von JavaScript basiert. JSON ist programmiersprachenunabhängig, verwendet aber Konventionen aus Sprachen wie C, C++, C#, Java™, JavaScript, Perl und Python.

JSON-Schema

Ein JavaScript Object Notation-Dokument, das die Struktur und Einschränkungen der Inhalte von anderen JSON-Dokumenten beschreibt.

REST-konform

Bezieht sich auf Anwendungen und Services, die Representational State Transfer-Einschränkungen (REST) entsprechen.

Service-Provider

Die Sammlung von Software, die einen Web-Service bereitstellt.

Service-Provider-Anwendung

Eine Anwendung, die in einem Service-Provider verwendet wird. Typischerweise stellt eine Service-Provider-Anwendung die Geschäftslogikkomponente eines Service-Providers bereit.

Service-Requester

Die Sammlung von Software, die verantwortlich ist für das Anfordern eines Web-Service von einem Service-Provider.

Service-Requester-Anwendung

Eine Anwendung, die in einem Service-Requester verwendet wird. Typischerweise stellt eine Service-Requester-Anwendung die Geschäftslogikkomponente eines Service-Requesters bereit.

Simple Object Access Protocol

Siehe SOAP.

SOAP Ursprünglich ein Akronym für *Simple Object Access Protocol*. Ein einfaches Protokoll für den Austausch von Informationen in einer dezentralen verteilten Umgebung. Es ist ein XML-basiertes Protokoll, das aus drei Teilen besteht:

- Ein Envelope, der ein Framework definiert, in dem beschrieben wird, was in einer Nachricht enthalten ist und wie sie verarbeitet wird.
- Ein Satz von Codierungsregeln zum Ausdrücken von Instanzen von anwendungsdefinierten Datentypen.
- Eine Konvention für die Darstellung ferner Prozeduraufrufe und -antworten.

SOAP kann mit anderen Protokollen wie HTTP verwendet werden.

Die Spezifikation für SOAP 1.1 ist unter Simple Object Access Protocol (SOAP) 1.1 veröffentlicht.

Die Spezifikation für SOAP 1.2 ist hier veröffentlicht:

SOAP Version 1.2 Part 0: Primer

SOAP Version 1.2 Part 1: Messaging Framework

SOAP Version 1.2 Part 2: Adjuncts

SOAP-Intermediär

Ein SOAP-Knoten, der sowohl ein SOAP-Empfänger als auch ein SOAP-Absender ist und aus einer SOAP-Nachricht erreicht werden kann. Er verarbeitet die an ihn gerichteten SOAP-Headerblöcke und leitet eine SOAP-Nachricht an einen letzten SOAP-Empfänger weiter.

SOAP-Nachrichtenpfad

Die Gruppe von SOAP-Knoten, die eine einzelne SOAP-Nachricht durchläuft. Diese Knoten enthalten den ursprünglichen SOAP-Absender, keine oder mehrere SOAP-Intermediäre und einen letzten SOAP-Empfänger.

SOAP-Knoten

Die Verarbeitungslogik einer SOAP-Nachricht.

SOAP-Empfänger

Ein SOAP-Knoten, der eine SOAP-Nachricht akzeptiert.

SOAP-Absender

Ein SOAP-Knoten, der eine SOAP-Nachricht überträgt.

Letzter SOAP-Empfänger

Der SOAP-Empfänger, der das endgültige Ziel einer SOAP-Nachricht ist. Es ist verantwortlich für die Verarbeitung der Inhalte des SOAP-Hauptteils und aller SOAP-Headerblöcke, die an ihn adressiert sind.

UDDI Siehe Universal Description, Discovery and Integration.

Universal Description, Discovery and Integration

Universal Description, Discovery and Integration (UDDI) ist eine Spezifikation für verteilte webbasierte Informationsregistries von Web-Services. UDDI ist außerdem ein öffentlich zugänglicher Satz von Implementierungen der Spezifikation, die es Unternehmen ermöglichen, Informationen zu

den Web-Services, die sie anbieten, festzuhalten, damit andere Unternehmen diese finden können. Die Spezifikation wird von OASIS veröffentlicht.

Web-Service

Ein Softwaresystem, das für die Unterstützung einer interoperablen Maschine-to-Machine-Interaktion über ein Netz konzipiert wurde. Er verfügt über eine Schnittstelle, die in einem maschinenverarbeitbaren Format (Web Service Description Language bzw. WSDL) beschrieben ist.

Web-Service-Adressierung

Die Web-Service-Adressierung (WS-Addressing) stellt einen transportneutralen Mechanismus zur Adressierung von Web-Services und Nachrichten bereit.

Die Spezifikationen für WS-Addressing sind hier veröffentlicht:

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

Web Services Atomic Transaction

Eine Spezifikation, die die Definition eines Koordination vom Typ atomarer Transaktion bereitstellt, die zum Koordinieren von Aktivitäten mit einer Alles-oder-nichts-Eigenschaft verwendet wird.

Die Spezifikation ist von OASIS unter Web Services Atomic Transaction veröffentlicht.

Web-Service-Bindungsdatei

Eine Datei, die einer WEBSERVICE-Ressource zugeordnet ist, die Informationen enthält, die CICS zum Zuordnen von Daten zwischen Eingabe- und Ausgabenachrichten sowie von Anwendungsdatenstrukturen verwendet.

Web-Service-Beschreibung

Ein XML-Dokument, über das ein Service-Provider die Spezifikationen für den Aufruf eines Web-Service an einen Service-Requester kommuniziert. Web-Service-Beschreibungen werden in WSDL (Web Service Description Language) geschrieben.

Web Service Description Language

Eine XML-Anwendung zum Beschreiben von Web-Services. Sie wurde konzipiert, um die Beschreibungen der abstrakten Funktionen, die von einem Service angeboten werden, und die konkreten Details eines Service, z. B. wie und wo diese Funktionen angeboten werden, voneinander zu trennen.

Die Spezifikation ist unter Web Services Description Language (WSDL) veröffentlicht.

Web Services Security

Eine Gruppe von Erweiterungen für SOAP-Messaging, die Nachrichtenintegrität und -vertraulichkeit gewährleisten. Die Spezifikation ist von OASIS unter Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) veröffentlicht.

WS-Atomic Transaction

Siehe Web Services Atomic Transaction.

WS-I Basic Profile

Eine Gruppe von nicht proprietären Web-Service-Spezifikationen, sowie Erläuterungen und Ergänzungen dieser Spezifikationen, die zusammen die Interoperabilität zwischen verschiedenen Implementierungen von Web-Ser-

vices fördern. Das Profil ist von der Web Services Interoperability Organization (WS-I) definiert und Version 1.0 ist unter Web Services Interoperability Organization (WS-I) Basic Profile 1.0 verfügbar.

WSDL

Siehe Web Service Description Language.

WSS Siehe Web Services Security.

XML Extensible Markup Language.

Die Spezifikationen für XML sind hier veröffentlicht:

SOAP Version 1.2 Part 0: Primer

SOAP Version 1.2 Part 1: Messaging Framework

SOAP Version 1.2 Part 2: Adjuncts

XML-Namensbereich

Eine Sammlung von Namen, angegeben durch eine URI-Referenz, die in XML-Dokumenten als Elementtypen und Attributnamen verwendet werden.

XML-Schema

Ein XML-Dokument, das die Struktur anderer XML-Dokumente beschreibt und die Inhalte dieser einschränkt.

XML Schema Definition Language

Eine XML-Syntax zum Schreiben von XML-Schemas, empfohlen vom World Wide Web Consortium (W3C).

CICS- und SOAP-Web-Services

CICS unterstützt zwei unterschiedliche Ansätze zur Implementierung Ihrer CICS-Anwendungen in einer Web-Service-Umgebung. Ein Ansatz ermöglicht eine schnelle Implementierung mit möglichst geringem Programmieraufwand. Der andere Ansatz bietet Ihnen umfassende Flexibilität und Kontrolle über Ihre Web-Service-Anwendungen, mithilfe von Code, den Sie selbst für Ihre spezifischen Anforderungen schreiben. Beide Ansätze werden von einer Infrastruktur unterstützt, die ein oder mehrere *Pipeline-* und *Nachrichtenhandlerprogramme* enthält, das Web-Service-Anforderungen und -Antworten verarbeiten.

Wenn Sie Ihre CICS-Anwendungen in einer Web-Service-Umgebung implementieren, haben Sie folgende Optionen:

- Verwenden Sie den CICS-Web-Service-Assistenten als Unterstützung beim Implementieren einer Anwendung mit möglichst geringem Programmieraufwand.

Wenn Sie beispielsweise eine vorhandene Anwendung als Web-Service verfügbar machen möchten, können Sie mit einer Datenstruktur einer höheren Programmiersprache beginnen und die Web-Service-Beschreibung generieren. Wenn Sie mit einem vorhandenen Web-Service kommunizieren möchten, können Sie alternativ mit der zugehörigen Web-Service-Beschreibung beginnen und eine Struktur einer höheren Programmiersprache generieren, die Sie in Ihren Programmen verwenden können.

Der CICS-Web-Service-Assistent generiert auch die CICS-Ressourcen, die Sie benötigen, um Ihre Anwendung zu implementieren. Und wenn Ihre Anwendung ausgeführt wird, setzt CICS Ihre Anwendungsdaten bei der Ausgabe in eine SOAP-Nachricht um und die SOAP-Nachricht bei Eingabe zurück in Anwendungsdaten um.

- Haben Sie die vollständige Kontrolle über die Verarbeitung Ihrer Daten, indem Sie Ihren eigenen Code schreiben, um Ihre Anwendungsdaten der Nachricht, die zwischen dem Service-Requester- und dem Service-Provider übergeben wird, zuzuordnen.

Wenn Sie beispielsweise Nicht-SOAP-Nachrichten in der Web-Service-Infrastruktur verwenden möchten, können Sie Ihren eigenen Code schreiben, um zwischen dem Nachrichtenformat und dem von Ihrer Anwendung verwendeten Format umzusetzen.

Unabhängig von dem Ansatz, für den Sie sich entscheiden, können Sie Ihre eigenen Nachrichtenhandler für eine zusätzliche Verarbeitung Ihrer Anforderungs- und Antwortnachrichten verwenden, oder Sie können von CICS-bereitgestellte Nachrichtenhandler verwenden, die speziell zur Unterstützung der Verarbeitung Ihrer SOAP-Nachrichten geschrieben wurden.

SOAP-Knoten

Ein SOAP-Knoten ist die Verarbeitungslogik, die für eine SOAP-Nachricht ausgeführt wird.

Ein SOAP-Knoten kann diese Operationen ausführen:

- SOAP-Nachricht übertragen
- SOAP-Nachricht empfangen
- SOAP-Nachricht verarbeiten
- SOAP-Nachricht weiterleiten

Ein SOAP-Knoten kann einer der folgenden Typen sein:

SOAP-Absender

Ein SOAP-Knoten, der eine SOAP-Nachricht überträgt.

SOAP-Empfänger

Ein SOAP-Knoten, der eine SOAP-Nachricht akzeptiert.

Ursprünglicher SOAP-Absender

Der SOAP-Absender, der eine SOAP-Nachricht ab dem Ausgangspunkt eines SOAP-Nachrichtenpfads erstellt.

SOAP-Intermediär

Ein SOAP-Empfänger ist sowohl ein SOAP-Empfänger als auch ein SOAP-Absender und kann aus einer SOAP-Nachricht erreicht werden. Er verarbeitet die an ihn gerichteten SOAP-Headerblöcke und leitet eine SOAP-Nachricht an einen letzten SOAP-Empfänger weiter.

Letzter SOAP-Empfänger

Der SOAP-Empfänger, der das endgültige Ziel einer SOAP-Nachricht ist. Es verarbeitet die Inhalte des SOAP-Hauptteils und aller SOAP-Headerblöcke, die an ihn adressiert sind. Unter bestimmten Umständen kommt eine SOAP-Nachricht möglicherweise nicht bei ihrem letzten SOAP-Empfänger an, z. B. aufgrund eines Problems bei einem SOAP-Intermediär.

Service-Provider-Pipeline

In einer Service-Provider-Pipeline empfängt CICS eine Anforderung, die über eine Pipeline zum Zielanwendungsprogramm übergeben wird. Die Antwort der Anwendung wird über dieselbe Pipeline an den Service-Requester zurückgegeben.

Wenn CICS die Rolle des Service-Providers ausfüllt, führt es die folgenden Operationen aus:

1. Empfangen der Anforderung vom Service-Requester.

2. Überprüfen der Anforderung und Extrahieren der Inhalte, die für das Zielanwendungsprogramm relevant sind.
3. Aufrufen des Anwendungsprogramms, wobei aus der Anforderung extrahierte Daten übergeben werden.
4. Wenn das Anwendungsprogramm die Steuerung zurückgibt, erstellen einer Antwort anhand von Daten, die vom Anwendungsprogramm zurückgegeben werden.
5. Senden einer Antwort an den Service-Requester.

Abb. 1 stellt eine Pipeline mit drei Nachrichtenhandlern in einer Service-Provider-Einstellung dar:

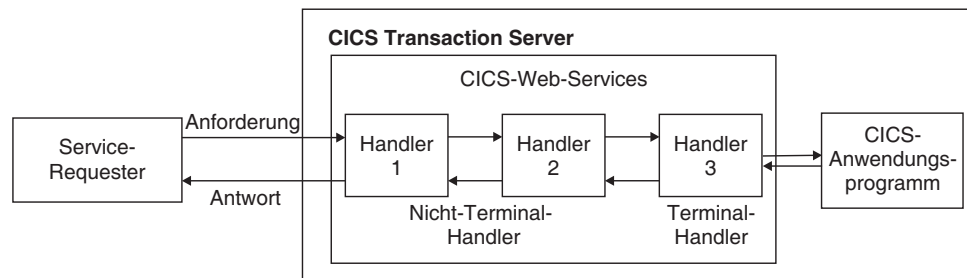


Abbildung 1. Service-Provider-Pipeline

1. CICS empfängt eine Anforderung vom Service-Requester. Es übergibt die Anforderung an den Nachrichtenhandler 1.
2. Nachrichtenhandler 1 führt einige Verarbeitungsschritte aus und übergibt die Anforderung an Handler 2. (Um genau zu sein, gibt er die Steuerung an CICS zurück, das die Pipeline verwaltet. CICS übergibt anschließend die Steuerung an den nächsten Nachrichtenhandler).
3. Nachrichtenhandler 2 empfängt die Anforderung von Handler 1, führt einige Verarbeitungsschritte aus und übergibt die Anforderung an Handler 3.
4. Nachrichtenhandler 3 ist der Terminal-Handler der Pipeline. Er verwendet die Informationen in der Anforderung, um das Anwendungsprogramm aufzurufen. Anschließend verwendet er die Ausgabe des Anwendungsprogramms, um eine Antwort zu generieren, die er an Handler 2 zurückgibt.
5. Nachrichtenhandler 2 empfängt die Antwort von Handler 3, führt einige Verarbeitungsschritte aus und übergibt sie an Handler 1.
6. Nachrichtenhandler 1 empfängt die Antwort von Handler 2, führt einige Verarbeitungsschritte aus und übergibt sie an den Service-Requester.

Service-Requester-Pipeline

In einer Service-Requester-Pipeline erstellt ein Anwendungsprogramm eine Anforderung, die dann über eine Pipeline an den Service-Provider übergeben wird. Die Antwort des Service-Providers wird über dieselbe Pipeline an das Anwendungsprogramm zurückgegeben.

Wenn CICS die Rolle des Service-Requesters ausfüllt, führt es die folgenden Operationen aus:

1. Verwenden von Daten, die dem Anwendungsprogramm bereitgestellt wurden, um eine Anforderung zu erstellen.

2. Senden der Anforderung an den Service-Provider.
3. Empfangen einer Antwort vom Service-Provider.
4. Überprüfen der Antwort und Extrahieren der Inhalte, die für das Ursprungsanwendungsprogramm relevant sind.
5. Zurückgeben der Steuerung an das Anwendungsprogramm.

Abb. 2 stellt eine Pipeline mit drei Nachrichtenhandlern in einer Service-Requester-Einstellung dar:

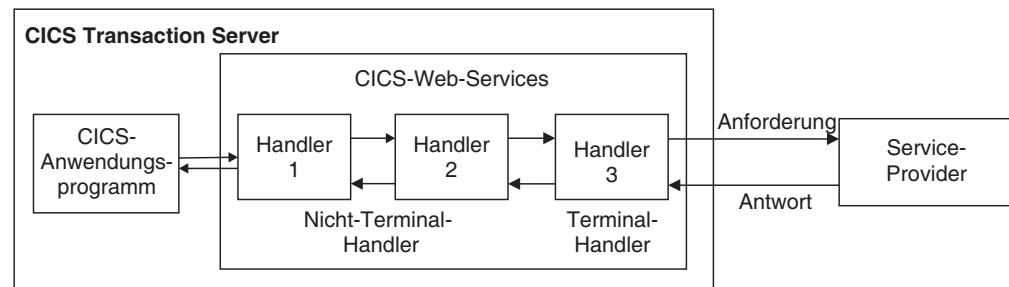


Abbildung 2. Service-Requester-Pipeline

1. Ein Anwendungsprogramm erstellt eine Anforderung.
2. Nachrichtenhandler 1 empfängt die Anforderung vom Anwendungsprogramm, führt einige Verarbeitungsschritte aus und übergibt sie an Handler 2. (Um genau zu sein, gibt er die Steuerung an CICS zurück, das die Pipeline verwaltet. CICS übergibt anschließend die Steuerung an den nächsten Nachrichtenhandler).
3. Nachrichtenhandler 2 empfängt die Anforderung von Handler 1, führt einige Verarbeitungsschritte aus und übergibt die Anforderung an Handler 3.
4. Nachrichtenhandler 3 empfängt die Anforderung von Handler 2, führt einige Verarbeitungsschritte aus und übergibt sie an den Service-Provider.
5. Nachrichtenhandler 3 empfängt die Antwort von Service-Provider, führt einige Verarbeitungsschritte aus und übergibt sie an Handler 2.
6. Nachrichtenhandler 2 empfängt die Antwort von Handler 3, führt einige Verarbeitungsschritte aus und übergibt sie an Handler 1.
7. Nachrichtenhandler 1 empfängt die Antwort von Handler 2, führt einige Verarbeitungsschritte aus und übergibt sie an das Anwendungsprogramm.

CICS-Pipelines und SOAP

Die Pipeline, die von CICS verwendet wird, um Web-Service-Anforderungen und -Antworten zu verarbeiten, ist eine generische Pipeline insofern als dass es wenige Einschränkungen für mögliche Verarbeitungsprozesse in den einzelnen Nachrichtenhandlern gibt. Allerdings verwenden viele Web-Service-Anwendungen SOAP-Nachrichten und jede Verarbeitung dieser Nachrichten sollte mit der SOAP-Spezifikation konform sein. Deshalb stellt CICS spezielle *SOAP-Nachrichtenhandlerprogramme* bereit, die Ihnen helfen können, Ihre Pipeline als SOAP-Knoten zu konfigurieren.

- Eine Pipeline kann für die Verwendung in einem Service-Requester oder in einem Service-Provider konfiguriert werden:
 - Eine Service-Requester-Pipeline ist der ursprüngliche SOAP-Absender für die Anforderung und der letzte SOAP-Empfänger für die Antwort.
 - Eine Service-Provider-Pipeline ist der letzte SOAP-Empfänger für die Anforderung und der ursprüngliche SOAP-Absender für die Antwort.

Sie können keine CICS-Pipeline konfigurieren, um als SOAP-Intermediär zu fungieren.

- Eine Service-Requester-Pipeline kann so konfiguriert werden, dass sie SOAP 1.1 oder SOAP 1.2, aber nicht beides, unterstützt. Eine Service-Provider-Pipeline kann jedoch für die Unterstützung sowohl von SOAP 1.1 als auch von SOAP 1.2 konfiguriert werden. In Ihrem CICS-System können viele Pipelines vorhanden sein, von denen einige SOAP 1.1 oder SOAP 1.2 und einige beides unterstützen.
- Sie können eine CICS-Pipeline für mehr als einen SOAP-Nachrichtenhandler konfigurieren.
- Die von CICS bereitgestellten SOAP-Nachrichtenhandler können so konfiguriert werden, dass sie eine oder mehrere benutzergeschriebene Routinen zur Headerverarbeitung aufrufen.
- Die von CICS-bereitgestellten SOAP-Nachrichtenhandler können so konfiguriert werden, dass manche Aspekte der Kompatibilität mit WS-I Basic Profile Version 1.1 und das Vorhandensein bestimmter Header in der SOAP-Nachricht erzwungen werden.

Die SOAP-Nachrichtenhandler und ihre zugehörigen Routinen zur Headerverarbeitung werden in der Pipelinekonfigurationsdatei angegeben.

SOAP-Nachrichtenpfad

Der SOAP-Nachrichtenpfad ist die Reihe von SOAP-Knoten, die eine einzelne SOAP-Nachricht passiert, darunter der ursprüngliche SOAP-Absender, keine oder mehrere SOAP-Intermediäre und ein letzter SOAP-Empfänger.

Im einfachsten Fall wird eine SOAP-Nachricht zwischen zwei Knoten übertragen, d. h. von einem *SOAP-Absender* an einen *SOAP-Empfänger*. In komplexeren Fällen können Nachrichten jedoch von *SOAP-Intermediären* verarbeitet werden, die eine SOAP-Nachricht empfangen und sie anschließend an den nächsten Knoten senden. Abb. 3 auf Seite 9 zeigt ein Beispiel eines SOAP-Nachrichtenpfads an, in dem eine SOAP-Nachricht vom ursprünglichen SOAP-Absenderknoten an den letzten SOAP-Empfängerknoten übertragen wird und dabei zwei SOAP-Zwischknoten passiert.

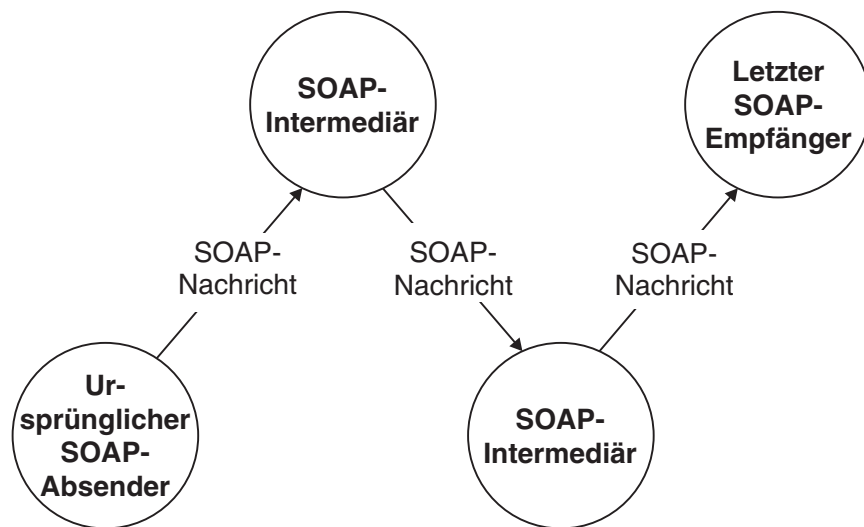


Abbildung 3. Beispiel eines SOAP-Nachrichtenpfads

Ein SOAP-Intermediär ist sowohl ein SOAP-Empfänger als auch ein SOAP-Absender. Er kann (und muss in einigen Fällen) die Headerblöcke in der SOAP-Nachricht verarbeiten und leitet die SOAP-Nachricht an ihren letzten Empfänger weiter.

Der *letzte SOAP-Empfänger* ist das endgültige Ziel einer SOAP-Nachricht. Er verarbeitet sowohl die Headerblöcke als auch den SOAP-Hauptteil. Unter bestimmten Umständen kommt eine SOAP-Nachricht möglicherweise nicht bei ihrem letzten SOAP-Empfänger an, z. B. aufgrund eines Problems bei einem SOAP-Intermediär.

SOAP-Nachrichten und die Anwendungsdatenstruktur

In vielen Fällen kann der CICS-Web-Service-Assistent den Code für die Umsetzung von Daten zwischen einer übergeordneten Datenstruktur, die in einem Anwendungsprogramm verwendet wird, und dem Inhalt des <Body>-Elements einer SOAP-Nachricht generieren. In diesen Fällen müssen Sie, wenn Sie Ihr Anwendungsprogramm schreiben, den SOAP-Hauptteil nicht parsen oder erstellen. CICS übernimmt dies für Sie.

Zum Umsetzen der Daten benötigt CICS zur Laufzeit Informationen zur Anwendungsdatenstruktur und zum Format der SOAP-Nachrichten. Diese Informationen werden in zwei Dateien gespeichert.

- In der Web-Service-Bindungsdatei

Diese Datei wird vom CICS-Web-Service-Assistenten aus der Datenstruktur einer Anwendungssprache mithilfe des Dienstprogramms DFHLS2WS oder aus einer Web-Service-Beschreibung mithilfe des Dienstprogramms DFHWS2LS generiert. CICS verwendet die Bindungsdatei, um die von der Web-Service-Anwendung verwendeten Ressourcen zu generieren und um die Zuordnung zwischen der Datenstruktur der Anwendung und den SOAP-Nachrichten auszuführen.

- In der Web-Service-Beschreibung

Dies kann eine vorhandene Web-Service-Beschreibung sein oder sie kann aus der Datenstruktur einer Anwendungssprache mithilfe des Dienstprogramms

DFHLS2WS generiert werden. CICS verwendet die Web-Service-Beschreibung, um eine umfassende Validierung von SOAP-Nachrichten auszuführen.

Abb. 4 zeigt, wo diese Dateien in einem Service-Provider verwendet werden.

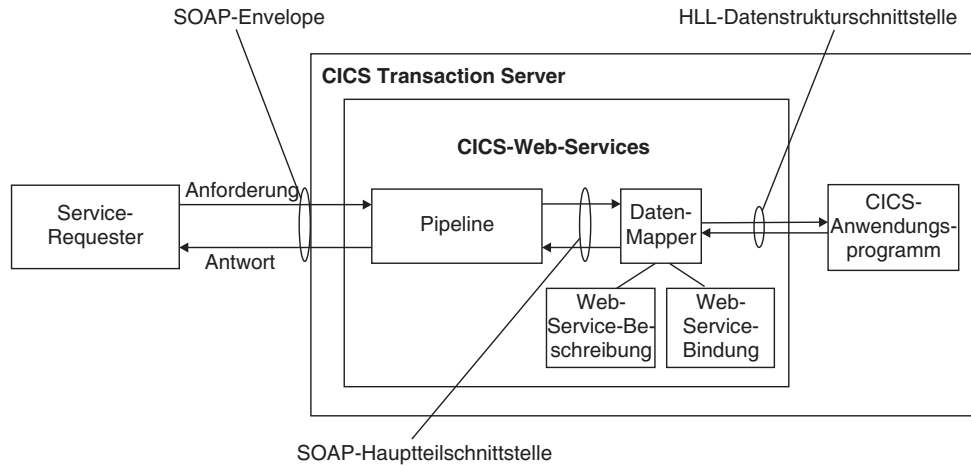


Abbildung 4. SOAP-Hauptteil zur Anwendungsdatenstruktur in einem Service-Provider zuordnen

Ein Nachrichtenhandler in der Pipeline (in der Regel ein von CICS bereitgestellter SOAP-Nachrichtenhandler) entfernt den SOAP-Envelope aus einer eingehenden Anforderung und übergibt den SOAP-Hauptteil an die Datenzuordnungsfunktion. Diese verwendet die Web-Service-Bindungsdatei zum Zuordnen der Inhalte des SOAP-Hauptteils zur Datenstruktur der Anwendung. Wenn die vollständige Validierung der SOAP-Nachricht aktiv ist, wird der SOAP-Hauptteil gegen die Web-Service-Beschreibung validiert. Im Fall einer ausgehenden Antwort wird der Prozess umgekehrt.

Abb. 5 auf Seite 11 zeigt, wo diese Dateien in einem Service-Requester verwendet werden.

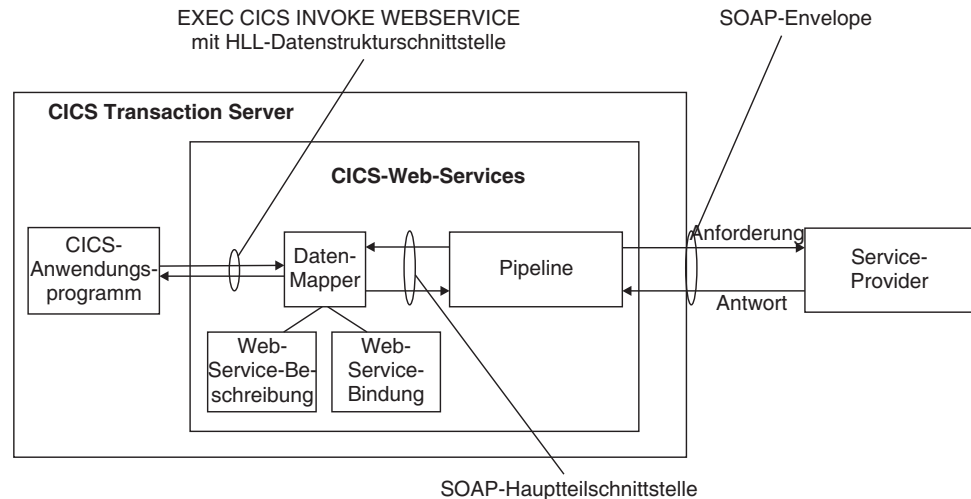


Abbildung 5. SOAP-Hauptteil zur Anwendungsdatenstruktur in einem Service-Requester zuordnen

Für eine ausgehende Anforderung erstellt die Datenzuordnungsfunktion einen SOAP-Hauptteil aus der Anwendungsdatenstruktur, wobei sie auf Informationen aus der Web-Service-Bindungsdatei zurückgreift. Ein Nachrichtenhandler in der Pipeline (in der Regel ein von CICS bereitgestellter SOAP-Nachrichtenhandler) fügt den SOAP-Envelope hinzu. Im Fall einer eingehenden Antwort wird der Prozess umgekehrt. Wenn die vollständige Validierung der SOAP-Nachricht aktiv ist, wird der eingehende SOAP-Hauptteil gegen die Web-Service-Beschreibung validiert.

In beiden Fällen wird die Ausführungsumgebung, die es einem bestimmten CICS-Anwendungsprogramm ermöglicht, in einer Web-Service-Einstellung zu arbeiten, durch drei Objekte definiert. Dabei handelt es sich um die Pipeline, die Web-Service-Bindungsdatei und die Web-Service-Beschreibung. Die drei Objekte sind in CICS als Attribute der WEBSERVICE-Ressourcendefinition definiert.

Es gibt Situationen, in denen Sie, selbst wenn Sie SOAP-Nachrichten verwenden, die Umsetzung nicht verwenden können, die der CICS-Web-Service-Assistent generiert:

- Wenn nicht dieselben Daten in der SOAP-Nachricht und in der höheren Programmiersprache dargestellt werden können.

Alle höheren Programmiersprachen, die von CICS unterstützt werden, und das XML-Schema unterstützen eine Vielzahl unterschiedlicher Datentypen. Es besteht jedoch keine Eins-zu-eins-Entsprechung zwischen den in den höheren Programmiersprachen verwendeten Datentypen und solchen, die im XML-Schema verwendet werden. Und es gibt Fälle, in denen Daten in der Programmiersprache, aber nicht im XML-Schema dargestellt werden können und umgekehrt. Ziehen Sie in diesen Situationen folgende Möglichkeiten in Betracht:

- Ändern Sie Ihre Anwendungsdatenstruktur. Dies ist unter Umständen nicht möglich, da möglicherweise Änderungen am Anwendungsprogramm erforderlich sind.
- Erstellen Sie ein Wrapperprogramm, das die Anwendungsdaten in ein Format bringt, das dann von CICS in einen SOAP-Nachrichtenhauptteil umgesetzt werden kann. Wenn Sie dies tun, können Sie Ihr Anwendungsprogramm un-

verändert lassen. In diesem Fall interagiert die CICS-Web-Service-Unterstützung direkt mit dem Wrapperprogramm und nur indirekt mit dem Anwendungsprogramm.

- Wenn Ihr Anwendungsprogramm in einer Sprache geschrieben wurde, die vom CICS-Web-Service-Assistenten nicht unterstützt wird.

Ziehen Sie in diesen Situationen folgende Möglichkeiten in Betracht:

- Erstellen Sie ein Wrapperprogramm, das in einer Sprache geschrieben ist, die vom CICS-Web-Service-Assistenten unterstützt wird (COBOL, PL/I, C oder C++).
- Statt den CICS-Web-Service-Assistenten zu verwenden, schreiben Sie Ihr eigenes Programm, um die Zuordnung zwischen den SOAP-Nachrichten und der Datenstruktur des Anwendungsprogramms auszuführen.

WSDL und die Anwendungsdatenstruktur

Eine Web-Service-Beschreibung enthält abstrakte Darstellungen der Eingabe- und Ausgabenachrichten, die vom Service verwendet werden. CICS verwendet die Web-Service-Beschreibung, um die von Anwendungsprogrammen verwendeten Datenstrukturen zu erstellen. Zur Laufzeit führt CICS die Zuordnung zwischen den Anwendungsdatenstrukturen und den Nachrichten durch.

Die Beschreibung eines Web-Service enthält unter anderem Folgendes:

- Eine oder mehrere Operationen
- Pro Operation eine Eingabenachricht und eine optionale Ausgabenachricht
- Pro Nachricht die Nachrichtenstruktur, definiert in Bezug auf XML-Datentypen. Komplexe Datentypen, die in den Nachrichten verwendet werden, werden in einem XML-Schema definiert, das im Element `<types>` in der Web-Service-Beschreibung enthalten ist. Einfache Nachrichten können ohne das Element `<types>` beschrieben werden.

WSDL enthält eine abstrakte Definition einer Operation und die zugehörigen Nachrichten. Sie kann nicht direkt in einem Anwendungsprogramm verwendet werden. Um die Operation zu implementieren, muss ein Service-Provider Folgendes tun:

- Er muss die WSDL parsen, um die Struktur der Nachrichten zu verstehen.
- Er muss die einzelnen Eingabenachrichten parsen und die Ausgabenachricht erstellen.
- Er muss die Zuordnungen zwischen den Inhalten der Eingabe- und Ausgabenachrichten und den im Anwendungsprogramm verwendeten Datenstrukturen durchführen.

Ein Service-Requester muss dasselbe tun, um die Operation aufzurufen.

Wenn Sie den CICSWeb-Service-Assistenten verwenden, wird vieles davon für Sie übernommen und Sie können Ihr Anwendungsprogramm schreiben, ohne über detaillierte Kenntnisse zu WSDL oder zur Erstellung von Eingabe- und Ausgabenachrichten zu verfügen.

Der CICS-Web-Service-Assistent besteht aus zwei Dienstprogrammen:

DFHWS2LS

Dieses Dienstprogramm verwendet eine Web-Service-Beschreibung als Ausgangspunkt. Es nutzt die Beschreibungen der Nachrichten und die in die-

sen Nachrichten verwendeten Datentypen, um Datenstrukturen höherer Programmiersprachen zu erstellen, die Sie in Ihren Anwendungsprogrammen verwenden können.

DFHLS2WS

Dieses Dienstprogramm verwendet eine Datenstruktur einer höheren Programmiersprache als Ausgangspunkt. Es nutzt die Struktur, um eine Web-Service-Beschreibung zu erstellen, die Beschreibungen von Nachrichten enthält, und die in diesen Nachrichten verwendeten Datentypen, die aus der Sprachstruktur abgeleitet wurden.

Beide Dienstprogramme generieren eine Web-Service-Bindungsdatei, die von CICS zur Laufzeit verwendet wird, um die Zuordnung zwischen den Datenstrukturen des Anwendungsprogramms und den SOAP-Nachrichten auszuführen.

Beispiel für eine COBOL-zu-WSDL-Zuordnung

Dieses Beispiel zeigt, wie die in einem COBOL-Programm verwendete Datenstruktur in der Web-Service-Beschreibung dargestellt wird, die von dem CICS-Web-Service-Assistenten generiert wird.

Abb. 6 zeigt eine einfache COBOL-Datenstruktur an.

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

Abbildung 6. COBOL-Datensatzdefinition einer in WSDL definierten Eingabenachricht

Die Schlüsselemente in dem entsprechenden Fragment der Web-Service-Beschreibung werden in Abb. 7 auf Seite 14 dargestellt:

```

<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>

```

Abbildung 7. Aus einer COBOL-Datenstruktur abgeleitetes WSDL-Fragment

WSDL und Nachrichtenaustauschmuster

Ein WSDL 2.0-Dokument enthält ein Nachrichtenaustauschmuster (Message Exchange Pattern, MEP), das definiert, wie SOAP 1.2-Nachrichten zwischen dem Web-Service-Requester und dem Web-Service-Provider ausgetauscht werden.

CICS unterstützt vier der acht Nachrichtenaustauschmuster, die in den Spezifikationen *WSDL 2.0 Part 2: Adjuncts* und *WSDL 2.0 Part 2: Additional MEPs* für die Service-Provider- und Service-Requester-Anwendungen definiert sind. Die folgenden MEPs werden unterstützt:

in-only

Eine Anforderungsnachricht wird an den Web-Service-Provider gesendet. Der Provider darf jedoch keine Antwort an den Web-Service-Requester senden.

- Wenn CICS im Providermodus eine Anforderungsnachricht von einem Web-Service empfängt, der das MEP 'in-only' verwendet, gibt es keine

Antwortnachricht zurück. Der Container DFHNORESPONSE wird in den SOAP-Handlerkanal gestellt, um anzugeben, dass die Pipeline keine Antwortnachricht senden darf.

- Im Requestermodus sendet CICS die Anforderungsnachricht an den Web-Service-Provider und wartet nicht auf eine Antwort.

in-out Eine Anforderungsnachricht wird an den Web-Service-Provider gesendet und eine Antwortnachricht wird an den Web-Service-Requester zurückgegeben. Die Antwortnachricht kann eine normale SOAP-Nachricht oder ein SOAP-Fehler sein.

- Wenn CICS im Providermodus eine Anforderungsnachricht von einem Web-Service empfängt, der das MEP 'in-out' verwendet, gibt es eine Antwortnachricht an den Requester zurück.
- Im Requestermodus sendet CICS eine Anforderungsnachricht und wartet auf eine Antwort. Bei dieser Antwort handelt es sich entweder um eine normale Antwortnachricht oder um eine SOAP-Fehlernachricht. Wie lange CICS auf eine Antwort wartet, wird in der Pipeline konfiguriert und gilt für alle Web-Services, die diese Pipeline verwenden. Wenn das Zeitlimit der Anforderung überschritten wird, bevor CICS eine Antwort empfängt, wird ein Fehler an die Service-Requester-Anwendung zurückgegeben.

in-optional-out

Eine Anforderungsnachricht wird an den Web-Service-Provider gesendet und optional wird eine Antwortnachricht an den Web-Service-Requester zurückgegeben. Die Antwort kann eine normale SOAP-Nachricht oder eine SOAP-Fehlernachricht sein.

- Im Providermodus wird die Entscheidung, ob eine SOAP-Antwortnachricht, ein SOAP-Fehler oder gar keine Antwort zurückgegeben werden soll, zur Laufzeit getroffen und hängt von der Anwendungslogik des Service-Providers ab. Wenn CICS keine Antwort an den Web-Service-Requester sendet, wird der Container DFHNORESPONSE in den SOAP-Handlerkanal gestellt, um anzugeben, dass die Pipeline keine Antwortnachricht senden darf. Wird keine Nachricht gesendet, muss die Service-Provider-Anwendung den Container DFHWS-DATA aus dem Kanal löschen.
- Im Requestermodus sendet CICS eine Anforderungsnachricht und wartet auf eine Antwort vom Web-Service-Requester. Wenn das Zeitlimit der Anforderung überschritten wird, bevor eine Antwort empfangen wird, nimmt CICS an, dass die Nachricht erfolgreich empfangen wurde und dass der Provider keine Antwort senden muss. Wie lange CICS auf eine Antwort wartet, wird in der Pipeline konfiguriert und gilt für alle Web-Services, die diese Pipeline verwenden.

robust in-only

Eine Anforderungsnachricht wird an den Web-Service-Provider gesendet und es wird nur dann eine Antwortnachricht an den Web-Service-Requester zurückgegeben, wenn ein Fehler auftritt. Im Fall eines Fehlers wird eine SOAP-Fehlernachricht an den Requester gesendet.

- Wenn die Pipeline im Providermodus die Anforderungsnachricht erfolgreich an die Anwendung übergibt, wird ein Container DFHNORESPONSE in den SOAP-Handlerkanal gestellt, um anzugeben, dass die Pipeline keine Antwortnachricht senden darf. Tritt in der Pipeline ein Fehler auf, wird eine SOAP-Fehlernachricht an den Requester zurückgegeben.
- Im Requestermodus sendet CICS die Anforderungsnachricht an den Web-Service-Provider und wartet für einen festgelegten Zeitraum auf

eine Antwort. Wie lange CICS auf eine Antwort wartet, wird in der Pipeline konfiguriert und gilt für alle Web-Services, die diese Pipeline verwenden. Im Fall einer Zeitlimitüberschreitung nimmt CICS an, dass die Anforderungsnachricht erfolgreich empfangen wurde.

Weitere Informationen zu Nachrichtenaustauschmustern in WSDL 2.0 finden Sie in den folgenden W3C-Spezifikationen:

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

In der Web-Service-Bindungsdatei

Die *Web-Service-Bindungsdatei* enthält Informationen, die CICS verwendet, um Daten zwischen Eingabe- und Ausgabenachrichten sowie Anwendungsdatenstrukturen zuzuordnen.

Eine Web-Service-Beschreibung enthält abstrakte Darstellungen der Eingabe- und Ausgabenachrichten, die vom Service verwendet werden. Wenn eine Service-Provider- oder Service-Requester-Anwendung ausgeführt wird, benötigt CICS Informationen darüber, wie der Inhalt der Nachrichten den von der Anwendung verwendeten Datenstrukturen zugeordnet wird. Diese Informationen werden in einer Web-Service-Bindungsdatei aufbewahrt.

Web-Service-Bindungsdateien werden erstellt:

- Vom Dienstprogramm DFHWS2LS, wenn Sprachstrukturen aus WSDL generiert werden.
- Vom Dienstprogramm DFHLS2WS, wenn WSDL aus einer Sprachstruktur generiert wird.

Zur Laufzeit verwendet CICS Informationen in der Web-Service-Bindungsdatei, um die Zuordnung zwischen Anwendungsdatenstrukturen und SOAP-Nachrichten durchzuführen. Web-Service-Bindungsdateien sind für CICS im WSBIND-Attribut der WEBSERVICE-Ressource definiert.

Externe Standards

CICS-Unterstützung für Web-Services ist mit einer Reihe von Branchenstandards und -spezifikationen konform.

SOAP-Architektur und -Nachrichtenformat

SOAP ist ein Protokoll für den Austausch von Informationen in einer verteilten Umgebung. SOAP-Nachrichten werden als XML-Dokumente verschlüsselt und können mit verschiedenen zugrunde liegenden Protokollen ausgetauscht werden.

Früher ein Akronym für *Simple Object Access Protocol*, wurde SOAP von dem World Wide Web Consortium (W3C) entwickelt und ist in den folgenden von W3C ausgegebenen Dokumenten definiert. In diesen Dokumenten finden Sie vollständige und maßgebliche Informationen zu SOAP.

Simple Object Access Protocol (SOAP) 1.1 (W3C-Notiz)

SOAP Version 1.2 Part 0: Primer (W3C-Empfehlung)

SOAP Version 1.2 Part 1: Messaging Framework (W3C-Empfehlung)

SOAP Version 1.2 Part 2: Adjuncts (W3C-Empfehlung)

Die SOAP-Spezifikationen beschreiben ein verteiltes Verarbeitungsmodell, in dem eine *SOAP-Nachricht* zwischen *SOAP-Knoten*. Die Nachricht stammt von einem *SOAP-Absender* und wird an einen *SOAP-Empfänger* gesendet. Zwischen dem Absender und dem Empfänger kann die Nachricht von einem oder mehreren *SOAP-Intermediären* verarbeitet.

Eine SOAP-Nachricht ist eine unidirektionale Übertragung zwischen SOAP-Knoten, von einem SOAP-Absender an einen SOAP-Empfänger, aber Nachrichten können kombiniert werden, um komplexere Interaktionen zu konstruieren, z. B. Anforderung und Antwort oder Peer-to-Peer-Konversationen.

Die Spezifikation enthält auch diese Informationen:

- Ein Satz von Codierungsregeln zum Ausdrücken von Instanzen von anwendungsdefinierten Datentypen.
- Eine Konvention für die Darstellung ferner Prozeduraufrufe und -antworten.

SOAP-Web-Service-Architektur

Die SOAP-Web-Service-Architektur basiert auf Interaktionen zwischen drei Komponenten: einem Service-Provider, einem Service-Requester und einer optionalen Service-Registry.

Service-Provider

Die Sammlung von Software, die einen Web-Service bereitstellt

- Das Anwendungsprogramm
- Die Middleware
- Die Plattform, auf der dies ausgeführt wird

Service-Requester

Die Sammlung von Software, die verantwortlich ist für das Anfordern eines Web-Service von einem Service-Provider.

- Das Anwendungsprogramm
- Die Middleware
- Die Plattform, auf der dies ausgeführt wird

Service-Registry

Die Service-Registry ist eine zentrale Stelle, an der Service-Provider ihre Servicebeschreibungen veröffentlichen können und Service-Requester diese Servicebeschreibungen finden können.

Die Registry ist eine optionale Komponente der Web-Service-Architektur, da Service-Requester und Service-Provider in vielen Fällen ohne sie miteinander kommunizieren können. Beispielsweise kann die Organisation, die einen Service bereitstellt, die Servicebeschreibung auf vielfältige Weise direkt an die Benutzer des Service verteilen, z. B. indem Sie die Servicebeschreibung zum Download von einer FTP-Site anbieten.

Die Verwendung einer Service-Registry birgt eine Reihe von Vorteilen sowohl für den Requester als auch für den Provider. Beispielsweise kann die Verwendung von IBM® WebSphere Service Registry and Repository (WSRR) den Requester darin unterstützen, Services schneller zu finden, und den Provider darin, eine Versionssteuerung der angebotenen Services durchzusetzen.

CICS stellt direkten Support für die Implementierung von Service-Requester- und Service-Provider-Komponenten bereit. Sie benötigen jedoch zusätzliche Software, um eine Service-Registry in CICS zu implementieren. Wenn Sie IBM WebSphere Service Registry and Repository (WSRR) verwenden, stellt CICS Unterstützung für

WSRR über den Web-Service-Assistenten bereit. Alternativ können Sie eine Service-Registry auf einer anderen Plattform implementieren.

Interaktionen zwischen einem Service-Provider, einem Service-Requester und einer Service-Registry

Die Interaktionen zwischen dem Service-Provider, Service-Requester und der Service-Registry umfassen die folgenden Operationen:

Veröffentlichen

Wenn eine Service-Registry verwendet wird, veröffentlicht ein Service-Provider seine Servicebeschreibung in einer Service-Registry für den Service-Requester.

Suchen

Wenn eine Service-Registry verwendet wird, sucht der Service-Requester die Servicebeschreibung in der Service-Registry.

Binden

Der Service-Requester verwendet die Servicebeschreibung für die Bindung mit dem Service-Provider und die Interaktion mit der Web-Service-Implementierung.

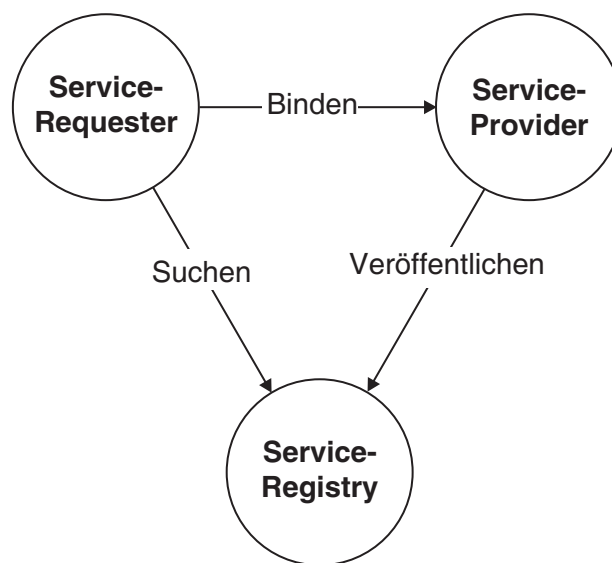


Abbildung 8. Web-Service-Komponenten und Interaktionen

Web-Service-Beschreibung:

Eine Web-Service-Beschreibung ist ein Dokument, in dem der *Service-Provider* die Spezifikationen für den Start des Web-Service an den *Service-Requester* kommuniziert. Web-Service-Beschreibungen werden in der XML-Anwendung namens Web Service Description Language (WSDL) geschrieben.

Die Servicebeschreibung beschreibt den Web-Service so, dass der Umfang an gemeinsam genutztem Wissen und an angepasster Programmierung, das bzw. die erforderlich ist, um die Kommunikation zwischen dem Service-Provider und dem Service-Requester sicherzustellen, minimiert wird. Beispielsweise müssen weder

der Requester noch der Provider wissen, auf welcher Plattform der jeweils andere ausgeführt wird, noch müssen sie die Programmiersprache kennen, in der sie geschrieben sind.

Eine Servicebeschreibung kann entweder der WSDL 1.1- oder der WSDL 2.0-Spezifikation entsprechen. Die beiden Spezifikationen unterscheiden sich hinsichtlich der Terminologie und der Hauptelemente, die in die Servicebeschreibung eingeschlossen werden können. In den folgenden Informationen werden WSDL 1.1-Terminologie und -Elemente verwendet, um den Zweck der Servicebeschreibung zu erläutern.

Die Struktur von WSDL ermöglicht die Aufteilung einer Servicebeschreibung in zwei Definitionen:

- Eine abstrakte *Serviceschnittstellendefinition*, die die Schnittstellen des Service beschreibt und es möglich macht, Programme zu schreiben, die den Service implementieren und starten.
- Eine konkrete *Serviceimplementierungsdefinition*, die die Position im Netz (oder den *Endpunkt*) des Web-Service des Providers und andere implementierungsspezifische Details beschreibt. Sie ermöglicht einem Service-Requester, eine Verbindung zum Service-Provider herzustellen.

Siehe Abb. 9 auf Seite 20.

Ein WSDL 1.1-Dokument verwendet die folgenden Hauptelemente in der Definition von Netzservices:

<types>

Ein Container für Datentypdefinitionen mit einem Typsystem (z. B. als XML-Schema). Definiert die Datentypen, die in der Nachricht verwendet werden. Das Element <types> ist nicht erforderlich, wenn alle Nachrichten aus einfachen Datentypen bestehen.

<message>

Gibt an, welche XML-Datentypen verwendet werden, um die Eingabe- und Ausgabeparameter einer Operation zu definieren.

<portType>

Definiert die Gruppe von Operationen, die von einem oder mehreren Endpunkten unterstützt werden. In einem <portType>-Element ist jede Operation durch ein <operation>-Element beschrieben.

<operation>

Gibt an, welche XML-Nachrichten in den Eingabe- und Ausgabedatenflüssen angezeigt werden können. Eine Operation ist vergleichbar mit einer Methodensignatur in einer Programmiersprache.

<binding>

Beschreibt das Protokoll, das Datenformat, die Sicherheit und andere Attribute für ein bestimmtes <portType>-Element.

<port>

Gibt die Netzadresse eines Endpunkts an und ordnet sie einem <binding>-Element zu.

<service>

Definiert den Web-Service als eine Sammlung zusammengehöriger Endpunkte. Ein <service>-Element enthält mindestens ein <port>-Element.

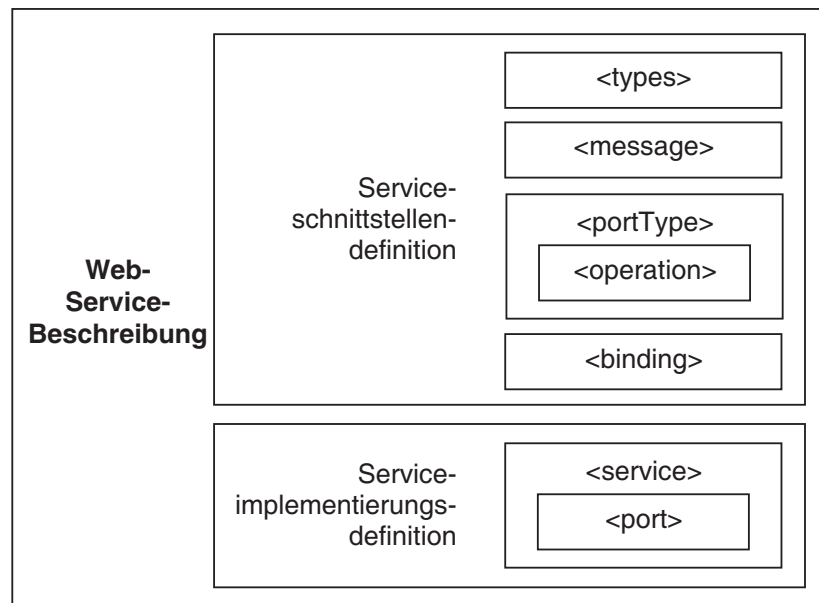


Abbildung 9. Struktur einer Web-Service-Beschreibung

Da Sie die Web-Service-Beschreibung unterteilen können, kann auch die Verantwortung für das Erstellen einer vollständigen Servicebeschreibung geteilt werden. Nehmen wir beispielsweise einen Service, der von einem Standardisierungsgremium für die branchenweite Verwendung definiert und von einzelnen Unternehmen in dieser Branche implementiert wird.

- Das Standardisierungsgremium stellt eine Serviceschnittstellendefinition bereit, die die folgenden Elemente enthält:
 - <types>
 - <message>
 - <portType>
 - <binding>
- Ein Service-Provider, der eine Implementierung des Service anbieten möchte, stellt eine Serviceimplementierungsdefinition bereit, die die folgenden Elemente enthält:
 - <port>
 - <service>

Serviceveröffentlichungen:

Sie können eine Servicebeschreibung mithilfe einer Reihe verschiedener Mechanismen veröffentlichen. Jeder Mechanismus eignet sich für eine andere Situation. CICS unterstützt IBM WebSphere Service Registry and Repository (WSRR) zur Veröffentlichung von Servicebeschreibungen. Alternativ können Sie andere Methoden einsetzen, um eine Servicebeschreibung zu veröffentlichen.

WSSR CICS unterstützt WSRR zur Veröffentlichung von Servicebeschreibungen. Weitere Informationen zu der Unterstützung, die CICS für WSSR bereitstellt, finden Sie im Thema "Interoperabilität zwischen dem Web-Service-Assistenten und WSRR" im Information Center.

Alle folgenden Mechanismen, von denen keiner direkt von CICS unterstützt wird, können mit CICS verwendet werden, um Servicebeschreibungen zu veröffentlichen.

Direkte Veröffentlichung

Dieser Mechanismus ist der direkteste Weg der Veröffentlichung von Servicebeschreibungen: der Service-Provider sendet die Servicebeschreibung direkt an den Service-Requester - in Form eines E-Mail-Anhangs, über eine FTP-Site oder auf einer CD-ROM.

DISCO

Diese proprietären Protokolle stellen einen dynamischen Veröffentlichungsmechanismus bereit. Der Service-Requester verwendet einen einfachen HTTP-GET-Mechanismus zum Abruf einer Web-Service-Beschreibung von einem Netzspeicherort, der vom Service-Provider angegeben und anhand einer URL identifiziert wird.

Universal Description, Discovery and Integration (UDDI)

Eine Spezifikation für verteilte webbasierte Informationsregistries von Web-Services. UDDI ist außerdem ein öffentlich zugänglicher Satz von Implementierungen der Spezifikation, die es Unternehmen ermöglichen, Informationen zu den Web-Services, die sie anbieten, festzuhalten, damit andere Unternehmen diese finden können.

Eine Servicebeschreibung kann bei Bedarf in mehr als einer Form veröffentlicht werden.

Struktur einer SOAP-Nachricht

Eine SOAP-Nachricht ist als XML-Dokument codiert, das aus einem <Envelope>-Element besteht, das wiederum ein optionales <Header>-Element und ein obligatorisches <Body>-Element enthält. Das <Fault>-Element, das im <Body>-Element enthalten ist, dient zum Berichten von Fehlern.

SOAP-Envelope

Der SOAP-<Envelope> ist das Stammelement jeder SOAP-Nachricht. Er enthält zwei untergeordnete Elemente, ein optionales <Header>-Element und ein obligatorisches <Body>-Element.

SOAP-Header

Der SOAP-<Header> ist ein optionales Unterelement des SOAP-Envelopes. Er wird verwendet, um anwendungsbezogene Informationen zu übergeben, die von SOAP-Knoten entlang des Nachrichtenpfads verarbeitet werden sollen.

SOAP-Hauptteil

Der SOAP-Hauptteil (<Body>) ist ein obligatorisches Unterelement des SOAP-Envelopes. Er enthält Informationen für den letzten Empfänger der Nachricht.

SOAP-Fehler

Der SOAP-Fehler (<Fault>) ist ein Unterelement des SOAP-Hauptteils und wird zum Berichten von Fehlern verwendet.

Mit Ausnahme des <Fault>-Elements, das im <Body>-Element einer SOAP-Nachricht enthalten ist, sind XML-Elemente in den <Header>- und <Body>-Elementen durch die Anwendungen definiert, die sie nutzen. Die SOAP-Spezifikation gibt jedoch einige Einschränkungen hinsichtlich ihrer Struktur vor.

Abb. 10 auf Seite 22 zeigt die Hauptelemente einer SOAP-Nachricht.

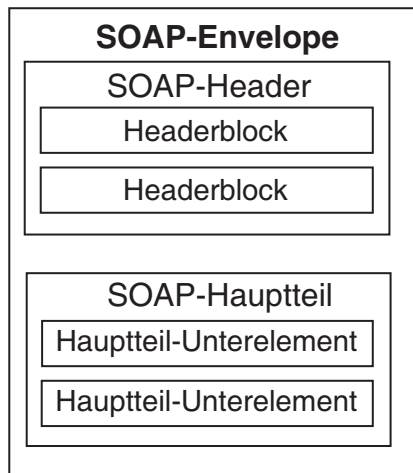


Abbildung 10. Struktur einer SOAP-Nachricht

Abb. 11 auf Seite 23 ist ein Beispiel einer SOAP-Nachricht, die Headerblöcke (die Elemente `<m:reservation>` und `<n:passenger>`) und einen Hauptteil (mit den Elementen `<p:itinerary>` und `<q:lodging>`) enthält.

```

<?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>

```

Abbildung 11. Beispiel einer SOAP 1.2-Nachricht

SOAP-Header:

Das SOAP-Header ist ein optionales Element in einer SOAP-Nachricht. Er wird verwendet, um anwendungsbezogene Informationen zu übergeben, die von SOAP-Knoten entlang des Nachrichtenpfads verarbeitet werden sollen.

Die direkt untergeordneten Elemente des Elements <Header> werden als Headerblöcke bezeichnet. Ein Headerblock ist ein anwendungsdefiniertes XML-Element. Es stellt eine logische Gruppierung von Daten dar, die an SOAP-Knoten gerichtet sein können, die möglicherweise im Pfad einer Nachricht von einem Absender zum letzten Empfänger passiert werden.

SOAP-Headerblöcke können von SOAP-Zwischenknoten und von dem letzten SOAP-Empfängerknoten verarbeitet werden. In einer realen Anwendung verarbeitet jedoch nicht jeder Knoten jeden Headerblock. Vielmehr sind die einzelnen Knoten in der Regel so konzipiert, dass sie bestimmte Headerblöcke verarbeiten, und umgekehrt ist jeder Headerblock zur Verarbeitung durch bestimmte Knoten vorgesehen.

Mit dem SOAP-Header können Features zu einer SOAP-Nachricht dezentral hinzugefügt werden, ohne dass eine Vereinbarung zwischen den kommunizierenden Par-

teilen bestehen muss. SOAP definiert einige Attribute, die verwendet werden können, um anzugeben, wozu ein Feature genutzt wird und ob es optional oder obligatorisch ist. Solche Steuerinformationen sind beispielsweise das Übergeben von Anweisungen oder Kontextinformationen bezüglich der Verarbeitung der Nachricht. So kann eine SOAP-Nachricht auf eine anwendungsspezifische Weise erweitert werden.

Obwohl die Headerblöcke anwendungsdefiniert sind, geben SOAP-definierte Attribute in den Headerblöcken an, wie die Headerblöcke von den SOAP-Knoten verarbeitet werden sollen. Beachten Sie die folgenden wichtigen Attribute:

encodingStyle

Gibt die Regeln zum Codieren der Teile einer SOAP-Nachricht an. SOAP definiert eine restriktivere Gruppe von Regeln zum Codieren von Daten als das sehr flexible XML.

role (SOAP 1.2)

actor (SOAP 1.1)

In SOAP 1.2 gibt das Attribut **role** an, ob ein bestimmter Knoten eine Nachricht bearbeitet. Wenn die für den Knoten angegebene Rolle mit dem Rollenattribut des Headerblocks übereinstimmt, verarbeitet der Knoten den Header. Wenn die Rollen nicht übereinstimmen, verarbeitet der Knoten den Headerblock nicht. In SOAP 1.1 hat das Attribut **actor** dieselbe Funktion.

Rollen können von den Anwendungsdaten definiert werden und werden durch einen URI bezeichnet. Beispiel: `http://example.com/Log` kann die Rolle eines Knotens bezeichnen, der die Protokollierung ausführt. Headerblöcke, die von diesem Knoten verarbeitet werden sollen, geben `env:role="http://example.com/Log"` an, wobei das Namensbereichspräfix `env` dem SOAP-Namensbereichsnamen `http://www.w3.org/2003/05/soap-envelope` zugeordnet ist.

Die SOAP 1.2-Spezifikation definiert drei Standardrollen zusätzlich zu den von der Anwendung definierten.

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

Keiner der SOAP-Knoten im Nachrichtenpfad verarbeitet den Headerblock direkt. Headerblöcke mit dieser Rolle können verwendet werden, um Daten zu übertragen, die für die Verarbeitung anderer SOAP-Headerblöcke erforderlich sind.

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

Alle SOAP-Knoten im Nachrichtenpfad müssen den Headerblock untersuchen, vorausgesetzt, dass der Header nicht an einem früheren Punkt im Nachrichtenpfad entfernt wurde.

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

Nur der letzte Empfängerknoten wird den Headerblock verarbeiten.

mustUnderstand

Dieses Attribut wird verwendet, um sicherzustellen, dass SOAP-Knoten keine Headerblöcke ignorieren, die für den grundlegenden Zweck der Anwendung wichtig sind. Wenn ein SOAP-Knoten mithilfe des Attributs **role** oder **actor** feststellt, dass er einen Headerblock verarbeiten wird, und wenn das Attribut **mustUnderstand** den Wert "true" hat, muss der Knoten den Headerblock in einer Weise verarbeiten, die seiner Spezifikation entspricht, oder er verarbeitet ihn gar nicht (und löst stattdessen einen Fehler aus). Hat das Attribut jedoch den Wert "false", ist der Knoten nicht verpflichtet, den Headerblock zu verarbeiten.

Tatsächlich gibt das Attribut **mustUnderstand** an, ob die Verarbeitung des Headerblocks obligatorisch oder optional ist.

Das Attribut **mustUnderstand** hat die folgenden Werte:

true (SOAP 1.2)

1 (SOAP 1.1)

Der Knoten muss den Headerblock auf eine Weise verarbeiten, die seiner Spezifikation entspricht, oder er verarbeitet ihn gar nicht (und löst einen Fehler aus).

false (SOAP 1.2)

0 (SOAP 1.1)

Der Knoten ist nicht verpflichtet, den Headerblock zu verarbeiten.

relay (nur SOAP 1.2)

Wenn ein SOAP-Zwischenknoten einen Headerblock verarbeitet, entfernt er ihn aus der SOAP-Nachricht. Standardmäßig werden auch alle Headerblöcke entfernt, die ignoriert werden, da das Attribut **mustUnderstand** den Wert "false" hatte. Wenn für das Attribut **relay** jedoch der Wert "true" angegeben ist, behält der Knoten den nicht verarbeiteten Headerblock in der Nachricht bei.

SOAP-Hauptteil:

<Body> ist das obligatorische Element im SOAP-Envelope, in dem die wichtigsten End-to-End-Informationen einer SOAP-Nachricht übertragen werden.

Das Element <Body> und die zugehörigen untergeordneten Elemente werden verwendet, um Informationen zwischen dem ursprünglichen SOAP-Absender und dem letzten SOAP-Empfänger auszutauschen. SOAP definiert ein untergeordnetes Element für das Element <Body>: das Element <Fault>, das zum Berichten von Fehlern verwendet wird. Andere Elemente im Element <Body> werden von dem Web-Service definiert, der sie verwendet.

SOAP-Fehler:

Das SOAP-Element <Fault> transportiert Fehler- und Statusinformationen in der SOAP-Nachricht.

Wenn in einem Web-Service ein Fehler auftritt, wird eine Fehlernachricht an den Client zurückgegeben. Die Basisstruktur der Fehlernachricht wird in den SOAP-Spezifikationen definiert. Jede Fehlernachricht kann XML enthalten, die die spezifische Fehlerbedingung beschreibt. Wenn beispielsweise eine Anwendung in einem CICS-Web-Service abstürzt, wird eine entsprechende Fehlernachricht an den Client zurückgegeben.

CICS kann verschiedene Typen von Fehlernachrichten senden:

- SOAP-Standardfehlernachrichten sind durch die SOAP-Spezifikationen oder durch eine der Web-Service-Spezifikationen definiert, die in CICS unterstützt werden. Die Fehler melden gängige Fehlerbedingungen, z. B. fehlerhafte SOAP-Envelopes.
- SOAP-Fehlernachrichten in Anwendungen werden mithilfe der **EXEC CICS SOAPFAULT**-API-Befehle in Antwort auf Bedingungen generiert, die von der Anwendung erkannt bzw. behandelt werden. Die Struktur dieser Fehlernachrichten ist der Anwendung bekannt, aber nicht CICS.
- Fehlernachrichten des SOAP-Handlers werden von den SOAP-Handlerprogrammen in Antwort auf eine allgemeine Fehlerbehandlung in CICS generiert. Die

SOAP-Handlerprogramme senden beispielsweise SOAP-Fehler bezüglich Ausfällen, XML-Parsingfehlern und anderen gängigen Fehlern.

- Fehlernachrichten der Anwendungshandler werden von CICS-SOAP-Anwendungshandlern in Antwort auf bei der Verarbeitung des Hauptteils einer SOAP-Nachricht auftretende Fehler generiert. Diese Fehler treten während der Umsetzung von XML in binäre Anwendungsdaten oder bei der Generierung der Antwort auf.

Tritt ein Fehler auf, muss das SOAP-Element `<Fault>` ein Hauptteileintrag sein und darf nicht mehr als einmal in einem Element `<Body>` vorkommen. Die XML-Elemente im SOAP-Element `<Fault>` unterscheiden sich in SOAP 1.1 und SOAP 1.2.

SOAP 1.1

In SOAP 1.1 enthält das SOAP-Element `<Fault>` die folgenden Elemente:

`<faultcode>`

Das Element `<faultcode>` ist ein obligatorisches Element im Element `<Fault>`. Es liefert Informationen zu dem Fehler in einem von der Software verarbeitbaren Format. SOAP definiert einen kleinen Satz von SOAP-Fehlercodes, die grundlegende SOAP-Fehler abdecken. Dieser Satz kann von Anwendungen erweitert werden.

`<faultstring>`

Das Element `<faultstring>` ist ein obligatorisches Element im Element `<Fault>`. Es liefert Informationen zu dem Fehler in einem für den menschlichen Leser geeigneten Format.

`<faultactor>`

Das Element `<faultactor>` enthält den URI des SOAP-Knotens, der den Fehler generiert hat. Ein SOAP-Knoten, bei dem es sich nicht um den letzten SOAP-Empfänger handelt, muss das Element `<faultactor>` enthalten, wenn er einen Fehler erzeugt. Der letzte SOAP-Empfänger muss dieses Element nicht enthalten, kann es aber.

`<detail>`

Das Element `<detail>` trägt anwendungsspezifische Fehlerinformationen zum Element `<Body>` in sich. Es muss vorhanden sein, wenn der Inhalt des Elements `<Body>` nicht erfolgreich verarbeitet wurde. Es darf nicht verwendet werden, um Fehlerinformationen zu Headereinträgen zu übertragen. Detaillierte Fehlerinformationen zu Headereinträgen müssen in den Headereinträgen selbst gespeichert sein.

SOAP 1.2

In SOAP 1.2 enthält das SOAP-Element `<Fault>` die folgenden Elemente:

`<Code>`

Das Element `<Code>` ist ein obligatorisches Element im Element `<Fault>`. Es liefert Informationen zu dem Fehler in einem von der Software verarbeitbaren Format. Es enthält ein Element `<Value>` und ein optionales Element `<Subcode>`.

`<Reason>`

Das Element `<Reason>` ist ein obligatorisches Element im Element `<Fault>`. Das Element `<Reason>` enthält mindestens ein `<Text>`-Element, das Informationen zu dem Fehler in einer bestimmten nativen Sprache enthält.

<Node>

Das Element <Node> enthält den URI des SOAP-Knotens, der den Fehler generiert hat. Ein SOAP-Knoten, bei dem es sich nicht um den letzten SOAP-Empfänger handelt, muss das Element <Node> enthalten, wenn er einen Fehler erzeugt. Der letzte SOAP-Empfänger muss dieses Element nicht enthalten, kann es aber.

<Role>

Das Element <Role> enthält einen URI, der die Rolle angibt, welche der Knoten zu dem Zeitpunkt hatte, als der Fehler auftrat.

<Detail>

Das Element <Detail> ist ein optionales Element, das anwendungsspezifische Fehlerinformationen zu den SOAP-Fehlercodes enthält, die den Fehler beschreiben. Das Vorhandensein des Elements <Detail> hat keinen Einfluss darauf, welche Teile der fehlerhaften SOAP-Nachricht verarbeitet wurden.

Beispiel und Schemas eines SOAP-Fehlers

Das folgende Beispiel zeigt eine SOAP-Fehlernachricht, die vom Anwendungshandler, DFHPITP, bei der Verarbeitung des Hauptteils einer SOAP-Nachricht generiert wird.

```
<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">
      DFHPIT1008 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>
```

Der größte Teil des Inhalts in diesem Beispiel ist in allen Fehlernachrichten enthalten. Das Element <Detail> enthält die Informationen, die das Problem eindeutig beschreiben, das der Anwendungshandler erkannt hat. Diese spezifische Fehlermeldung enthält eine Kopie einer Fehlermeldung, die in die CICS-Nachrichtenprotokolle geschrieben wird. Wenn Sie SOAP-Fehler des Anwendungshandlers programmgesteuert parsen möchten, verwenden Sie das folgende XML-Schema:

```
<?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>
```

Die allgemeingültigen Fehlermeldungen sind komplizierter, da der Abschnitt <Detail> auf verschiedene Arten strukturiert sein kann. Wenn Sie SOAP-Fehlerhandler programmgesteuert parsen möchten, verwenden Sie das XML-Schema, das unter *usshome/schemas/soapfault/soapfault.xsd* bereitgestellt wird, wobei *usshome*

der Wert des Systeminitialisierungsparameters **USSHOME** ist.

Verwendung von SOAP-Web-Services planen

Bevor Sie die Verwendung von SOAP-Web-Services in CICS planen können, müssen Sie die folgenden Fragen für die einzelnen Anwendungen in Betracht ziehen.

Before you begin

Möchten Sie Ihre CICS-Anwendung in der Rolle eines Service-Providers oder eines Service-Requesters implementieren?

Sie möchten zwei Anwendungen mithilfe der CICS-Unterstützung für Web-Services verknüpfen. In diesem Fall ist eine Anwendung der Service-Provider und die andere der Service-Requester.

Möchten Sie Ihre eigenen Anwendungsprogramme verwenden oder neue schreiben?

Wenn Ihre vorhandenen Anwendungen mit einer wohl definierten Schnittstelle zur Geschäftslogik konzipiert sind, können Sie sie wahrscheinlich in einer Web-Service-Einstellung verwenden, entweder als Service-Provider oder als Service-Requester. In den meisten Fällen werden Sie jedoch ein Wrapperprogramm schreiben müssen, das Ihre Geschäftslogik mit der Web-Service-Logik verbindet.

Wenn Sie planen, neue Anwendungen zu schreiben, sollten Sie Ihre Geschäftslogik von Ihrer Web-Service-Logik getrennt halten, und auch hier müssen Sie ein Wrapperprogramm schreiben, um diese Trennung zu ermöglichen. Wenn Ihre Anwendung jedoch für Web-Services konzipiert wird, ist der Wrapper möglicherweise einfacher zu schreiben.

Möchten Sie SOAP-Nachrichten verwenden?

SOAP ist von grundlegender Bedeutung für die Web-Service-Architektur. Ein großer Teil der Unterstützung, die in CICS bereitgestellt wird, setzt voraus, dass Sie SOAP verwenden. Es kann jedoch Situationen geben, in denen andere Nachrichtenformate geeigneter sind. Angenommen, Sie haben Ihre eigenen Nachrichtenformate entwickelt, die Sie mit der CICS-Web-Service-Infrastruktur implementieren möchten. Sie können dies in CICS tun, aber Sie werden manche Funktionen, die CICS bereitstellt, z. B. den Web-Service-Assistenten und die SOAP-Nachrichtenhandler nicht verwenden können.

Wenn Sie sich entscheiden, SOAP nicht zu verwenden, übernehmen Ihre Anwendungsprogramme das Parsen eingehender Nachrichten und das Erstellen von ausgehenden Nachrichten.

Möchten Sie den CICSWeb-Service-Assistenten verwenden, um die Zuordnungen zwischen Ihren Datenstrukturen und SOAP-Nachrichten zu generieren?

Der Assistent bietet eine schnelle Implementierung vieler Anwendungen in einer Web-Services-Einstellung mit wenig oder keinem zusätzlichen Programmieraufwand. Und wenn zusätzliche Programmierung erforderlich ist, ist sie üblicherweise wenig komplex und involviert keine Änderungen an vorhandener Geschäftslogik.

Es gibt jedoch Fälle, in denen besser ohne den Web-Service-Assistenten gearbeitet wird. Wenn Sie beispielsweise über vorhandenen Code verfügen, der Datenstrukturen SOAP-Nachrichten zuordnet, besteht kein Vorteil darin, Ihre Anwendung mit dem Web-Service-Assistenten umzustrukturieren.

Der CICS-Web-Service-Assistent unterstützt zwar die gängigsten Datentypen und Strukturen, aber eben nicht alle. In dieser Situation sollten Sie die Liste der nicht unterstützten Datentypen und Strukturen für die jeweilige

Sprache prüfen und in Betracht ziehen, eine Programmebene bereitstellen, die Ihre Anwendungsdaten einem Format zuordnet, das der Assistent unterstützen kann. Ist dies nicht möglich, müssen Sie die Nachricht selbst parsen. Detaillierte Angaben dazu, was der Assistent unterstützt oder nicht unterstützt, finden Sie unter High-level language and XML schema mapping.

Wenn Sie sich entscheiden, den CICS-Web-Service-Assistenten nicht zu verwenden, können Sie ein Toll wie IBM Developer for Z einsetzen, um die erforderlichen Artefakte zu erzeugen, und anschließend können Sie Ihren eigenen Code zum Parsen eingehender Nachrichten und zum Erstellen ausgehender Nachrichten bereitstellen. Sie können auch die bereitgestellte Anbieterschnittstellen-API verwenden.

Möchten Sie eine vorhandene Servicebeschreibung verwenden oder eine neue erstellen?

In manchen Situationen werden Sie eine vorhandene Servicebeschreibung als Ausgangspunkt verwenden müssen. Beispiel:

- Ihre Anwendung ist ein Service-Requester und dafür konzipiert, einen vorhandenen Web-Service aufzurufen.
- Ihre Anwendung ist ein Service-Provider und Sie möchten, dass sie einer vorhandenen branchenüblichen Servicebeschreibung entspricht.

In wieder anderen Situationen müssen Sie möglicherweise eine neue Servicebeschreibung für Ihre Anwendung erstellen.

What to do next

CICS- und JSON-Web-Services

Es gibt mehrere Möglichkeiten für die ersten Schritte mit JSON-Web-Services in CICS. Welche sich für Sie eignen, richtet sich danach, wie viel Sie schon wissen und wie komplex Ihre Pläne für die Verwendung von Web-Services sind.

About this task

CICS unterstützt verschiedene Technologien zum Verfügbarmachen von Ressourcen als JSON-basierte Services. Dieses Thema bezieht sich auf eine ältere Technologie namens „JSON-Web-Service“. Die entsprechenden Informationen zu z/OS Connect for CICS 1.0 finden Sie unter „CICS und z/OS Connect“ auf Seite 38.

JSON-Web-Services ist eine Technologie, mit der CICS-Programme als JSON-Services aktiviert werden können. Dabei kann es sich um REST-konforme Services oder um Anforderung-/Antwort-Services vom Typ Remote Procedure Call handeln. Die daraus abgeleitete Technologie wird für SOAP-Web-Services verwendet und nutzt die JCL-Prozeduren DFHLS2JS und DFHJS2LS zum Generieren von WSBIND-Dateien. Diese WSBIND-Dateien werden in CICS als WEBSERVICE-Ressourcen implementiert. Die WEBSERVICE-Ressourcen vereinfachen die automatische Konvertierung zwischen JSON und Binärdatenformaten von Anwendungen.

Die wichtigste Technologie von IBM für JSON beinhaltet ein Produkt namens z/OS Connect Enterprise Edition, das in hohem Maß kompatibel mit CICS-JSON-Web-Services ist, aber viele zusätzliche Integrationsoptionen und -funktionen bereitstellt.

Weitere Informationen zu dieser Option finden Sie unter Getting started with z/OS Connect. Im Folgenden finden Sie ein paar Einstiegspunkte für JSON-Web-Services in CICS:

Procedure

- Installieren Sie die Beispielanwendung. CICS stellt ein Beispiel für eine Katalogmanagementanwendung bereit, die Sie als JSON-Web-Service-Provider aktivieren können. Verwenden Sie zu diesem Zweck DFHLS2JS, um einen Web-Service aus den bereitgestellten Sprachstrukturen zu generieren. Sie können einen Web-Browser oder eine Clientanwendung eines anderen Anbieters verwenden, um den JSON-Web-Service zu testen. Weitere Informationen finden Sie unter Creating a service provider application from a data structure.

Verwenden Sie die Beispielanwendung, wenn Sie die Verwendung von Web-Services in CICS in der Praxis kennenlernen möchten. Die Beispielanwendung ist unter Beispielanwendung des CICS-Katalogmanagers beschrieben.

- Planen Sie die Implementierung einer Anwendung als Service-Provider. Möglicherweise wissen Sie bereits ausreichend genau, wie Sie Web-Services in CICS verwenden möchten, um mit der Planung Ihrer Anwendungen und der zugehörigen Infrastruktur zu beginnen.
- Eine weitere Option für JSON-Web-Services umfasst die Verwendung von z/OS Connect. Weitere Informationen zu dieser Option finden Sie unter Getting started with z/OS Connect.

Konzepte von JSON-Web-Services

In diesem Thema lernen Sie die Konzepte hinter JSON-Web-Services kennen.

Web-Services

„Web-Service“ ist ein generischer Begriff für eine Softwarefunktion, die an einer adressierbaren Netzposition gehostet wird. In diesem sehr allgemeinen Sinn kann dies einen cloudbasierten Service, einen Dienstprogrammservice oder auch eine Abteilungsanwendung beschreiben. Der Begriff „Web-Service“ kann auch in einem spezifischeren Sinn verwendet werden, z. B. für einen gehosteten Service unter Verwendung von SOAP, der mithilfe eines WSDL-Dokuments beschrieben wird. Es ist diese spezifischere Bedeutung, die der Begriff „Web-Services in CICS“ üblicherweise hat. Allerdings wird der allgemeinere Begriff oft von der JSON-Community zur Beschreibung JSON-basierter Services verwendet. JSON-Web-Services verwenden den Begriff in seiner generischen Bedeutung.

Es gibt einige wichtige Unterschiede zwischen SOAP und JSON:

- Der Inhalt einer SOAP-Nachricht sind XML-Daten, während eine JSON-Nachricht JSON-Daten enthält. JSON und XML sind unterschiedliche Codierungsmechanismen für die Beschreibung strukturierter Daten. JSON ist in der Regel der effizientere Codierungsmechanismus, d. h. eine typische JSON-Nachricht ist kleiner als die entsprechende XML-Nachricht.
- JSON lässt sich einfach in JavaScript-Anwendungen integrieren, XML nicht. Dies führt dazu, dass JSON das bevorzugte Datenformat vieler mobiler Anwendungsentwickler ist.
- SOAP bietet einen Mechanismus zum Hinzufügen von Headern zu einer Nachricht sowie eine Reihe von Spezifikationen für Servicequalitäten (z. B. die Sicherheitskonfiguration und verteilte Transaktionen). JSON stellt diesen Mechanismus nicht bereit, sondern stützt sich stattdessen auf die Services des zugrunde liegenden HTTP-Netzprotokolls. Dies hat zur Folge, dass weniger Optionen für das Sichern und Konfigurieren einer Workload zur Verfügung stehen.
- SOAP-Web-Services werden mithilfe von WSDL-Dokumenten beschrieben. JSON-Web-Services sind weniger formal strukturiert; sie sind eher lose verbunden und bevorzugen eine Dokumentation anhand von Beispielen.

- SOAP-Web-Services haben ein explizites Fehlerformat, das SOAP-Fehlernachrichten einbezieht. Es gibt keine Entsprechung in JSON.

Es gibt auch viele Ähnlichkeiten zwischen JSON und SOAP:

- Die CICS-Implementierung von JSON wird aus der SOAP-Architektur abgeleitet und teilt viele der Konzepte und Artefakte.
- Beide beziehen Offlinedienstprogramme mit ein, die die Zuordnung von Anwendungsdaten zu und von der externen Datendarstellung unterstützen. Für SOAP gibt es DFHLS2WS und DFHWS2LS, für JSON gibt es DFHLS2JS und DFHJS2LS.
- Der Implementierungsmechanismus für beide Technologien umfasst eine PIPELINE-Ressource, eine WEBSERVICE-Ressource und eine URIMAP-Ressource.

JSON-Schema

Ein Nachteil von JSON im Vergleich zu SOAP besteht in der Schwierigkeit, die Struktur einer JSON-Schnittstelle zu dokumentieren. SOAP-Web-Services haben den Vorteil von WSDL-Dokumenten, in Kombination mit XML-Schemas. Ein WSDL-Dokument ist möglicherweise schwierig zu verstehen, aber es stehen viele Tools für die Arbeit mit WSDL-Dokumenten zur Verfügung.

In JSON kommt dem am ehesten die JSON-Schemaspezifikation gleich, die unter <http://json-schema.org/> verfügbar ist. Zum Zeitpunkt des Schreibens dieses Artikels handelt es sich um einen Spezifikationsentwurf, der den IETF-Standardisierungsprozess durchläuft. Der CICS-JSON-Assistent (DFHLS2JS und DFHJS2LS) stellt eine partielle Implementierung von Entwurf 4 dieser neu entwickelten Spezifikation bereit. DFHLS2JS kann verwendet werden, um JSON-Schemas zu generieren, und DFHJS2LS kann verwendet werden, um sie zu verarbeiten.

Mit dem JSON-Schema können Sie die gültige Syntax und das Inhaltsmodell für einen JSON-Web-Service nachvollziehen, der in CICS implementiert wurde. Die JSON-Schemaspezifikation verfügt nicht über dasselbe Tooling-Ökosystem wie die XML-Schemaspezifikation, aber es wird an einer neuen Generation von JSON-Tools gearbeitet, die dieses Datenformat verwenden.

CICS-Implementierung von JSON-basierten Web-Services

CICS unterstützt drei Modi von JSON-Web-Services, z/OS Connect, Anforderung/Antwort und REST-konform. CICS unterstützt auch ein programmgesteuertes Szenario, in dem Anwendungen JSON-Daten in und aus COBOL-Datenformaten selbst umsetzen können.

z/OS Connect

z/OS Connect macht es möglich, CICS-Programme über eine JSON-Schnittstelle (JavaScript Object Notation) aufzurufen. z/OS Connect for CICS wurde zunächst als Alternative für die JSON-Funktionalitäten der JAVA-Pipelines eingeführt, die im CICS TS-Feature-Pack für mobile Erweiterungen bereitgestellt und in CICS TS Version 5.2 integriert wurden. Seitdem hat sich z/OS Connect zu einem eigenständigen Produkt namens z/OS Connect Enterprise Edition entwickelt, das über zusätzliche Funktionalitäten verfügt.

z/OS Connect ist die wichtigste Technologie von IBM für die Implementierung von JSON-Services und -APIs in CICS. Es ist in drei Versionen erhältlich und unterstützt verschiedenen Implementierungsoptionen. z/OS Connect for CICS ist die Einstiegsedition und eine kostengünstige Option,

verfügt aber nicht über die Erweiterungen der anderen Versionen. z/OS Connect EE Version 2.0 und 3.0 bietet eine breitere Auswahl an Integrationsoptionen.

z/OS Connect EE ist ein separat erhältliches IBM Produkt. Es baut auf den Funktionalitäten von z/OS Connect for CICS auf, was auch Unterstützung für JSON-Services einschließt. Mit z/OS Connect EE können API-Entwickler JSON-APIs aus JSON-Services erstellen. Die APIs werden mit dem Eclipse-basierten API-Editor erstellt und gepackt, der mit z/OS Connect EE bereitgestellt wird, und anschließend in der z/OS Connect-Laufzeit implementiert. Das API-Paket enthält Swagger 2.0-Definitionen, um für Entwickler das Einbinden der APIs in ihre Anwendungen zu vereinfachen. Zentrale z/OS Connect-Funktionen wie die Berechtigungssicherheitsprüfung für den Serviceaufruf, das Erstellen von SMF-Datensätzen (System Management Facility, Systemverwaltungsfunktion) und das Protokollieren von REST-konformen Serviceanforderungen gelten auch für die APIs.

Anforderung-Antwort

Das Anforderung-/Antwort-JSON-Muster ist dem von SOAP-basierten Web-Services in CICS sehr ähnlich. Der Web-Service wird mithilfe eines PROGRAM-Elements in CICS implementiert. Das PROGRAM hat Eingabe- und Ausgabedatenformate, die mithilfe von Sprachstrukturen (wie COBOL-Copybooks) beschrieben werden, und CICS ist verfügbar für das Umsetzen eingehender JSON-Nachrichten in Anwendungsdaten und für das Verknüpfen mit der Anwendung. Die Anwendung gibt Ausgabedaten an CICS zurück und CICS setzt diese in JSON-Daten um, um sie an den Client zurückzugeben.

In diesem Szenario muss der JSON-Client unter Verwendung der HTTP-POST-Methode eine Verbindung zu CICS herstellen.

Ein JSON-Web-Service im Anforderung-/Antwort-Modus kann entweder Bottom-up oder Top-down entwickelt werden. Im Bottom-up-Modus wird ein vorhandenes CICS-PROGRAM mithilfe des JSON-Assistenten DFHLS2JS als JSON-Web-Service verfügbar gemacht. Im Top-down-Modus kann ein neuer JSON-Web-Service entwickelt werden, um eine Schnittstelle zu implementieren, die unter Verwendung von vorhandenen JSON-Schemas beschrieben wird. Im Top-down-Modus wird der JSON-Assistent DFHJS2LS verwendet, um neue Sprachstrukturen zu generieren, und es muss eine Anwendung erstellt werden, um diese zu verwenden.

Das Anforderung-/Antwort-Muster kann verwendet werden, um JSON-Web-Services zu erstellen, deren Ziel entweder Kommunikationsbereiche oder über einen Kanal angeschlossene CICS-Programme sind. Ein Anforderung-/Antwort-JSON-Web-Service kann nur im Providermodus verwendet werden (wobei CICS als Server fungiert).

REST-konform

Dieses Szenario unterscheidet sich von dem für SOAP-Web-Services. Das Konzept eines REST-konformen JSON-Web-Service ist umfassender unter Konzepten REST-konformer JSON-Web-Services erläutert. Ein REST-konformer JSON-Web-Service implementiert die Architekturprinzipien des REST-Designmusters (REpresentational State Transfer). Dieses Designmuster ist wahrscheinlich nicht relevant für vorhandene CICS-Anwendungen und deshalb nur im Top-down-Modus verfügbar.

Ein JSON-Schema kann von DFHJS2LS im REST-konformen Modus verarbeitet werden. Es muss eine Anwendung geschrieben werden, um den Ser-

vice zu implementieren, und ihr Verhalten muss sich nach der HTTP-Methode richten, die für die eingehenden Anforderungen verwendet wurde.

CICS implementiert einen reinen Stil von REST-konformen Anwendungen, bei denen das Datenformat für POST (erstellen) GET (abfragen) und PUT (ersetzen) dasselbe ist.

REST-konforme JSON-Web-Service-Anwendungen müssen eine kanalbasierte Programmschnittstelle verwenden. Kommunikationsbereiche werden nicht unterstützt. Ein REST-konformer JSON-Web-Service kann nur im Providermodus verwendet werden (wobei CICS als Server fungiert).

Programmgesteuerter Modus

In diesem Szenario kann eine Anwendung eine Verknüpfung mit einem von CICS bereitgestellten Programm, DFHJSON, einrichten und das Programm anweisen, Anwendungsdaten in JSON-Daten oder JSON-Daten in Anwendungsdaten umzusetzen. Eine Anwendung kann diese Funktion beispielsweise verwenden, um JSON-Daten zu generieren, die an einen fernen JSON-Web-Service gesendet werden sollen. Dazu muss der ferne JSON-Web-Service mithilfe der CICS-Web-API kontaktiert werden.

CICS verfügt nicht über integrierte Unterstützung für JSON-Web-Services im Requestermodus, aber eine Anwendung kann mithilfe des programmgesteuerten Modus einen fernen JSON-Web-Service aufrufen.

Konzepte REST-konformer JSON-Web-Services

In diesem Thema lernen Sie die Konzepte hinter REST-konformen Web-Services kennen.

REST-konforme Web-Services

Representational State Transfer, oder REST, ist ein Designmuster für die Interaktion mit Ressourcen, die auf einem Server gespeichert sind. Jede Ressource hat eine Identität, einen Datentyp und unterstützt eine Reihe von Aktionen.

Das REST-konforme Designmuster wird normalerweise in Kombination mit HTTP verwendet, der Sprache des Internets. In diesem Kontext ist die Identität der Ressource ihr URI, der Datentyp ist ihr Medientyp und die Aktionen sind die HTTP-Standardmethoden (GET, PUT, POST und DELETE).

Diese Art von Service unterscheidet sich von Web-Services vom Typ Anforderung/Antwort.

- Anforderung/Antwort-Services interagieren mit einer Anwendung, während REST-konforme Services typischerweise mit Daten (auch als Ressourcen bezeichnet) interagieren.
- Anforderung/Antwort-Services umfassen anwendungsdefinierte Operationen, aber REST-konforme Services vermeiden anwendungsspezifische Konzepte.
- Anforderung/Antwort-Services haben unterschiedliche Datenformate für jede Nachricht, aber REST-konforme Services verwenden typischerweise dasselbe Datenformat für verschiedene HTTP-Methoden.

Die vier wichtigsten HTTP-Methoden definieren die vier Operationen, die gängigerweise von REST-konformen Services implementiert werden. Die HTTP-POST-Methode wird zum Erstellen einer Ressource verwendet, GET wird zum Abfragen der Ressource verwendet, PUT wird verwendet, um sie zu ändern, und DELETE

wird verwendet, um sie zu zerstören. Die gängigste REST-konforme Architektur umfasst ein für diese vier Operationen gemeinsam genutztes Datenmodell. Dieses Datenmodell definiert die Eingabe für die POST-Methode (erstellen), die Ausgabe für die GET-Methode (abfragen) und die Eingabe für die PUT-Methode (ersetzen). Dieses einfache Designmuster ist in der REST-konformen Community beliebt, es ist aber nicht das einzige REST-konforme Designmuster. Der HTTP-Statuscode wird verwendet, um über Erfolg oder Fehlschlagen der Operation zu informieren. Manche REST-konformen APIs sind auf andere Weise konzipiert.

Eine fünfte HTTP-Methode namens HEAD wird manchmal von REST-konformen Web-Services unterstützt. Diese Methode ist äquivalent zu GET, mit der Ausnahme, dass sie nur HTTP-Header, aber keine Hauptteilaten zurückgibt. Sie wird manchmal verwendet, um das Vorhandensein einer Ressource zu testen. Nicht alle REST-konformen APIs unterstützen die Verwendung der HEAD-Methode.

Traditionelle CICS-Anwendungen stimmen wahrscheinlich nicht mit dem REST-konformen Architekturmuster überein. Typische CICS-Anwendungen implementieren mehrere Operationen, von denen jede über Datenmodelle für Eingabe- und Ausgabeformate verfügt. Diese vorhandenen Operationen stimmen wahrscheinlich nicht direkt mit den vier HTTP-Methoden überein. Aus diesem Grund zielt das REST-konforme Architekturmuster primär auf neue Anwendungen in CICS ab. Um vorhandene CICS-Anwendungen als REST-konforme Services verfügbar zu machen, müssen Sie sie unter Umständen in eine neue Schnittstelle einschließen, die den REST-konformen Prinzipien entspricht.

URI

Die Identität eines REST-konformen Service wird durch den zugehörigen URI angegeben. Ein URI kann aus mehreren Komponenten bestehen, einschließlich Hostname, Portnummer, Pfad und optionale Abfragezeichenfolge. Der Domänenname und die Portnummer zusammen zielen auf eine TCPIPService-Ressource in CICS. Weitere Informationen finden Sie unter TCPIPService resources. Der URI-Pfad ist ein Qualifikationsmerkmal und kann ausreichend sein, um den Service eindeutig zu identifizieren. Viele REST-konforme Web-Services verwenden jedoch eine zusätzliche Abfragezeichenfolge, um die genaue Ressource zu identifizieren. Betrachten Sie die folgenden Beispiele:

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

Im ersten Beispiel ist der URI-Pfad `JSONServices/AccountService`. Im zweiten Beispiel ist der Pfad `JSONServices` und es gibt die zusätzliche Zeichenfolge `Service=Account`. Beide URI-Stile sind für JSON zulässig. Dies ist ein wichtiger Unterschied zu SOAP. In SOAP wird der erste URI-Stil bevorzugt.

CICS-JSON-Services können mit z/OS Connect Enterprise Edition in REST-konforme Services konvertiert werden. Über eine grafische Benutzerschnittstelle werden URI-Fragmente und HTTP-Header Feldern eines vorhandenen Copybooks zugeordnet und verschiedene Programme können als Ziel für die einzelnen HTTP-Methoden fungieren. Die Fähigkeit, REST-konforme Services aus vorhandenen Anwendungsassets zu erstellen, ist einer der Hauptvorteile von z/OS Connect gegenüber den anderen JSON-Technologien in CICS.

CICS verfügt auch über eine ältere Technologie zum Implementieren einer eingeschränkten Form eines REST-konformen Service. Eine URIMAP-Ressource kann verwendet werden, um die entsprechende WEBSERVICE- und PIPELINE-Instanz

anzugeben, wenn eine eingehende Nachricht verarbeitet wird. Die URIMAP kann einen URI einer bestimmten PIPELINE- oder WEBSERVICE-Instanz zuordnen und dabei möglicherweise die Abfragezeichenfolge des URI in diese Zuordnung einschließen.

CICS verwendet eine URIMAP-Ressource, um die entsprechende zu verwendende WEBSERVICE- und PIPELINE-Instanz zu identifizieren, wenn eine eingehende Nachricht verarbeitet wird. Die URIMAP unterstützt die Verwendung einer Abfragezeichenfolge als Teil des Attributs path. Daher eignet sich die URIMAP für die Verwendung mit beiden Typen von URI.

Verwendung von JSON-Web-Services planen

Bevor Sie die Verwendung von JSON-Web-Services in CICS planen können, müssen Sie die folgenden Fragen für die einzelnen Anwendungen in Betracht ziehen.

Before you begin

Möchten Sie Ihre eigenen Anwendungsprogramme verwenden oder neue schreiben?

Wenn Ihre vorhandenen Anwendungen mit einer wohl definierten Schnittstelle zur Geschäftslogik konzipiert sind, können Sie sie wahrscheinlich in einer Web-Service-Einstellung verwenden, entweder als Service-Provider oder als Service-Requester. In den meisten Fällen werden Sie jedoch ein Wrapperprogramm schreiben müssen, das Ihre Geschäftslogik mit der Web-Service-Logik verbindet.

Wenn Sie planen, neue Anwendungen zu schreiben, sollten Sie Ihre Geschäftslogik von Ihrer Web-Service-Logik getrennt halten, und auch hier müssen Sie ein Wrapperprogramm schreiben, um diese Trennung zu ermöglichen. Wenn Ihre Anwendung jedoch für Web-Services konzipiert wird, ist der Wrapper möglicherweise einfacher zu schreiben.

Möchten Sie den CICS-Assistenten verwenden, um die Zuordnungen zwischen Ihren Datenstrukturen und JSON-Schemas zu generieren?

Der Assistent bietet eine schnelle Implementierung vieler Anwendungen in einer JSON-Web-Services-Einstellung mit wenig oder keinem zusätzlichen Programmieraufwand. Und wenn zusätzliche Programmierung erforderlich ist, ist sie üblicherweise wenig komplex und involviert keine Änderungen an vorhandener Geschäftslogik.

Es gibt jedoch Fälle, in denen besser ohne den JSON-Assistenten gearbeitet wird. Wenn Sie beispielsweise über vorhandenen Code verfügen, der Datenstrukturen JSON-Nachrichten zuordnet, besteht kein Vorteil darin, Ihre Anwendung mit dem JSON-Assistenten umzustrukturieren.

Der CICS-Assistent unterstützt zwar die gängigsten Datentypen und Strukturen, aber eben nicht alle. In dieser Situation sollten Sie die Liste der nicht unterstützten Datentypen und Strukturen für die jeweilige Sprache prüfen und in Betracht ziehen, eine Programmebene bereitzustellen, die Ihre Anwendungsdaten einem Format zuordnet, das der Assistent unterstützen kann. Ist dies nicht möglich, müssen Sie die Nachricht selbst parsen. Detaillierte Angaben dazu, was der Assistent unterstützt oder nicht unterstützt, finden Sie unter High-level language and JSON schema mapping.

JSON-Service-Provider-Anwendung planen

Im Allgemeinen sollten CICS-Anwendungen so strukturiert sein, dass die Trennung von Geschäftslogik und Kommunikationslogik sichergestellt ist. Wenn Sie dies beachten, können Sie neue und vorhandene Anwendungen einfacher in einem Web-Service-Provider implementieren. In manchen Situationen werden Sie ein ein-

zernes Wrapperprogramm zwischen Ihr Anwendungsprogramm und die CICS-Web-Service-Unterstützung schalten müssen.

Abb. 12 zeigt eine typische Anwendung, die partitioniert wurde, um eine Trennung zwischen Kommunikationslogik und Geschäftslogik sicherzustellen.

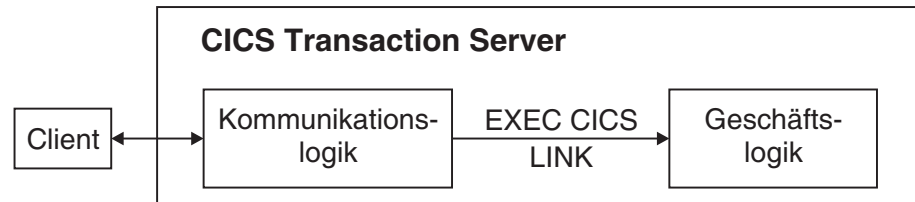


Abbildung 12. In Kommunikations- und Geschäftslogik partitionierte Anwendung

In vielen Fällen können Sie die Geschäftslogik direkt als Service-Provider-Anwendung implementieren. Dies wird in Abb. 13 dargestellt.

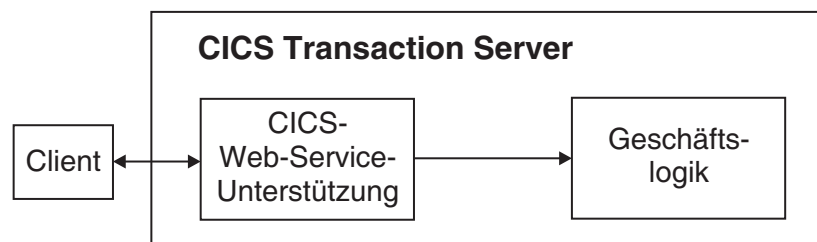


Abbildung 13. Einfache Implementierung einer CICS-Anwendung als Web-Service-Provider

Für die Verwendung dieses einfachen Modells gelten die folgenden Bedingungen:

Wenn z/OS Connect verwendet wird, um die Zuordnung zwischen einem JSON-Schema oder einem Swagger-Dokument und Anwendungsdatenstrukturen zu generieren:

z/OS Connect Enterprise Edition unterstützt eine breitere Auswahl an Anwendungs-Copybook-Strukturen als vom CICS-Assistenten unterstützt werden. In den meisten Fällen werden diese Copybooks unverändert unterstützt. Falls nicht, können Sie ein Wrapperprogramm zwischen den z/OS Connect-Service und Ihre Geschäftslogik schalten.

Wenn der CICS-Assistent verwendet wird, um die Zuordnung zwischen dem JSON-Schema und Anwendungsdatenstrukturen zu generieren:

Die in der Schnittstelle zur Geschäftslogik verwendeten Datentypen müssen von dem CICS-Assistenten unterstützt werden. Ist dies nicht der Fall, müssen Sie ein Wrapperprogramm zwischen die CICS-Web-Service-Unterstützung und Ihre Geschäftslogik schalten.

Sie werden auch ein Wrapperprogramm brauchen, wenn Sie ein vorhandenes Programm implementieren, um einen Service zur Verfügung zu stellen, der einer vorhandenen Web-Service-Beschreibung entspricht: Wenn Sie die Web-Service-Beschreibung mithilfe des Assistenten verarbeiten, stimmen die resultierenden Datenstrukturen wahrscheinlich nicht mit der Schnittstelle Ihrer Geschäftslogik überein.

Wenn der CICS-Assistent nicht verwendet wird:

Nachrichtenhändler in Ihrer Service-Provider-Pipeline müssen direkt mit Ihrer Geschäftslogik interagieren.

Wrapperprogramm verwenden

Verwenden Sie ein Wrapperprogramm, wenn der CICS-Assistent keinen Code generieren kann, um direkt mit der Geschäftslogik zu interagieren. Beispielsweise verwendet die Schnittstelle zur Geschäftslogik möglicherweise eine Datenstruktur, die der CICS-Assistent nicht direkt einer JSON-Nachricht zuordnen kann. In dieser Situation können Sie ein Wrapperprogramm verwenden, um alle erforderlichen zusätzlichen Datenbearbeitungsmöglichkeiten zur Verfügung zu stellen:

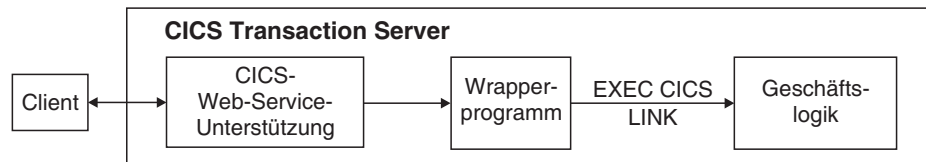


Abbildung 14. Implementierung einer CICS-Anwendung als Web-Service-Provider mithilfe eines Wrapperprogramms

Sie werden eine zweite Datenstruktur entwerfen müssen, die der Assistent unterstützen kann, und diese als Schnittstelle zu Ihrem Wrapperprogramm verwenden. Das Wrapperprogramm muss dann zwei einfache Funktionen erfüllen:

- Daten zwischen den beiden Datenstrukturen verschieben
- Geschäftslogik über ihre vorhandene Schnittstelle aufrufen

Fehlerbehandlung

Wenn Sie den CICS-Assistenten verwenden möchten, sollten Sie auch berücksichtigen, wie Änderungen rückgängig gemacht werden können, wenn Fehler auftreten. Wenn eine JSON-Anforderungsnachricht von einem Service-Requester eingeht, wird die JSON-Nachricht von CICS umgesetzt, bevor sie an Ihr Anwendungsprogramm übergeben wird. Wenn während dieser Umsetzung ein Fehler auftritt, werden nicht automatisch alle Änderungen, die an der Nachricht vorgenommen wurden, von CICS rückgängig gemacht. Wenn Sie die JSON-Nachricht beispielsweise mithilfe von Handlern in der Pipeline weiter bearbeiten möchten, müssen Sie entscheiden, ob diese alle zurücknehmbaren Änderungen, die bereits vorgenommen wurden, wieder rückgängig machen sollen.

Wenn Ihre Service-Provider-Anwendung beispielsweise eine Antwortnachricht an einen Service-Requester sendet, werden in ausgehenden JSON-Nachrichten alle zurücknehmbaren Änderungen, die vom Anwendungsprogramm vorgenommen wurden, automatisch zurückgesetzt, falls in CICS bei Generieren der JSON-Antwortnachricht ein Fehler auftritt. Sie sollten überlegen, ob das Hinzufügen von Synchronisationspunkten für Ihr Anwendungsprogramm geeignet ist.

JSON-Service-Requester-Anwendung planen

CICS bietet keine integrierte Unterstützung für JSON-Web-Services im Requestermodus. Diese Services werden von z/OS Connect Enterprise Edition V3.0 bereitgestellt.

Weitere Informationen zu z/OS Connect finden Sie unter „CICS und z/OS Connect“. Um einen fernen JSON-Web-Service ohne z/OS Connect aus einer CICS-Anwendung aufzurufen, verwenden Sie die **EXEC CICS WEB-API**-Befehle. Weitere Informationen finden Sie unter Creating a JSON web service client application.

CICS und z/OS Connect

z/OS Connect macht es möglich, CICS-Programme über eine JSON-Schnittstelle (JavaScript Object Notation) aufzurufen.

z/OS Connect for CICS 1.0 wurde zunächst als Alternative für die JSON-Funktionalitäten der Java Pipelines for JSON-Pipelines eingeführt, die im CICS TS-Feature-Pack für mobile Erweiterungen bereitgestellt und in CICS TS Version 5.2 integriert wurden. Seitdem hat sich z/OS Connect zu einem eigenständigen Produkt namens z/OS Connect Enterprise Edition entwickelt, das über zusätzliche Funktionalitäten verfügt. In den folgenden Informationen werden die Unterschiede zwischen den alternativen Angeboten und den Schritten zur Implementierung beschrieben.

z/OS Connect ist die primäre Technologie von IBM für die Implementierung von JSON-Services und APIs in CICS. Es ist in drei Versionen erhältlich und unterstützt verschiedenen Implementierungsoptionen. z/OS Connect for CICS 1.0 ist die Einstiegsedition und eine kostengünstige Option, verfügt aber nicht über die Erweiterungen der anderen Versionen. z/OS Connect Enterprise Edition Version 2.0 und 3.0 bietet eine breitere Auswahl an Integrationsoptionen.

Eine Übersicht über JSON-Services finden Sie unter CICS as a service provider for JSON requests.

Wartungsüberlegungen für z/OS Connect Enterprise Edition

Da z/OS Connect Enterprise Edition eigenständig oder in Liberty eingebettet in CICS ausgeführt werden kann, müssen die beiden Umgebungen eindeutig beschrieben werden. Sie können beide Umgebungen gleichzeitig ausführen, aber sie sind nicht gegenseitig ausschließend, deshalb müssen Sie Vorsicht walten lassen, wenn Sie die WebSphere Liberty Profile-Clientbibliotheken (WLP) warten, die die BBOA*-Module enthalten.

Weitere Informationen finden Sie unter Keeping CICS TS 5.3 and z/OS Connect EE 2.0 maintenance in sync.

z/OS Connect Enterprise Edition

z/OS Connect Enterprise Edition ist ein separat erhältliches IBM Produkt. Es baut auf den Funktionalitäten von z/OS Connect for CICS 1.0 auf, was auch Unterstützung für JSON-Services einschließt. Mit z/OS Connect Enterprise Edition können API-Entwickler JSON-APIs aus JSON-Services erstellen. Die APIs werden mit dem Eclipse-basierten API-Editor erstellt und gepackt, der mit z/OS Connect Enterprise Edition bereitgestellt wird, und anschließend in der z/OS Connect-Laufzeit implementiert. Das API-Paket enthält Swagger 2.0-Definitionen, um für Entwickler das Einbinden der APIs in ihre Anwendungen zu vereinfachen. Zentrale z/OS Connect-Funktionen wie die Berechtigungssicherheitsprüfung für den Serviceaufruf, das Erstellen von SMF-Datensätzen (System Management Facility, Systemverwaltungsfunktion) und das Protokollieren von REST-konformen Serviceanforderungen gelten auch für die APIs.

z/OS Connect Enterprise Edition kann so für die Ausführung auf einem Liberty-JVM-Server in CICS konfiguriert werden. Ein z/OS Connect Enterprise Edition-Server, der auf einem Liberty-JVM-Server in CICS ausgeführt wird, verwendet einen Service-Provider für die Verbindung mit den CICS-Anwendungsprogrammen. Bei z/OS Connect Enterprise Edition V3.0 können Sie auswählen, welcher CICS-Service-Provider für Ihre Konfiguration verwendet werden soll.

- Wenn Sie mit z/OS Connect Enterprise Edition V2.0 arbeiten, verwenden Sie den CICS-Service-Provider, der mit CICS TS bereitgestellt wird, und das Feature `cicsts:zosConnect-2.0`. Diese Konfiguration ermöglicht eine lokale leistungsfähige Konnektivität mit CICS-Programmen. Dies ist der einzige unterstützte Service-Provider für z/OS Connect Enterprise Edition V2.0.
- Wenn Sie z/OS Connect Enterprise Edition V3.0 für die Verbindung mit einer lokalen CICS-Region nutzen, verwenden Sie den CICS-Service-Provider, der mit z/OS Connect Enterprise Edition V3.0 bereitgestellt wird, und das Feature `zosconnect:cicsService-1.0`. Diese Konfiguration verwendet eine lokale optimierte Verbindung zur lokalen CICS-Region.
- Wenn Sie z/OS Connect Enterprise Edition V3.0 für die Verbindung mit einer fernen CICS-Region verwenden, nutzen Sie den CICS-Service-Provider, der mit z/OS Connect Enterprise Edition V3.0 bereitgestellt wird, und das Feature `zosconnect:cicsService-1.0`. Diese Konfiguration verwendet eine IP-Interkonnektivitätsverbindung mit einer fernen CICS-Region.

JSON-Services, die für die Verwendung mit z/OS Connect for CICS 1.0 entwickelt wurden, ebenso wie die meisten JSON-Services, die für die Verwendung mit Java Pipelines for JSON entwickelt wurden, können in z/OS Connect Enterprise Edition gehostet werden. Weitere Details finden Sie unter „z/OS Connect for CICS 1.0“. Benutzer des API-Editors, der im Lieferumfang von z/OS Connect Enterprise Edition enthalten ist, sollten jedoch wissen, dass es gewisse Einschränkungen gibt. Details finden Sie unter Using APIs from z/OS Connect Enterprise Edition.

Weitere Informationen finden Sie unter z/OS Connect Enterprise Edition V3.0 product documentation.

z/OS Connect for CICS 1.0

z/OS Connect for CICS 1.0 ist ein gebührenfreies Feature, das ab Version 5.2 in CICS TS enthalten ist. Es entspricht weitestgehend den Java Pipelines for JSON und die meisten JSON-Web-Services können zwischen Umgebungen erneut implementiert werden, ohne dass Änderungen an Anwendungen oder WSBIND-Dateien vorgenommen werden müssen. Allerdings können sich der URI und die Sicherheitskonfiguration in den einzelnen Umgebungen unterscheiden. JSON-Web-Services, die in z/OS Connect for CICS 1.0 implementiert werden, können auch in z/OS Connect Enterprise Edition implementiert werden.

Die Version von z/OS Connect, die in CICS TS eingebettet ist, nutzt einen Großteil der Funktionalität von z/OS Connect in WebSphere Liberty for z/OS gemeinsam, ist aber für den lokalen Zugriff auf CICS optimiert und verwendet CICS-Standard-services. Wenn Sie sich mit z/OS Connect in WebSphere Liberty for z/OS auskennen, sind die Hauptunterschiede zwischen dieser Anwendung und z/OS Connect for CICS 1.0 die folgenden:

- z/OS Connect for CICS 1.0 unterstützt nur die Implementierung von JSON-Web-Services für CICS. Es kann nicht für andere z/OS-Subsysteme wie IMS oder Batch verwendet werden.

- z/OS Connect for CICS 1.0 kann in CICS-Ressourcen integriert werden. JSON-Web-Services werden mithilfe von WEBSERVICE- und URIMAP-Ressourcen implementiert.
- Verfahren zur JSON-Serviceaktivierung über die Java Pipelines for JSON werden in z/OS Connect for CICS 1.0 unterstützt, einschließlich solcher, die normalerweise nicht in z/OS Connect verfügbar sind.
- JSON-Services können mithilfe von Bottom-up- und Top-down-Entwicklungsstrategien entwickelt werden.
- Für Java Pipelines for JSON generierte JSON-Web-Services können in z/OS Connect for CICS 1.0 erneut implementiert werden.

Weitere Informationen zu z/OS Connect in WebSphere Liberty for z/OS finden Sie unter [IBM z/OS Connect overview](#).

Frühere Technologien

Andere JSON-Technologien für CICS sind älter als z/OS Connect, darunter das „Feature-Pack für mobile Erweiterungen“ und die „JSON-Services“. Das Feature-Pack für mobile Erweiterungen V1.0 fand Anwendung in CICS TS for z/OS V4.2 und V5.1. Mit dem Feature-Pack konnten CICS-Anwendungen als REST-konforme Web-Services mit JSON-Nutzdaten verfügbar gemacht, vorhandene JSON-Anwendungen aufgerufen und JSON in und aus Anwendungsdaten konvertiert werden. In CICS TS V5.2 und höher wurde diese Funktionalität in CICS TS integriert. Services, die für die älteren Technologien implementiert werden, sind in der Regel kompatibel mit z/OS Connect. z/OS Connect Enterprise Edition liefert jedoch die beste Erfahrung für neue Services und APIs.

Implementierungsoptionen für z/OS Connect Enterprise Edition

Es gibt drei zentrale Implementierungsoptionen für z/OS Connect Enterprise Edition:

Eigenständiges z/OS Connect Enterprise Edition

Konnektivität mit CICS wird entweder mithilfe von IPIC oder mithilfe von WOLA-Verbindungen aus einem externen Adressraum erreicht. Weitere Informationen finden Sie unter [Configuring in z/OS Connect Enterprise Edition V3.0 product documentation](#).

z/OS Connect Enterprise Edition V3 auf einem CICS-JVM-Server (JVMSERVER), der von z/OS Connect verwaltete Services und APIs hostet

z/OS Connect Enterprise Edition V3.0 wird auf einem Liberty-JVM-Server in CICS gehostet und die Kommunikation wird über eine lokal optimierte IPIC-Verbindung hergestellt. Services werden mithilfe von funktionsreichen Benutzerschnittstellentools erstellt und mithilfe von SAR- und AAR-Dateien implementiert. Diese Konfigurationsoption ähnelt der Option 'Eigenständiges z/OS Connect Enterprise Edition', mit der Ausnahme, dass sie im CICS-Adressraum gehostet wird.

z/OS Connect Enterprise Edition auf einem CICS-JVM-Server (JVMSERVER), der von CICS verwaltete Services hostet (Kompatibilitätsmodus)

Diese Option ist für alle Versionen von z/OS Connect verfügbar. Sie bietet einen Hybridmodus, in dem Services, die zuvor für CICS-JSON-Services-Technologie implementiert wurden, in z/OS Connect zusammengefasst werden. WSBIND-Dateien werden unter Verwendung der JCL-Prozeduren DFHLS2JS oder DFHJS2LS erstellt, die in CICS als WEBSERVICE-Ressourcen implementiert, jedoch von CICS in z/OS Connect installiert werden. Diese Konfigurationsoption bietet viele der Vorteile von z/OS Connect

Enterprise Edition für den älteren Typ von JSON-Services, bringt aber auch einige Einschränkungen und Unterschiede in der Funktionalität mit sich.

z/OS Connect-Funktionen vergleichen

In diesem Thema werden die Funktionen von z/OS Connect Enterprise Edition und z/OS Connect for CICS 1.0 verglichen. Ein Vergleich für die Java Pipelines for JSON (oder das Feature-Pack für mobile Erweiterungen) ist ebenfalls enthalten, weil dies z/OS Connect for CICS 1.0 vorausging.

Tabelle 1. Funktionen von z/OS Connect Enterprise Edition, z/OS Connect for CICS 1.0 und dem Feature-Pack für mobile Erweiterungen vergleichen

Funktion	z/OS Connect Enterprise Edition	z/OS Connect for CICS 1.0	Java Pipelines for JSON (oder das Feature-Pack für mobile Erweiterungen)
Unterstützung für z/OS Connect-Standardfunktionen	Ja. Für die Serviceerkennung sind zusätzliche Konfigurationsschritte erforderlich.	Ja. Für die Serviceerkennung sind zusätzliche Konfigurationsschritte erforderlich.	Nein
Aktivierung von CICS-Programmen als JSON-Web-Services	Ja	Ja	Ja
Unterstützung für REST-konforme JSON-Services	Ja	Ja	Ja
Unterstützung für REST-konforme APIs	Ja	Nein	Nein
Unterstützung für das Starten von fernem JSON-Services	V3: ja; V2: nein, DFHJSON verwenden	Nein, DFHJSON verwenden	Nein (DFHJSON verwenden)
Unterstützung für CICS-Datenumsetzung mithilfe der Assistenten DFHLS2JS und DFHJS2LS	Ja. Siehe Hinweis 1.	Ja	Ja
Unterstützung für SAR-basierte Serviceimplementierung	V3: optional; V2: nein	Nein	Nein
Unterstützung für JSON-Schemaspezifikation	Ja (draft-04)	Ja (draft-04)	Ja (draft-04)
Auswahl von Parsertechnologien für JSON-Umsetzung	Ja	Ja	Nein
Unterstützung für PIPELINE-Handlerprogramme	Nein	Nein	Ja
Unterstützung für Interceptorprogramme	Ja, mit z/OS Connect-Interceptors	Ja, mit z/OS Connect-Interceptors	Ja, mit Axis2-Handlern

Tabelle 1. Funktionen von z/OS Connect Enterprise Edition, z/OS Connect for CICS 1.0 und dem Feature-Pack für mobile Erweiterungen vergleichen (Forts.)

Funktion	z/OS Connect Enterprise Edition	z/OS Connect for CICS 1.0	Java Pipelines for JSON (oder das Feature-Pack für mobile Erweiterungen)
Anwendungszugriff auf CICS-Steuercontainer	Ja	Ja	Ja
WSBind-Implementierung über eine CICS-PIPELINE-Resource	V3: optional; V2: ja. Siehe Hinweis 2.	Ja	Ja
Konfiguration mit URIMAP- und WEBSERVICE-Resources	V3: optional; V2: ja. Siehe Hinweis 3.	Ja	Ja
Netzkonfiguration	Mit WebSphere Liberty	Mit WebSphere Liberty	Mit einer TCPIPService-Resource
Sicherheitskonfiguration	Mit WebSphere Liberty und CICS-Sicherheit	Mit WebSphere Liberty und CICS-Sicherheit	Mit CICS
Unterstützung für Transport Layer Security (TLS)	Ja	Ja	Ja
Statistik, Überwachung, Benutzerexits und weitere Diagnoseinfrastruktur	Konsistent mit WebSphere Liberty in CICS	Konsistent mit WebSphere Liberty in CICS	Konsistent mit Pipelines in CICS
JVM-Umgebung	Ein Liberty-JVM-Server, der vorzugsweise für die alleinige Verwendung durch z/OS Connect isoliert ist.	Ein Liberty-JVM-Server, der vorzugsweise für die alleinige Verwendung durch z/OS Connect isoliert ist.	Ein Nicht-OSGi-JVM-Server, der mit JAVA_PIPELINE=YES konfiguriert wurde und vorzugsweise für die alleinige Verwendung durch die Java Pipelines for JSON isoliert ist.

Anmerkung:

1. Wenn Sie den CICS-Service-Provider verwenden, der mit CICS TS bereitgestellt wird, wird diese Funktion für Services unterstützt. Verwenden Sie für APIs die Assistenten, die mit z/OS Connect Enterprise Edition bereitgestellt werden.
Wenn Sie den CICS-Service-Provider verwenden, der mit z/OS Connect Enterprise Edition V3.0 bereitgestellt wird, wird diese Funktion nicht unterstützt. Verwenden Sie das z/OS Connect Enterprise Edition-API-Toolkit oder das Build-Toolkit, um Services zu erstellen, und anschließend das API-Toolkit, um APIs zu erstellen.
2. Wenn Sie den CICS-Service-Provider verwenden, der mit CICS TS bereitgestellt wird, wird diese Funktion für Services unterstützt, aber die Implementierung von APIs umfasst zusätzliche Artefakte. Weitere Informationen finden Sie unter z/OS Connect Enterprise Edition V3.0 product documentation.

Wenn Sie den CICS-Service-Provider verwenden, der mit z/OS Connect Enterprise Edition V3.0 bereitgestellt wird, wird diese Funktion für Services nicht unterstützt. Verwenden Sie das z/OS Connect Enterprise Edition-Serviceverzeichnis. Die Implementierung von APIs umfasst zusätzliche Artefakte. Weitere Informationen finden Sie unter z/OS Connect Enterprise Edition V3.0 product documentation.

3. Diese Funktion wird nur dann für Services unterstützt, wenn Sie den CICS-Service-Provider verwenden, der mit CICS TS bereitgestellt wird. Der CICS-Service-Provider, der mit z/OS Connect Enterprise Edition V3.0 bereitgestellt wird, unterstützt die manuelle Erstellung von URIMAPs. Mit einer URIMAP können Sie beispielsweise die CICS-Standardtransaktion (CJSA) überschreiben.


Funktionalität von z/OS Connect for CICS


Die integrierte z/OS Connect-Funktionalität in CICS unterscheidet sich von z/OS Connect in anderen Umgebungen. z/OS Connect for CICS ist für den lokalen Zugriff auf CICS optimiert und verwendet CICS-Standardservices.

z/OS Connect for CICS hat dieselben Features und Funktionen wie z/OS Connect in WebSphere Liberty for z/OS. Aber es gibt auch Unterschiede.

- WSBIND-Dateien können mit den von CICS bereitgestellten Tools DFHLS2JS und DFHJS2LS generiert werden.
- WSBIND-Dateien werden in der z/OS Connect for CICS-Infrastruktur unter Verwendung der **WEBSERVICE**- und **URIMAP**-Ressourcen implementiert. Dies führt zu einem ähnlichen operativen Management von z/OS Connect for CICS und anderen CICS-Web-Services.
- JSON-Services, die in CICS gehostet werden, können mit CICS-Programmen über eine optimierte lokale Verbindung interagieren.
- JSON-Services können mithilfe von Bottom-up- und Top-down-Entwicklungsstrategien entwickelt werden.
- Das REST-konforme Design von JSON-Services kann in CICS integriert werden.
- Mit dem integrierten z/OS Connect for CICS kann nicht direkt mit Ressourcen von anderen z/OS-Subsystemen wie IMS oder Batch interagiert werden.
- Wenn der PIPELINE-Scanbefehl zur Installation von Services verwendet wird, sind keine Änderungen an den einzelnen Servicekonfigurationen für CICS in der Datei server.xml des WebSphere-Liberty-Profiles erforderlich. Aber eine Konfiguration der einzelnen Servicekonfigurationen in server.xml wird unterstützt.
- Sowohl z/OS Connect als auch z/OS Connect for CICS fordern den Stammkontext für URIs an. Es wird empfohlen, z/OS Connect for CICS in einer eigenen Liberty-JVM-Server-Umgebung zu implementieren.

Related information:

 [Configuring z/OS Connect for CICS](#)

 [Security for z/OS Connect for CICS](#)

Kapitel 2. Web-Services in CICS konfigurieren

Sie können CICS für die Unterstützung von Web-Services konfigurieren, wobei CICS-Anwendungen zu Web-Service-Requestern oder Web-Service-Providern werden können. CICS unterstützt verschiedene Web-Service-Spezifikationen, einschließlich binärer Anhänge und Web-Service-Adressierung. Sie können CICS auch so konfigurieren, dass Web-Service-Anforderungen von WebSphere MQ oder HTTP akzeptiert werden, und dass WSDL-Dateien aus WSRR abgerufen werden.

CICS-System für Web-Services konfigurieren

Bevor Sie Web-Services verwenden können, muss Ihr CICS-System ordnungsgemäß konfiguriert werden.

Procedure

1. Stellen Sie sicher, dass Sie die Language Environment-Unterstützung für PL/I installiert haben. Weitere Informationen finden Sie unter Language Environment-Unterstützung installieren.
2. Aktivieren Sie z/OS-Unterstützung für Unicode. Sie müssen die z/OS-Konvertierungsservices aktivieren und ein Konvertierungsbild installieren, das die Datenkonvertierungen angibt, die CICS zwischen SOAP-Nachrichten und einem Anwendungsprogramm durchführen soll. Weitere Informationen finden Sie unter z/OS Unicode Services User's Guide and Reference.

CICS-Ressourcen für Web-Services

PIPELINE-, WEBSERVICE-, URIMAP- und TCPIPSERVICE-Ressourcen unterstützen Web-Service in CICS.

PIPELINE

Für jeden Web-Service ist eine PIPELINE-Ressourcendefinition erforderlich. Sie stellt Informationen zu den Nachrichtenhandlerprogrammen bereit, die eine Serviceanforderung und die Antwort verarbeiten. In der Regel definiert eine einzelne PIPELINE-Ressourcendefinition eine Infrastruktur, die von vielen Anwendungen verwendet werden kann. Die Informationen über die Nachrichtenhandler werden indirekt bereitgestellt: Die PIPELINE-Ressourcendefinition gibt den Namen einer z/OS UNIX-Datei an, die eine XML-Beschreibung der Handler und ihre Konfiguration enthält.

Eine PIPELINE-Ressource, die für einen Service-Requester erstellt wird, kann nicht für einen Service-Provider verwendet werden, und umgekehrt. Die zwei Typen von PIPELINE-Definitionen werden anhand der Inhalte der Pipelinekonfigurationsdatei unterschieden, die im CONFIGFILE-Attribut angegeben ist: Für einen Service-Provider ist das übergeordnete Element <provider_pipeline>, für einen Service-Requester ist es <requester_pipeline>.

WEBSERVICE

Eine WEBSERVICE-Ressourcendefinition ist nur erforderlich, wenn die Zuordnung zwischen der Anwendungsdatenstruktur und SOAP-Nachrichten mithilfe des CICS-Web-Service-Assistenten generiert wurde. Sie definiert Aspekte der Laufzeitumgebung für ein CICS-Anwendungsprogramm, das in einer Web-Service-Einstellung implementiert ist.

Obwohl CICS die gängigen Ressourcendefinitionsmechanismen für WEBSERVICE-Ressourcen bereitstellen, werden sie in der Regel automatisch aus einer Web-Service-Bindungsdatei generiert, wenn das Abholverzeichnis für die PIPELINE-Ressourcendefinition gescannt wird. Dies kann der Fall sein, wenn die PIPELINE-Ressource installiert wird, oder als Ergebnis eines Befehls `PERFORM PIPELINE SCAN`. Die in diesem Fall auf die WEBSERVICE-Ressource angewendeten Attribute stammen aus einer Web-Service-Bindungsdatei, die von dem Web-Service-Assistenten erstellt wird. Die Informationen in der Bindungsdatei stammen aus der Web-Service-Beschreibung oder werden als Parameter des Web-Service-Assistenten bereitgestellt.

Eine WEBSERVICE-Ressource, die für einen Service-Requester erstellt wird, kann nicht für einen Service-Provider verwendet werden, und umgekehrt. Die zwei Typen von WEBSERVICE-Ressourcen werden anhand des `PROGRAM`-Attributs in der Ressourcendefinition unterschieden: für einen Service-Provider muss das Attribut angegeben sein, für einen Service-Requester darf es nicht angegeben sein.

URIMAP

Eine URIMAP-Definition ist in einem Service-Provider erforderlich, wenn sie Informationen enthält, die den URI einer eingehenden Web-Service-Anforderung anderen Ressourcen (wie der PIPELINE-Ressource) zuordnen, die die Anforderung verarbeiten wird. Diese URIMAP-Definition ist auch erforderlich, wenn Sie HTTP-Basisauthentifizierung verwenden, weil die URIMAP-Ressourcendefinition angibt, dass die Benutzer-ID-Informationen des Service-Requesters in einem HTTP-Autorisierungsheader an den Service-Provider übergeben werden.

Eine zweite optionale URIMAP-Definition kann in einem Service-Provider für WSDL-Erkennung vorhanden sein. Diese URIMAP-Ressourcendefinition enthält Informationen, die den URI einer eingehenden Anforderung für das WSDL-Dokument oder die WSDL-Dokumente zuordnen, das bzw. die dem Web-Service zugeordnet sind.

Für Service-Provider, die mit dem CICS-Web-Service-Assistenten implementiert werden, werden die URIMAP-Ressourcen in der Regel automatisch erstellt, wenn das Abholverzeichnis gescannt wird, obwohl CICS die üblichen Ressourcendefinitionsmechanismen bereitstellt. Dieser Scan wird bei der Installation der PIPELINE-Ressource oder als Ergebnis des Befehls `PERFORM PIPELINE SCAN` ausgeführt. Die URIMAP-Ressource, die CICS die Informationen für die Zuordnung der WEBSERVICE-Ressource zu einem bestimmten URI bereitstellt, ist eine erforderliche Ressource. Die Attribute für diese Ressource werden von einer Web-Service-Bindungsdatei im Abholverzeichnis angegeben. Die URIMAP-Ressource, die CICS die Informationen für die Zuordnung der WSDL-Archivdatei oder des WSDL-Dokuments zu einem bestimmten URI bereitstellt, ist eine optionale Ressource und wird erstellt, wenn entweder eine WSDL-datei oder eine WSDL-Archivdatei im Abholverzeichnis vorhanden sind. Weitere Informationen zur Erstellung von URIMAP-Ressourcen für Web-Service-Provider finden Sie unter `Creating a web service provider by using the web services assistant`.

Für Service-Requester erstellt CICS keine URIMAP-Ressourcen automatisch, wenn die PIPELINE-Ressource installiert wird, oder als Ergebnis des Befehls `PERFORM PIPELINE SCAN`. Service-Requester müssen keine URIMAP-Ressourcen verwenden, wenn sie Anforderungen stellen. Sie können den URI der ausgehenden Anforderung direkt im Anwendungsprogramm angeben. Wenn Sie jedoch eine URIMAP-Ressource für die Clientanforde-

rung erstellen und Ihre Service-Requester die URIMAP-Ressource verwenden, um den URI bereitzustellen, können Sie die folgenden Vorteile nutzen:

- Systemadministratoren können alle Änderungen am Endpunkt der Verbindung verwalten, sodass Sie Ihre Anwendungen nicht erneut kompilieren müssen, wenn sich der URI eines Service-Providers ändert.
- Sie können auswählen, dass in CICS die Verbindungen, die mit der URIMAP-Ressource geöffnet wurden, nach der Verwendung geöffnet bleiben und in einen Pool gestellt werden, zur Wiederverwendung durch die Anwendung für nachfolgende Anforderungen oder durch eine andere Anwendung, die denselben Service aufruft. Das Verbindungspooling ist nur verfügbar, wenn Sie eine URIMAP-Ressource angeben, für die das Attribut SOCKETCLOSE festgelegt ist. Weitere Informationen zu den Leistungsvorteilen des Verbindungspoolings finden Sie unter Connection pooling for HTTP client performance.

Das Konfigurieren von URIMAP-Ressourcenattributen auf spezifische Weise kann aktivieren, dass eingehende Anforderungen von direkt angehängten Benutzertransaktionen verarbeitet werden und dass die über das Web angehängte Task übergangen wird. Weitere Informationen finden Sie unter HTTP requests are processed by directly attached user transactions.

TCPIPSERVICE

Eine TCPIPSERVICE-Definition ist in einem Service-Provider erforderlich, der den HTTP-Transport verwendet. Sie enthält Informationen zu dem Port, an dem eingehende Anforderungen empfangen werden.

Welche Ressourcen zur Unterstützung eines bestimmten Anwendungsprogramms erforderlich sind, hängt von den folgenden Kriterien ab:

- Ob das Anwendungsprogramm ein Service-Provider oder ein Service-Requester ist.
- Ob die Anwendung mit dem CICS-Web-Service-Assistenten implementiert wird.

Service-Requester oder -Provider	CICS-Web-Service-Assistent wird verwendet	PIPELINE ist erforderlich	WEBSERVICE ist erforderlich	URIMAP ist erforderlich	TCPIPSERVICE ist erforderlich
Provider	Ja	Ja	Ja (aber siehe Anmerkung 1)	Ja (aber siehe Anmerkung 1)	Siehe Anmerkung 2
Provider	Nein	Ja	Nein	Ja	Siehe Anmerkung 2
Requester	Ja	Ja	Ja	Siehe Anmerkung 3	Nein
Requester	Nein	Ja	Nein	3	Nein

Anmerkungen:

1. Wenn der CICS-Web-Service-Assistent verwendet wird, um ein Anwendungsprogramm zu implementieren, können die WEBSERVICE- und zwei URIMAP-Ressourcen automatisch erstellt werden, wenn das Abholverzeichnis der PIPELINE gescannt wird. Die erste URIMAP-Ressource ist erforderlich und stellt CICS die Informationen für die Zuordnung der WEBSERVICE-Ressource zu einem bestimmten URI bereit. Die zweite URIMAP-Ressource ist optional und stellt CICS die Informationen für die Zuordnung der WSDL-Archivdatei zu einem bestimmten URI bereit, sodass die externen Requester mit dem URI die WSDL-Archivdatei oder das WSDL-Dokument erkennen können. Das Abhol-

verzeichnis dieses PIPELINE-Scans wird bei der Installation der PIPELINE-Ressource oder als Ergebnis des Befehls PERFORM PIPELINE SCAN ausgeführt.

2. Eine TCPIPService-Ressource ist erforderlich, wenn der HTTP-Transport verwendet wird. Wird der WebSphere MQ-Transport verwendet, ist keine TCPIPService-Ressource erforderlich.
3. Eine URIMAP-Ressource ist für einen Service-Requester optional, und der CICS-Web-Service-Assistent generiert keine automatisch. Wenn Sie Ihre eigenen URIMAP-Ressourcen für zu verwendende Service-Requester definieren, können Sie Verbindungspooling implementieren und Änderungen an den URIs für Service-Provider vornehmen.

Das Konfigurieren von TCPIP-Ressourcenattributen auf spezifische Weise kann aktivieren, dass eingehende Anforderungen von direkt angehängten Benutzertransaktionen verarbeitet werden und dass die über das Web angehängte Task übergangen wird. Weitere Informationen finden Sie unter HTTP requests are processed by directly attached user transactions.

Wenn Sie viele Web-Service-Anwendungen in einem CICS-System implementieren, gibt es in der Regel von jedem Typ von Ressource mehrere Vorkommen. In diesem Fall können Sie manche Ressourcen in mehreren Anwendungen verwenden. Jede Web-Service-Datei oder -Ressource wird einer oder mehreren CICS-Ressourcen eines anderen Typs zugeordnet.

Tabelle 2. Andere CICS-Ressourcen, die den einzelnen Web-Service-Dateien und -Ressourcen zugeordnet sind

Web-Service-Datei oder -Ressource	Zugeordnete Ressourcen
Pipelinekonfigurationsdatei	<ul style="list-style-type: none"> Mehr als eine PIPELINE-Ressource, die sich auf die Datei bezieht.
PIPELINE	<ul style="list-style-type: none"> Mehr als eine PIPELINE-Ressource, die sich auf die PIPELINE-Ressource bezieht. Mehr als eine WEBSERVICE-Ressource, die sich auf die PIPELINE-Ressource bezieht. Mehr als eine Web-Service-Bindungsdatei im Abholverzeichnis der PIPELINE-Ressource.
Web-Service-Bindungsdatei	<ul style="list-style-type: none"> Eine URIMAP-Ressource, die automatisch aus der Bindungsdatei generiert wird. Sie können weitere URIMAP-Ressourcen für einen Service-Provider definieren und Sie können URIMAP-Ressourcen für einen Service-Requester definieren. Eine WEBSERVICE -Ressource, die automatisch aus der Bindungsdatei generiert wird. Sie können bei Bedarf weitere WEBSERVICE-Ressourcen definieren.

Tabelle 2. Andere CICS-Ressourcen, die den einzelnen Web-Service-Dateien und -Ressourcen zugeordnet sind (Forts.)

Web-Service-Datei oder -Ressource	Zugeordnete Ressourcen
WEBSERVICE	<ul style="list-style-type: none"> Mehr als eine URIMAP-Ressource. Wenn die WEBSERVICE-Ressource automatisch aus der Bindungsdatei für einen Service-Provider generiert wird, generiert CICS eine entsprechende URIMAP-Ressource. Sie können weitere URIMAP-Ressourcen für einen Service-Provider definieren und Sie können URIMAP-Ressourcen für einen Service-Requester definieren.
URIMAP	<ul style="list-style-type: none"> Nur eine TCPIPService-Ressource, wenn sie explizit in der URIMAP-Ressource genannt ist.
TCPIPService	<ul style="list-style-type: none"> Viele URIMAP-Ressourcen.

Web-Service-Erkennung

WSDL-Dokumente, die einem Web-Service im Providermodus zugeordnet sind, werden automatisch im Web veröffentlicht.

Es gibt eine Konvention in Web-Service-Hosting-Umgebungen, die es ermöglicht, die WSDL für einen Web-Service von einem fernen Client abfragen zu lassen (in der Regel ist dies ein Anwendungsentwickler, der einen Web-Browser verwendet). Dieser ferne Client verwendet den URI für den Web-Service mit dem Suffix `?wsdl`. Diese Konvention kann das Verteilen der WSDL an interessierte Parteien vereinfachen, ohne dass ein formales WSDL-Repository erforderlich ist. Diese Konvention ist in CICS implementiert.

Beispiel: Sie haben einen Web-Service in CICS gehostet und unter dem folgenden URI veröffentlicht:

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

Das zugeordnete WSDL-Dokument könnte wiederhergestellt werden, indem der folgende URI unter Verwendung eines Web-Browsers angefordert wird:

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

WSDL-Dokumente für Service-Provider können für die Erkennung mithilfe von URIMAP-Ressourcen veröffentlicht werden. Wenn Sie die einzelnen PIPELINE-Ressourcen installieren, scannt CICS das im WSDIR-Attribut der PIPELINE-Ressource angegebene Verzeichnis (das Abholverzeichnis). Wenn dieses Verzeichnis entweder eine WSDL-Archivdatei oder ein WSDL-Dokument enthält, wird eine zweite URIMAP-Ressource installiert. Diese neue URIMAP-Ressource stellt CICS die Informationen für die Zuordnung der WSDL-Archivdatei oder des WSDL-Dokuments zu einem bestimmten URI bereit, sodass die externen Requester mit dem URI die WSDL-Archivdatei oder das WSDL-Dokument erkennen können. Dieser URI hat denselben Pfad wie der URI, der dem WEBSERVICE-Element zugeordnet ist, wobei das Suffix `?wsdl` angehängt ist.

Die WSDL-Archivdatei kann ein oder mehrere WSDL-Dokumente enthalten. Wenn das Abholverzeichnis eine WSDL-Archivdatei und ein WSDL-Dokument enthält, gibt der URI nur das WSDL-Archiv zurück. Das unterstützte Format der Archivdatei ist eine `.zip`-Datei. Es ist auch möglich, die WSDL-Archivdatei oder das WSDL-

Dokument unter Verwendung von SPI und CEMT zu erkennen. Das WSDL-Dokument in einer WSDL-Archivdatei kann für die SOAP-Nachrichtenprüfung verwendet werden.

CICS für die Verwendung des WebSphere MQ-Transports verwenden

Um den WebSphere MQ-Transport mit SOAP-Web-Services in CICS verwenden zu können, müssen Sie Ihre CICS-Region entsprechend konfigurieren.

About this task

Anmerkung: Sie können den Websphere MQ-Transport nicht für JSON-Web-Services verwenden.

Procedure

1. Schließen Sie die IBM MQ-Bibliothek *thlqual*.SCSQAUTH in die STEPLIB-Verkettung in Ihrer CICS-Prozedur ein. Schließen Sie die Bibliothek nach den CICS-Bibliotheken ein, um sicherzustellen, dass der korrekte Code verwendet wird. *thlqual* ist das übergeordnete Qualifikationsmerkmal für die IBM MQ-Bibliotheken.
2. Schließen Sie die folgenden IBM MQ-Bibliotheken in die DFHRPL-Verkettung in Ihrer CICS-Prozedur ein. Schließen Sie die Bibliotheken nach den CICS-Bibliotheken ein, um sicherzustellen, dass der korrekte Code verwendet wird.

thlqual.SCSQCICS

thlqual.SCSQLOAD

thlqual.SCSQAUTH

thlqual ist das übergeordnete Qualifikationsmerkmal für die IBM MQ-Bibliotheken. Wenn Sie den API-Steuerübergabeexit von CICS-MQ (CSQCAPX) verwenden, fügen Sie auch den Namen der Bibliothek hinzu, die das Lademodul für das Programm enthält. Die SCSQCICS-Bibliothek ist nur erforderlich, wenn Sie von IBM MQ bereitgestellte Beispiele ausführen möchten. Andernfalls kann sie aus der CICS-Prozedur entfernt werden.

3. Definieren und installieren Sie eine MQCONN-Ressource für die CICS-Region. Die MQCONN-Ressource gibt die Attribute der Verbindung zwischen CICS und IBM MQ an, einschließlich des Namens des IBM MQ-Standardwarteschlangenmanagers oder der Gruppe mit gemeinsamer Warteschlange für die Verbindung. Anweisungen finden Sie unter Defining and installing an MQCONN resource.
4. Geben Sie den CICS-Systeminitialisierungsparameter **MQCONN=YES** an, um die WebSphere MQ-Verbindung in CICS bei der CICS-Initialisierung automatisch zu starten. Details finden Sie unter MQCONN system initialization parameter.
5. Wenn Sie den CICS-MQ-Adapter in einem CICS-System verwenden, das über eine regionsübergreifende Kommunikation mit fernen CICS-Systemen verfügt, stellen Sie sicher, dass die IRC-Facility geöffnet (OPEN) ist, bevor Sie den Adapter starten, indem Sie den CICS-Systeminitialisierungsparameter IRCSTRT=YES angeben. Die IRC-Facility muss geöffnet (OPEN) sein, wenn die IRC-Zugriffsmethode als speicherübergreifend definiert ist, d. h. ACCESSMETHODOD(XM).
6. Stellen Sie sicher, dass die codierten Zeichensatz-IDs (CCSIDs), die von Ihrem Warteschlangenmanager und von CICS verwendet werden, und die UTF-8- und UTF-16-Codepages für z/OS-Umsetzungsservices konfiguriert sind. Die CICS-Codepage wird im Systeminitialisierungsparameter **LOCALCCSID** angegeben.

7. Aktualisieren Sie Ihre CICS-CSD wie folgt:
 - a. Wenn Sie Ihre CSD nicht mit früheren Versionen von CICS gemeinsam nutzen, entfernen Sie die Gruppen CSQCAT1 und CSQCKB und alle Kopien dieser Gruppen oder Elemente aus diesen Gruppen. Sie müssen außerdem CKQQ TDQUEUE aus der Gruppe CSQCAT1 löschen. Die Definition für CKQQ wird jetzt in der CICS-CSD-Gruppe DFHDCTG bereitgestellt.
 - b. Wenn Sie Ihre CSD mit früheren Versionen von CICS gemeinsam nutzen, stellen Sie sicher, dass CSQCAT1 und CSQCKB und alle Kopien dieser Gruppen oder ihrer Inhalte nicht für CICS TS 4.1 oder CICS TS 3.2 installiert sind. Sie müssen außerdem CKQQ TDQUEUE aus der Gruppe CSQCAT1 löschen. Die Definition für CKQQ wird jetzt in der CICS-CSD-Gruppe DFHDCTG bereitgestellt. Für CICS TS-Versionen vor CICS TS 3.2 installieren Sie die Gruppen CSQCAT1 und CSQCKB als Teil einer Gruppenliste, nachdem Sie DFHLIST installiert haben, um die Gruppe DFHMQ zu überschreiben und die erforderlichen Definitionen korrekt zu installieren.
8. Aktualisieren Sie die IBM MQ-Definitionen für die Warteschlange für nicht zustellbare Nachrichten, die Standardübertragungswarteschlange und die CICS-MQ-Adapterobjekte. Sie können das Beispiel CSQ4INYG verwenden, müssen aber eventuell den Namen der Initialisierungswarteschlange so ändern, dass er mit dem Standardnamen der Initialisierungswarteschlange in der MQINI-Resourcendefinition in Ihrer CICS-Region übereinstimmt. Sie können dieses Member in der CSQINP2 DD-Verkettung der Warteschlangenmanager-Startprozedur verwenden oder als Eingabe für die COMMAND-Funktion des CSQUTIL-Dienstprogramms, um die erforderlichen DEFINE-Befehle auszugeben. Es wird empfohlen, das CSQUTIL-Dienstprogramm zu verwenden, weil Sie diese Objekte dann nicht bei jedem Neustart von IBM MQ neu definieren müssen.

IBM MQ-Transport

CICS kann SOAP-Nachrichten mithilfe des IBM MQ-Transports empfangen und an IBM MQ senden, sowohl in der Rolle des Service-Providers als auch in der des Service-Requesters.

Als **Service-Provider** verwendet CICS IBM MQ-Triggering, um SOAP-Nachrichten aus einer Anwendungswarteschlange zu verarbeiten. Das Triggering verwendet eine Initialisierungswarteschlange und lokale Warteschlangen. Eine lokale (Anwendungs-)Warteschlangendefinition enthält die folgenden Informationen:

- Die Kriterien für den Zeitpunkt, an dem eine Auslösenachricht generiert wird. Zum Beispiel wenn die erste Nachricht in der lokalen Warteschlange ankommt, oder für alle Nachrichten, die in der lokalen Warteschlange ankommen. Geben Sie für die CICS-SOAP-Verarbeitung an, dass das Triggering auftritt, wenn die erste Nachricht in der lokalen Warteschlange ankommt.
Die lokale Warteschlangendefinition kann auch angeben, dass Auslöserdaten an die Zielanwendung übergeben werden. Im Fall der CICS-SOAP-Verarbeitung (Transaktion CPIL) gibt sie die Standardziel-URL an, die verwendet werden soll, wenn dies nicht mit der eingehenden Nachricht übergeben wird.
- Der *Prozessname*, der die *Prozessdefinition* angibt. Die Prozessdefinition beschreibt, wie die Nachricht verarbeitet wird. Geben Sie im Fall einer CICS-SOAP-Verarbeitung die Transaktion CPIL an.
- Der Name der Initialisierungswarteschlange, an die die Auslösenachricht gesendet werden soll.

Wenn eine Nachricht in der lokalen Warteschlange ankommt, generiert und sendet der Warteschlangenmanager eine Auslösenachricht an die angegebene Initialisierungswarteschlange. Die Auslösenachricht schließt die Informationen aus der Prozessdefinition mit ein. Der Auslösemonitor ruft die Auslösenachricht aus der Initia-

lisierungwarteschlange ab und plant die Transaktion CPIL zum Starten der Verarbeitung der Nachrichten in der lokalen Warteschlange. Weitere Informationen zum Triggering finden Sie unter Task initiator or trigger monitor (CKTI).

Sie können CICS so konfigurieren, dass der (von IBM MQ bereitgestellte) Auslösemonitor beim Eintreffen einer Nachricht in einer lokalen Warteschlange die Transaktion CPIL zum Verarbeiten der Nachrichten in der lokalen Warteschlange und zum Steuern der CICS-Pipeline für die Verarbeitung der SOAP-Nachrichten in der Warteschlange plant.

Wenn CICS eine Antwort auf eine SOAP-Nachricht erstellt, die von IBM MQ empfangen wird, wird die Korrelations-ID mit der Nachrichten-ID der Eingabenachricht gefüllt, es sei denn, die Berichtsoption MQRO_PASS_CORREL_ID wurde festgelegt. Wenn diese Berichtsoption festgelegt wurde, wird die Korrelations-ID aus der Eingabenachricht an die Antwort weitergegeben.

Als **Service-Requester** können Sie für ausgehende Anforderungen angeben, dass die Antworten für den Ziel-Web-Service in einer bestimmten Antwortwarteschlange zurückgegeben werden.

In beiden Fällen erfordern CICS und IBM MQ, dass in der Konfiguration die erforderlichen Ressourcen und Warteschlangen definiert sind.

Lokale Warteschlangen in einem Service-Provider definieren

Um den WebSphere MQ-Transport in einem Service-Provider zu verwenden, müssen Sie eine oder mehrere lokale Warteschlangen definieren, die Anforderungsnachrichten bis zu ihrer Verarbeitung speichern. Und Sie müssen einen Auslöseprozess definieren, der die CICS-Transaktion angibt, die die Anforderungsnachrichten verarbeiten wird.

Procedure

1. Definieren Sie eine Initialisierungwarteschlange. Verwenden Sie den folgenden Befehl:

```
DEFINE
QLOCAL('initialisierungwarteschlange')
DESCR('beschreibung')
```

Dabei entspricht *initialisierungwarteschlange* dem für das Attribut QNAME angegebenen Wert der installierten MQMONITOR-Ressourcendefinition für die CICS-Region, oder dem für das Attribut INITQNAME angegebenen Wert der installierten MQCONN-Ressourcendefinition.

2. Definieren Sie für jede lokale Anforderungwarteschlange ein QLOCAL-Objekt. Verwenden Sie den folgenden Befehl:

```
DEFINE
QLOCAL('warteschlangennamenname')
DESCR('beschreibung')
PROCESS(prozessname)
INITQ('initialisierungwarteschlange')
TRIGGER
TRIGTYPE(FIRST)
TRIGDATA('standardzielservice')
BOTHRESH(nnn)
BOQNAME('name_der_warteschlange_zum_erneuten_einreihen')
```

Dabei gilt Folgendes:

- *warteschlangenname* ist der Name der lokalen Warteschlange.
 - *prozessname* ist der Name der Prozessinstanz, die die Anwendung angibt, die vom Warteschlangenmanager im Fall eines Auslöseereignisses gestartet wird. Geben Sie denselben Namen für jedes QLOCAL-Objekt an.
 - *initialisierungswarteschlange* ist der Name der Initialisierungswarteschlange, die verwendet werden soll, z. B. die im Attribut QNAME der installierten MQMONITOR-Ressourcendefinition für die CICS-Region angegebene Initialisierungswarteschlange.
 - *standardzielservice* ist der Standardzielservice, der verwendet werden soll, wenn kein Service in der Anforderung angegeben ist. Der Zielservice hat das Format '/string' und wird zum Abgleich des Pfads einer URIMAP-Definition verwendet, z. B. '/SOAP/test/test1'. Das erste Zeichen muss '/' sein.
 - *nnn* ist die Anzahl von Wiederholungsversuchen.
 - *name_der_warteschlange_zum_erneuten_einreihen* ist der Name der Warteschlange, an die fehlgeschlagene Nachrichten gesendet werden.
3. Definieren Sie ein PROCESS-Objekt, das den Auslöseprozess angibt. Verwenden Sie den folgenden Befehl:

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

Dabei gilt Folgendes:

prozessname ist der Name des Prozesses und muss dem Namen entsprechen, der beim Definieren der Anforderungswarteschlangen angegeben wird.

Lokale Warteschlangen in einem Service-Requester definieren

Wenn Sie den WebSphere MQ-Transport für ausgehende Anforderungen in einem Service-Requester verwenden, können Sie in dem URI für den Ziel-Web-Service angeben, dass Ihre Antworten in einer vordefinierten Antwortwarteschlange zurückgegeben werden sollen. In diesem Fall müssen Sie jede Antwortwarteschlange mit einem QLOCAL-Objekt definieren.

About this task

Wenn der einer Anforderung zugeordnete URI keine Antwortwarteschlange angibt, verwendet CICS eine dynamische Warteschlange für die Antwort.

Procedure

Optional: Verwenden Sie den folgenden Befehl, um die einzelnen QLOCAL-Objekte zu definieren, die eine vordefinierte Antwortwarteschlange angeben.

```
DEFINE  
QLOCAL('antwortwarteschlange')  
DESCR('beschreibung')  
BOTHRESH(nnn)
```

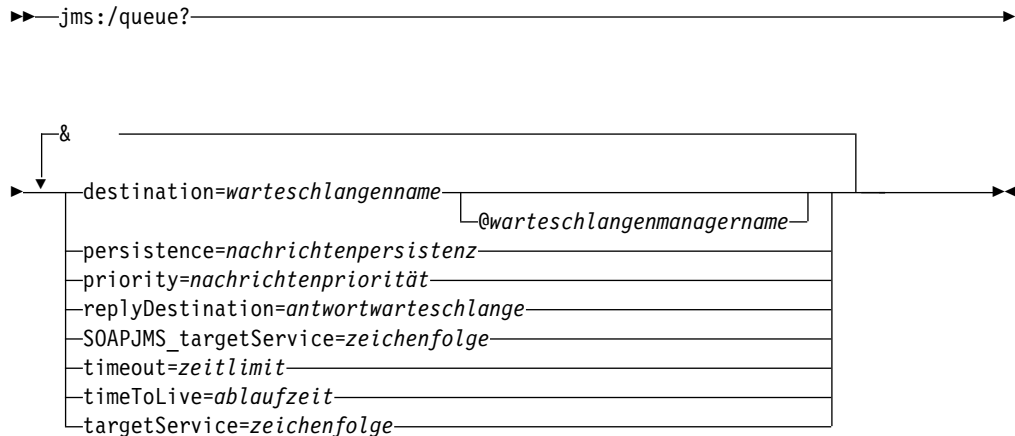
Dabei gilt Folgendes:

antwortwarteschlange ist der lokale Warteschlangenname.
nnn ist die Anzahl von Wiederholungsversuchen.

URI für den IBM MQ-Transport

Wenn für die Kommunikation zwischen dem Service-Requester und dem Service-Provider IBM MQ verwendet wird, hat der URI des Ziels ein Format, das das Ziel als Warteschlange identifiziert und Angaben zur Verarbeitung der Anforderung und Antwort von IBM MQ einschließt.

Syntax



Optionen

CICS verwendet die folgenden Optionen. Andere Web-Service-Provider verwenden möglicherweise weitere Optionen, die hier nicht beschrieben werden. Der gesamte URI wird an den Service-Provider übergeben, aber CICS ignoriert alle Optionen, die es nicht unterstützt und die im URI codiert sind. In CICS spielt die Groß-/Kleinschreibung der Optionsnamen keine Rolle. In anderen Implementierungen, die diesen Typ von URI unterstützen, muss die Groß-/Kleinschreibung jedoch beachtet werden.

destination=warteschlangenname [*@warteschlangenmanagername*]

warteschlangenname ist der Name der Eingabewarteschlange im Zielwarteschlangenmanager.

warteschlangenmanagername ist der Name des Zielwarteschlangenmanagers.

persistence=nachrichtenpersistenz

Geben Sie eine der folgenden Optionen an:

- 0 Persistenz wird durch die Standardwarteschlange definiert.
- 1 Nachrichten sind nicht persistent.
- 2 Nachrichten sind persistent.

Wenn die Option nicht angegeben oder nicht ordnungsgemäß angegeben wurde, wird die Standardwarteschlangenpersistenz verwendet.

priority=nachrichtenpriorität

Gibt die Nachrichtenpriorität an. CICS unterstützt ganzzahlige Werte im Bereich 0 - 9 für Nachrichtenprioritäten, wobei 9 Nachrichten mit der höchsten Priorität und 0 Nachrichten mit der niedrigsten Priorität zugewiesen wird. Geben Sie alternativ **-1** an, um die Standardpriorität zu verwenden, die für die Zielwarteschlange definiert ist.

replyDestination=antwortwarteschlange

Gibt die Warteschlange an, die für die Antwortnachricht verwendet werden soll. Wird diese Option nicht angegeben, verwendet CICS eine dynamische Warteschlange für die Antwortnachricht. Sie müssen die Antwortwarteschlange in einem QLOCAL-Objekt definieren, bevor Sie diese Option verwenden können.

SOAPJMS_targetService=zeichenfolge

Gibt den Zielservice an. Wenn CICS der Service-Provider ist, sollte der Zielservice das Format '/string' haben, da CICS dies beim Versuch eines Abgleichs mit URIMAP als Pfad verwendet. Wird diese Option nicht angegeben, wird der Wert, der in TRIGDATA in der Eingabewarteschlange des Service-Providers angegeben ist, verwendet.

timeout=zeitlimit

Das Zeitlimit in Millisekunden, für das der Service-Requester auf eine Antwort wartet. Wenn der Wert Null angegeben wird oder wenn diese Option weggelassen wird, wird die Anforderung nicht ausgeführt.

timeToLive=ablaufzeit

Gibt die Ablaufzeit für die Anforderung in Millisekunden an. Wenn die Option nicht angegeben oder falsch angegeben wurde, läuft die Anforderung nicht ab.

targetService=zeichenfolge

Gibt den Zielservice an. Wenn CICS der Service-Provider ist, sollte der Zielservice das Format '/string' haben, da CICS dies beim Versuch eines Abgleichs mit URIMAP als Pfad verwendet. Wird diese Option nicht angegeben, wird der Wert, der in TRIGDATA in der Eingabewarteschlange des Service-Providers angegeben ist, verwendet.

Beispiel

Das Beispiel zeigt einen URI für den IBM MQ-Transport:

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

Weitere Informationen zu "connectionFactory" und "initialContextFactory" finden Sie unter IBM MQ product documentation.

CICS für die Unterstützung persistenter Nachrichten konfigurieren

CICS bietet Unterstützung für das Senden von persistenten Nachrichten mithilfe des WebSphere MQ-Transportprotokolls an eine Web-Service-Provider-Anwendung, die in einer CICS-Region implementiert ist.

About this task

CICS verwendet Business Transaction Services (BTS), um sicherzustellen, dass persistente Nachrichten im Fall eines CICS-Systemfehlers wiederhergestellt werden. Führen Sie dafür die folgenden Schritte aus:

Procedure

1. Verwenden Sie IDCAMS, um die lokale Anforderungswarteschlange und die Repository-Datei für MVS zu definieren. Sie müssen einen geeigneten Wert für STRINGS für die Dateidefinition angeben. Der Standardwert 1 reicht wahrscheinlich nicht aus. Wir empfehlen, stattdessen den Wert 10 anzugeben.
2. Definieren Sie die lokale Anforderungswarteschlange und Repository-Datei für CICS. Details zur Definition der lokalen Anforderungswarteschlange für CICS

finden Sie unter „Lokale Warteschlangen in einem Service-Provider definieren“ auf Seite 52. Sie müssen einen geeigneten Wert für STRINGS in der Dateidefinition angeben. Der Standardwert 1 reicht wahrscheinlich nicht aus. Wir empfehlen, stattdessen den Wert 10 anzugeben.

3. Definieren Sie eine PROCESSTYPE-Ressource namens DFHMQSOA und verwenden Sie dabei den Namen der Repository-Datei als Wert für die FILE-Option.
4. Stellen Sie sicher, dass während der Verarbeitung einer persistenten Nachricht ein Programm einen Befehl **EXEC CICS SYNCPOINT** ausgibt, bevor der erste implizite Synchronisationspunkt angefordert wird. Beispielsweise während ein SPI-Befehl wie **EXEC CICS CREATE TDQUEUE** implizit einen Synchronisationspunkt annimmt. Die Ausgabe eines Befehls **EXEC CICS SYNCPOINT** ist die Bestätigung, dass die persistente Nachricht erfolgreich verarbeitet wurde. Wenn ein Programm nicht explizit einen Synchronisationspunkt anfordert, bevor es versucht, implizit einen Synchronisationspunkt zu verwenden, wird ein ASP7-Abbruch ausgegeben.

Results

What to do next

Wenn der Web-Service abgebrochen oder zurückgesetzt wird, werden für unidirektionale Anforderungsnachrichten ausreichend Informationen zurückbehalten, damit eine Transaktion oder ein Programm die fehlgeschlagene Anforderung erneut ausführen kann bzw. den Fehler angemessen berichten kann. Sie müssen diese Wiederherstellungsaktion oder dieses Wiederherstellungsprogramm bereitstellen. Ausführliche Informationen finden Sie unter „Persistente Nachrichtenverarbeitung“.

Persistente Nachrichtenverarbeitung

Wenn eine Web-Service-Anforderung in einer persistenten WebSphere MQ-Nachricht empfangen wird, erstellt CICS einen eindeutigen BTS-Prozess mit dem Prozessstyp DFHMQSOA. Daten, die sich auf die eingehende Anforderung beziehen, werden in BTS-Datencontainern erfasst, die dem Prozess zugeordnet sind.

Der Prozess soll dann asynchron ausgeführt werden. Wenn der Web-Service erfolgreich abgeschlossen und festgeschrieben wird, löscht CICS den BTS-Prozess. Dies passiert auch dann, wenn ein SOAP-Fehler generiert und an den Web-Service-Requester zurückgegeben wird.

Fehlerbehandlung

Wenn beim Erstellen des erforderlichen BTS-Prozesses ein Fehler auftritt, wird die Web-Service-Transaktion abnormal beendet und die eingehende Web-Service-Anforderung wird nicht verarbeitet. Wenn BTS nicht verwendet werden kann, wird die Nachricht DFHPI0117 ausgegeben und CICS wird ohne BTS fortgesetzt, unter Verwendung des vorhandenen kanalbasierten Containerverfahrens.

Wenn ein CICS-Fehler auftritt, bevor der Web-Service gestartet oder abgeschlossen wird, stellt die BTS-Wiederherstellung sicher, dass der Prozess neu terminiert wird, wenn CICS erneut gestartet wird.

Wenn der Web-Service abnormal beendet und zurückgesetzt wird, wird der BTS-Prozess mit einem Status ABENDED als abgeschlossen markiert. Für Anforderungsnachrichten, die eine Antwort erfordern, wird ein SOAP-Fehler an den Web-Service-Requester zurückgegeben. Der BTS-Prozess wird abgebrochen und CICS bewahrt keine Informationen über die fehlgeschlagene Anforderung auf. CICS gibt

die Nachricht DFHBA0104 in der Warteschlange mit transienten Daten CSBA und die Nachricht DFHPI0117 in der Warteschlange mit transienten Daten CPIO aus.

Für unidirektionale Nachrichten gibt es keine Möglichkeit, Informationen zum Fehler an den Requester zurückzugeben, deshalb verbleibt der BTS-Prozess im Status COMPLETE ABENDED. CICS gibt die Nachricht DFHBA0104 in der Warteschlange mit transienten Daten CSBA und die Nachricht DFHPI0116 in der Warteschlange mit transienten Daten CPIO aus.

Sie können die Transaktion CBAM verwenden, um alle Prozesse mit dem Status COMPLETE ABENDED anzuzeigen, oder Sie können eine Wiederherstellungstransaktion bereitstellen, um auf COMPLETE ABENDED-Prozesse in DFHMQSOA zu überprüfen und entsprechende Maßnahmen zu ergreifen.

Ihre Wiederherstellungstransaktion könnte sein:

1. Zurücksetzen des BTS-Prozesses mithilfe des Befehls **RESET ACQPROCESS**.
2. Ausgeben des Befehls **RUN ASYNC** zum Wiederholen des fehlgeschlagenen Web-Service. Für den Befehl kann es einen Wiederholungszähler in einem anderen Datencontainer im Prozess geben, um ein wiederholtes Fehlschlagen zu vermeiden.
3. Verwenden von Informationen in den zugehörigen Datencontainern, um über das Problem zu berichten:

Der Datencontainer DFHMQORIGINALMSG enthält die aus WebSphere MQ empfangene Nachricht, die möglicherweise RFH2-Header enthält.

Der Datencontainer DFHMQMSG enthält die WebSphere MQ-Nachricht, aus der alle RFH2-Header entfernt wurden.

Der Datencontainer DFHMQDLQ enthält den Namen der Warteschlange für nicht zustellbare Nachrichten, die der ursprünglichen Nachricht zugeordnet ist.

Der Datencontainer DFHMQCONT enthält den WebSphere MQ-Steuerblock MQMD, der sich auf den Befehl **MQ GET** für die ursprüngliche Nachricht bezieht.

Interoperabilität zwischen dem Web-Service-Assistenten und WSRR

Der CICS-Web-Service-Assistent kann mit IBM WebSphere Service Registry and Repository (WSRR) interagieren. Verwenden Sie WSRR, um Web-Services, die Sie anfordern, schneller zu finden und für Web-Services, die Sie bereitstellen, eine Versionssteuerung zu erzwingen.

Sowohl DFHLS2WS als auch DFHWS2LS enthalten Parameter, um mit WSRR zu interagieren. DFHLS2WS enthält außerdem einen optionalen Parameter, sodass Sie Ihre eigenen angepassten Metadaten zu dem WSDL-Dokument in WSRR hinzufügen können.

Wenn Sie möchten, dass der Web-Service-Assistent sicher mit WSRR kommuniziert, können Sie SSL-Verschlüsselung verwenden. Sowohl DFHLS2WS als auch DFHWS2LS enthalten Parameter für die Verwendung von SSL-Verschlüsselung.

Informationen zur Verwendung von SSL mit dem Web-Service-Assistenten finden Sie unter „Beispiel für die Verwendung von SSL mit dem Web-Service-Assistenten und WSRR“ auf Seite 58.

Beispiel für die Verwendung von SSL mit dem Web-Service-Assistenten und WSRR

Sie können sicher zwischen dem Web-Service-Assistenten und einem IBM WebSphere Service Registry and Repository-Server (WSRR) interagieren, indem Sie SSL-Verschlüsselung (Secure Socket Layer) verwenden. Um SSL-Verschlüsselung verwenden zu können, benötigen Sie einen Keystore und einen Truststore. Außerdem müssen Sie bestimmte Parameter für den Web-Service-Assistenten angeben.

About this task

Führen Sie die folgenden Schritte aus, um SSL-Verschlüsselung für Interaktionen zwischen dem Web-Service-Assistenten und WSRR zu verwenden.

Procedure

1. Erstellen Sie einen Keystore für Ihre privaten Schlüssel und Ihre Zertifikate für öffentliche Schlüssel.
 - a. Sie können einen Keystore mithilfe eines Schlüsselkonfigurationsprogramms wie IBM Key Management Utility (iKeyman) erstellen.
 - b. Geben Sie den Parameter **SSL-KEYSTORE** in DFHWS2LS oder DFHLS2WS mit dem vollständig qualifizierten Namen des Keystores an, den Sie erstellt haben.
 - c. Optional: Geben Sie den Parameter **SSL-KEYPWD** in DFHWS2LS oder DFHLS2WS mit dem Kennwort des Keystores an, den Sie erstellt haben.
2. Erstellen Sie einen Truststore für alle Ihre vertrauenswürdigen Zertifikate der Stammzertifizierungsstelle. Diese Zertifikate dienen zum Feststellen der Vertrauenswürdigkeit aller eingehenden Zertifikate für öffentliche Schlüssel.
 - a. Sie können einen Truststore mithilfe eines Schlüsselkonfigurationsprogramms wie IBM Key Management Utility (iKeyman) erstellen.
 - b. Geben Sie den Parameter **SSL-TRUSTSTORE** in DFHWS2LS oder DFHLS2WS mit dem vollständig qualifizierten Namen des Truststores an, den Sie erstellt haben.
 - c. Optional: Geben Sie den Parameter **SSL-TRUSTPWD** in DFHWS2LS oder DFHLS2WS mit dem Kennwort des Truststores an, den Sie erstellt haben.
3. Testen Sie, ob der Web-Service-Assistent in der Lage ist, mit WSRR unter Verwendung von SSL-Verschlüsselung zu kommunizieren.
 - a. Sie können die von IBM WebSphere Application Server bereitgestellten Beispieldateien verwenden, um den Web-Service-Assistenten mit WSRR zu testen.
 - Von WebSphere Application Server werden die Beispiel-Keystores `DummyClientKeyFile.jks` und `DummyServerKeyFile.jks` bereitgestellt.
 - Von WebSphere Application Server werden die Beispiel-Truststores `DummyClientTrustFile.jks` und `DummyServerTrustFile.jks` bereitgestellt.
 - b. Ersetzen Sie die Schlüssel Dateien der Beispiel-Keystores und -Truststores. Diese Schlüssel werden mit WebSphere Application Server ausgeliefert und müssen aus Sicherheitsgründen ersetzt werden.

Results

Der Web-Service-Assistent kann jetzt SSL-Verschlüsselung verwenden, um mit WSRR sicher über ein Netz zu kommunizieren.

Web-Service-Infrastruktur erstellen

Zur Implementierung eines Web-Service in CICS müssen Sie die erforderliche Transportinfrastruktur erstellen und mindestens eine Pipeline definieren, die Ihre Web-Service-Anforderungen verarbeiten soll. In der Regel kann eine Pipeline Anforderungen für viele verschiedene Web-Services verarbeiten. Und wenn Sie einen neuen Web-Service in Ihrem CICS-System implementieren, können Sie eine bereits vorhandene Pipeline verwenden.

Web-Service-Infrastruktur

CICS-Anwendungen in einer CICS-Region können entweder einen Service für Anwendungen bereitstellen oder einen Service von Anwendungen anfordern, die sich außerhalb dieser Region befinden, indem eine Web-Service-Pipeline eingesetzt wird. Wenn CICS ein Service-Provider ist, stellt die CICS-Anwendung einen Service für die externe Anwendung bereit. Wenn CICS ein Service-Requester ist, stellt die externe Anwendung einen Service für die CICS-Anwendung bereit. Web-Service-Pipelines können ggf. für die Verwendung von zEnterprise Application Assist Processor (zAAP) konfiguriert werden.

CICS als Service-Provider

Damit CICS einen Service für einen externen Service-Requester bereitstellt, muss es die Serviceanforderung empfangen und sie über eine Pipeline an das Zielanwendungsprogramm übergeben. Die Antwort der Anwendung wird über dieselbe Pipeline an den Service-Requester zurückgegeben.

Abb. 15 auf Seite 60 zeigt eine Beispielkonfiguration der Architektur und Ressourcen, die erforderlich sind, um eine Anforderung von einem externen Service-Requester zu verarbeiten, wenn CICS ein Service-Provider ist, der eine Java-Pipeline verarbeitet.

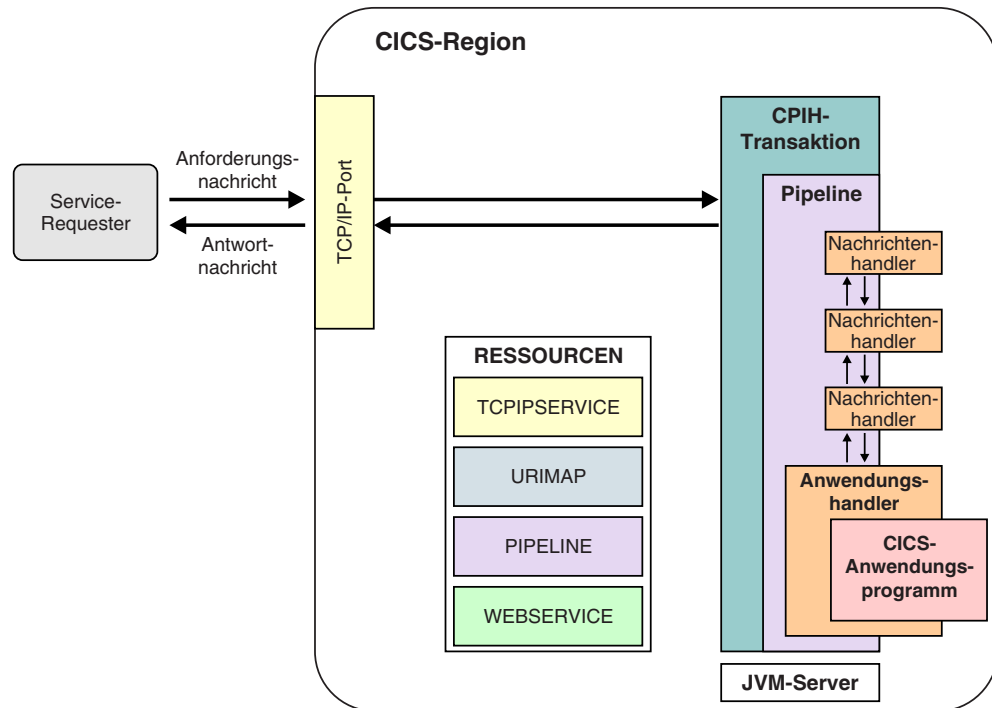


Abbildung 15. Architektur und Ressourcen für einen Service-Provider

Um eine Anforderung zu verarbeiten, muss CICS die folgenden Operationen ausführen:

1. Empfangen der Anforderung vom Service-Requester.
Die TCPIPService-Ressource gibt einen Port für eingehende Anforderungen an. Dieser Port wird von der von CICS bereitgestellten Socket-Listener-Transaktion überwacht.
2. Überprüfen der Anforderung und Extrahieren der Inhalte, die für das Zielanwendungsprogramm relevant sind.
Wenn die Anforderungsnachricht am entsprechenden Port empfangen wird, werden die URIMAP-Ressourcendefinitionen nach einer URIMAP-Ressource durchsucht, deren USAGE-Attribut auf PIPELINE und deren PATH-Attribut auf den in der Anforderung gesendeten URI gesetzt ist. Wenn eine entsprechende URIMAP-Definition gefunden wird, werden die PIPELINE- und WEBSERVICE-Definitionen aus den PIPELINE- und WEBSERVICE-Attributen der URIMAP-Definition verwendet. Das Attribut TRANSACTION der URIMAP-Definition bestimmt den Namen der Transaktion, die zur Verarbeitung der Pipeline angehängt werden soll. Standardmäßig ist das die Transaktion CPIH. Die URIMAP-Definition gibt auch die zu verwendenden PIPELINE- und WEBSERVICE-Ressourcen an. Diese Ressourcen steuern die Verarbeitung, die CICS ausführt.
3. Aufrufen des Anwendungsprogramms, wobei aus der Anforderung extrahierte Daten übergeben werden.
Die Nachrichtenhandler in der Pipeline und die Anwendungshandler konvertieren die Anforderungsnachricht in die Anwendungssprachstruktur, die das Service-Provider-Anwendungsprogramm erwartet. Das Programm verarbeitet diese Eingabe und gibt eine Antwort an den Anwendungshandler zurück.
4. Erstellen einer Antwort unter Verwendung der Daten, die vom Anwendungsprogramm zurückgegeben werden, und senden Sie sie an den Service-Requester.

Der Anwendungshandler und die Nachrichtenhandler konvertieren die Antwortnachricht, die von der Service-Provider-Anwendung empfangen wurde, in eine Nachricht im Format der ursprünglichen Anforderung. Diese Nachricht wird an den Service-Requester zurückgesendet.

Manche Teile der Verarbeitung in der Pipeline können mithilfe von zEnterprise Application Assist Processor (zAAP) ausgeführt werden, wenn die Pipeline entsprechend konfiguriert ist. Weitere Informationen finden Sie unter „Java-basierte SOAP-Pipelines“ auf Seite 62.

CICS als Service-Requester

Damit CICS einen externen Service aufruft, sendet ein Anwendungsprogramm eine Anforderung, die über eine Pipeline an einen Zielservice übergeben wird. Die Antwort des Service wird über dieselbe Pipeline an das Anwendungsprogramm zurückgegeben.

Abb. 16 zeigt eine Beispielkonfiguration der Architektur und Ressourcen, die eine Anforderung aus einem CICS-Anwendungsprogramm verarbeiten müssen, um mithilfe einer Java-Pipeline Daten von einem Service-Provider abzurufen, der sich außerhalb der CICS-Region befindet.

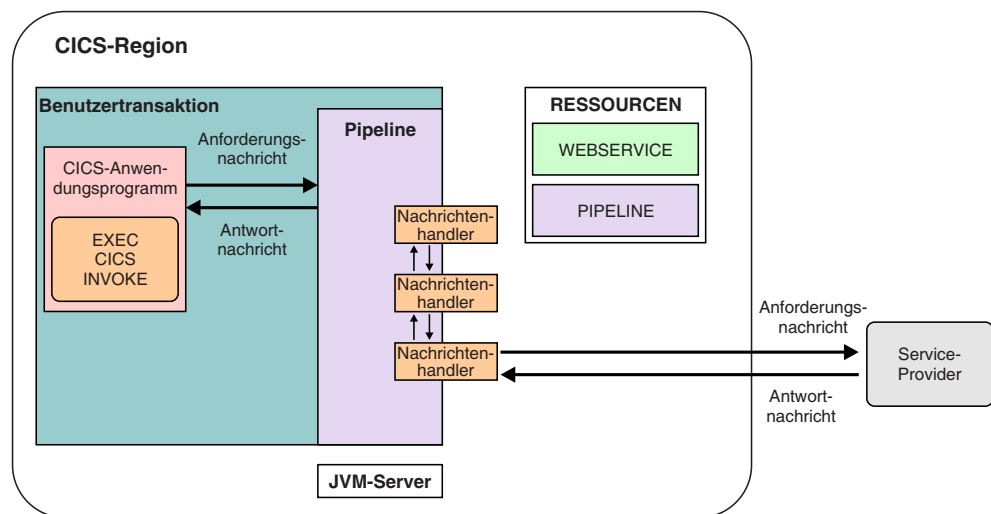


Abbildung 16. Architektur und Ressourcen für einen Service-Requester

Um eine Anforderung zu verarbeiten, muss CICS die folgenden Operationen ausführen:

1. Erstellen einer Anforderung unter Verwendung von Daten, die vom Anwendungsprogramm bereitgestellt werden.

Wenn das CICS-Anwendungsprogramm eine Anforderung an einen Service-Provider initialisiert, der sich außerhalb der CICS-Region befindet, ruft die Requester-Anwendung den Befehl **EXEC CICS INVOKE SERVICE** auf. Mit dem Befehl **EXEC CICS INVOKE SERVICE** wird die Pipeline aufgerufen. Die Pipeline konvertiert die Anwendungssprachstruktur in eine Sprache, die der Service-Provider verarbeiten kann, z. B. eine SOAP-Nachricht.

2. Senden der Anforderung an den Service-Provider.

CICS sendet die Anforderungsnachricht per HTTP oder WebSphere MQ an den fernen Service-Provider.

3. Empfangen einer Antwort vom Service-Provider.

Wenn die Antwortnachricht des Service-Providers empfangen wird, übergibt CICS die Nachricht an die Pipeline zurück.

4. Überprüfen der Antwort und Extrahieren der Inhalte, die für das Ursprungsanwendungsprogramm relevant sind.

Die Pipeline konvertiert die Antwortnachricht des Service-Providers in die Anwendungssprachstruktur, die an das Anwendungsprogramm übergeben wird. Anschließend wird die Steuerung an das Anwendungsprogramm zurückgegeben.

Manche Teile der Verarbeitung in der Pipeline können mithilfe von zEnterprise Application Assist Processor (zAAP) ausgeführt werden, wenn die Pipeline entsprechend konfiguriert ist. Weitere Informationen finden Sie unter „Java-basierte SOAP-Pipelines“.

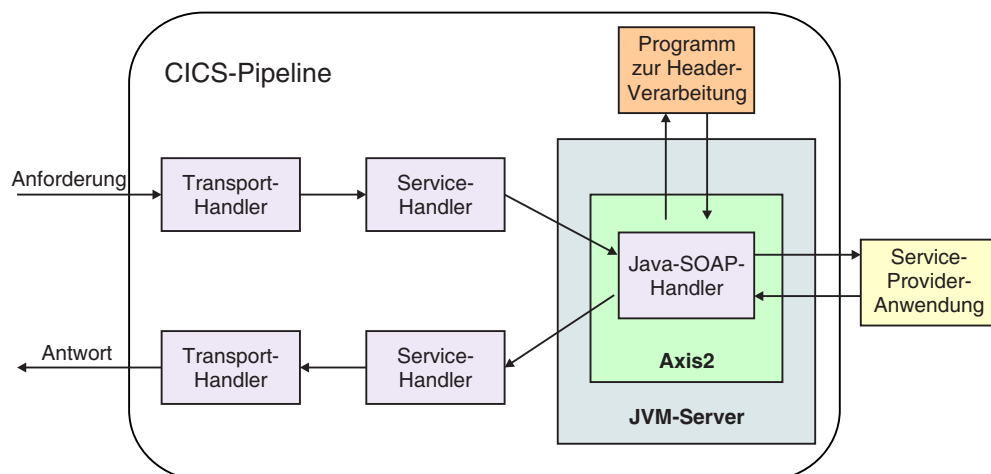
Java-basierte SOAP-Pipelines

CICS unterstützt die Java-basierte SOAP-Engine Axis2 für die Verarbeitung von Web-Service-Anforderungen in Provider- und Requester-Pipelines. Da Axis2 Java verwendet, kommt die SOAP-Verarbeitung für die Auslagerung auf den zAAP-Processor (zAAP - zEnterprise Application Assist Processor) infrage.

Axis2 ist eine Open Source-Web-Service-Engine der Apache Foundation und wird mit CICS bereitgestellt, um SOAP-Nachrichten in einer Java-Umgebung zu verarbeiten. Sie können entscheiden, Axis2 nicht zu verwenden, indem Sie einen Java-SOAP-Handler zu Ihrer Pipelinekonfigurationsdatei hinzufügen und einen JVM-Server für die die Axis2-Verarbeitung erstellen.

Für die Aktivierung von Axis2 ist es nicht erforderlich, die Bindungsdateien für vorhandene Web-Services, die die Pipeline verwenden, erneut zu generieren. Die Antwortzeiten können länger sein, wenn Sie Axis2 verwenden, aber Sie können die SOAP-Verarbeitung an zAAP auslagern. Weitere Informationen zum Auslagern an zAAP finden Sie unter Java support in CICS.

Wenn CICS ein Service-Provider ist, verwendet der Java-basierte Terminal-Handler Axis2 zum Parsen des SOAP-Envelopes für eine Anforderungsnachricht. Sie können Programme zur Headerverarbeitung verwenden, um alle SOAP-Header zu verarbeiten, die der SOAP-Nachricht zugeordnet sind. Axis2 erstellt außerdem die SOAP-Antwortnachricht. Dieser Prozess ist im folgenden Diagramm dargestellt:



Wenn CICS ein Service-Requester ist, verwendet der Java-basierte ursprüngliche Handler in der Pipeline Axis2, um den SOAP-Envelope für einen Nachrichtenhandler zu generieren. Sie können Programme zur Headerverarbeitung verwenden, um alle SOAP-Header zu verarbeiten, die der SOAP-Nachricht zugeordnet sind. Axis2 parst auch die SOAP-Antwortnachricht.

Web-Service-Anwendungen und Java

Für SOAP-Pipelines im Providermodus werden Anforderungs- und Antwortnachrichten zwischen dem Terminal-Handler der Pipeline und der Web-Service-Anwendung unter Verwendung eines Anwendungshandlers übergeben. Der Anwendungshandler verarbeitet den Hauptteil einer SOAP-Anforderung, sodass die Anforderung von der Anwendung verwendet werden kann. Der Anwendungshandler generiert auch unter Verwendung der zurückgegebenen Daten aus der Anwendung eine Antwort. Wenn der Terminal-Handler Ihrer Pipeline ein Java-basierter Nachrichtenhandler ist, können Sie den bereitgestellten Axis2-Anwendungshandler in der Pipelinekonfigurationsdatei angeben anstelle des bereitgestellten DFHPITP-Anwendungshandlers. Die Anwendungshandlerverarbeitung kann dann an zAAP ausgelagert werden. Weitere Informationen zu Anwendungshandlern finden Sie unter „Anwendungshandler“ auf Seite 106.

Für SOAP-Pipelines im Anforderungsmodus ruft die Web-Service-Anwendung die Pipeline mit dem Befehl **EXEC CICS INVOKE SERVICE** auf. Die Anforderungs- und Antwortnachrichten werden dann zwischen der Web-Service-Anwendung und dem ursprünglichen Handler in der Pipeline übergeben. Wenn Sie einen Java-basierten Handler als ursprünglichen Handler in der Pipeline angeben, wird der Befehl **EXEC CICS INVOKE SERVICE** von Axis2 verarbeitet, wodurch dieser Prozess an zAAP ausgelagert werden kann. Wenn der erste Handler kein Java-basierter Handler ist, wird der Befehl **EXEC CICS INVOKE SERVICE** von CICS verarbeitet.

Axis2-Verarbeitung auf einem JVM-Server

Axis2 erfordert einen JVM-Server, der durch eine JVMSERVER-Ressource in CICS dargestellt wird. Der JVM-Server ist eine Laufzeitumgebung, die mehrere gleichzeitige Anforderungen verschiedener Java-Programme in einer einzigen JVM verarbeiten kann. Der Klassenpfad für den JVM-Server muss die Java-Archivdateien von Axis2 enthalten. Sie können automatisch alle erforderlichen JAR-Dateien zu Klassenpfad hinzufügen, indem Sie die Option `JAVA_PIPELINE` im JVM-Profil angeben. Die Pipelinekonfigurationsdatei muss auch auf die JVMSERVER-Ressource verweisen, die für die Unterstützung von Axis2 konfiguriert ist.

Weitere Informationen zu JVM-Servern finden Sie unter Java support in CICS.

Axis2-Header-Handler

Obwohl Sie vorhandene Programme zur Headerverarbeitung verwenden können, ist es effizienter, Axis2-Handler in Java zu schreiben, um die SOAP-Header zu verarbeiten. Diese Handler können auch auf dem JVM-Server ausgeführt werden und kommen daher für die Auslagerung infrage. Weitere Informationen zum Erstellen von Axis2-Handlern finden Sie unter Writing Your Own Axis2 Module.

Ein Header-Handlerprogramm kann Axis2-APIs verwenden, um die Axis2-Umgebung, SOAP-Nachrichten und einzelne Web-Services zu ändern oder mit ihnen zu interagieren. Verwenden Sie diese APIs nicht, um Axis2 anzupassen, da Sie Axis2 unbeabsichtigt auf eine Weise ändern könnten, die dazu führt, dass CICS die Engine nicht ordnungsgemäß ausführen kann. Axis2-Handler werden nur unterstützt,

wenn sie mit der Axis2-Umgebung in einer Weise interagieren, die kompatibel mit der Verwendung von Axis2 durch CICS ist.

Axis2-Repository

Axis2 verwendet ein Repository, in dem es alle Konfigurationsdateien, Services und Module speichert. CICS stellt ein Standardrepository im Verzeichnis *usshome/lib/pipeline/repository* unter z/OS UNIX bereit, wobei *usshome* der Wert des **USSHOME**-Systeminitialisierungsparameters ist.

Das Standardrepository enthält die Konfigurationsdatei *axis2.xml*, die in CICS zur Verwendung von Axis2 erforderlich ist. Diese Datei befindet sich im Unterverzeichnis */conf* im Repository. Wenn Sie Ihr eigenes Repository erstellen, müssen Sie diese Datei in Ihr Repository stellen, damit CICS mit Axis2 funktioniert.

Bearbeiten Sie die Datei *axis2.xml* nur, wenn Sie Handlerprogramme registrieren. Diese Datei wird als interner Teil von CICS verwaltet, deshalb müssen Sie keine weiteren Änderungen an ihr vornehmen, es sei denn, Sie werden vom IBM Support dazu aufgefordert.

Datenformatierung für Web-Services

Diverse CICS-Technologien können JSON- und XML-Daten generieren, die gleichermaßen spezifikationskonform sind, sich aber physisch unterscheiden. Sie können aufgrund der Reihenfolge, in der sie Prüfungen zur Datenvalidierung anwenden, auch auf unterschiedliche Weise Fehler berichten, die in einer Eingabenachricht gefunden werden.

CICS verwendet diverse Technologien zur automatischen Umsetzung von JSON- und XML-Daten. Dazu zählen z/OS Connect for CICS- (Java- und Nicht-Java-Varianten), Axis2- und PIPELINE-Ressourcen. Diese Technologien generieren JSON- und XML-Daten in einem externen Format, das in den jeweiligen Spezifikationen vorgegeben wird.

Es kann mehrere Methoden zur Darstellung von Daten geben, die gleichermaßen spezifikationskonform sind, sich aber physisch unterscheiden. Die CICS-Technologien generieren immer konforme Daten, die sich jedoch physisch unterscheiden können. Wenn Sie beispielsweise zwischen den Java- und Nicht-Java-Optionen von z/OS Connect for CICS für JSON wechseln, können Sie kleine Unterschiede in der generierten JSON feststellen.

Solche Unterschiede sind beispielsweise unterschiedliche Fehlernachrichten, die bei Fehlerbedingungen ausgegeben werden, Unterschiede in der Art, wie Leerzeichen eingefügt werden, alternative (aber äquivalente) Darstellungen numerischer Daten sowie Varianten von Escapezeichen für Sonderzeichen. Diese Änderungen können auch das Resultat des Anwendens von Fehlerbehebungen in CICS sein oder im Zusammenhang mit neuen Releases von CICS stehen.

Partnersysteme wie ein JSON-Client sollten so geschrieben werden, dass sie spezifikationskonforme Abweichungen tolerieren. Partner setzen oft weit verbreitete, branchenübergreifende Bibliotheken und Technologien für die Interaktion mit JSON und XML ein. Solche Bibliotheken verarbeiten kleine Unterschiede in der Formatierung automatisch. Es ist jedoch möglich, dass weniger konforme Partnersysteme Abweichungen bei der Formatierung in den verschiedenen CICS-Technologien auf unterschiedliche Weise erkennen und darauf reagieren. Gehen Sie deshalb mit Bedacht vor, wenn Sie Anwendungen schreiben, die direkt mit JSON oder XML arbeiten, ohne die Vorteile eines standardbasierten Parsers zu nutzen.

CICS als Service-Provider für JSON-Anforderungen

Damit CICS einen Service für einen externen JSON-Client bereitstellt, muss es die Anforderung empfangen und sie über eine Pipeline an das Zielanwendungsprogramm übergeben. Die Antwort der Anwendung wird über dieselbe Pipeline an den JSON-Client zurückgegeben.

Es gibt verschiedene Möglichkeiten, CICS als Service-Provider für JSON-Anforderungen zu konfigurieren:

- Verwendung von z/OS Connect for CICS.

z/OS Connect for CICS ist eine Technologie für den Zugriff auf z/OS-Assets wie CICS-Programme unter Verwendung von JSON. Weitere Informationen zu z/OS Connect for z/OS und einige der Unterschiede zwischen z/OS Connect for CICS- und CICS-Java-Pipelines finden Sie unter *Getting started with z/OS Connect*.

- Verwendung von CICS-Java-Pipelines.

CICS-Java-Pipelines sind die Technologie, die in früheren Versionen von CICS für den Zugriff auf CICS-Programme unter Verwendung von JSON bereitgestellt wurde. Weitere Informationen zu CICS-Java-Pipelines finden Sie unter *Java-based SOAP pipelines*.

- Direkte Verwendung der JAX-RS- und JSON-Funktionen des CICS Liberty-JVM-Servers.
- Verwendung des von CICS bereitgestellten Terminal-Handlers DFHPIJT.

Sie können eine Provider-Pipeline mit dem Terminal-Handler DFHPIJT konfigurieren. Auf diese Weise kann die Pipeline JSON-Anforderungen verarbeiten, ohne dass ein JVM-Server installiert werden muss. Es gelten die folgenden Einschränkungen:

Restriction:

- REST-konforme JSON-Web-Services werden nicht unterstützt.
- Ein Kontextwechsel in der Pipeline wird nicht unterstützt.
- Es ist nicht möglich, SOAP- und JSON-Web-Services in einer JSON-Pipeline zu verwenden. DFHPIJT verarbeitet nur JSON-Nachrichten. Der Empfang einer SOAP-Nachricht führt zu einer Fehlerantwort.

CICS empfängt JSON-Daten und setzt sie in strukturierte Anwendungsdaten um, die von dem CICS-Anwendungsprogramm verstanden werden. Die Antworten der CICS-Anwendung werden in JSON-Nutzdaten für die ausgehende Antwort umgesetzt. Für die Umsetzungen müssen die Nachrichten geparkt werden. Wenn Sie z/OS Connect for CICS verwenden, haben Sie die Möglichkeit, einen Java-basierten JSON-Parser oder einen nicht-Java-basiertes Äquivalent zu verwenden. Die Konfigurationsoption wird in der z/OS Connect for CICS-Pipelinekonfigurationsdatei angegeben. Weitere Informationen zum Konfigurieren von z/OS Connect for CICS finden Sie unter *Configuring z/OS Connect for CICS*. Wenn Sie CICS-Java-Pipelines verwenden, wird das Parsing nur mit Java im JVM-Server durchgeführt. Die Auswirkungen der verschiedenen Konfigurationsentscheidungen werden im Folgenden beschrieben:

- Wenn Sie JSON mit Java parsen, kommt die Verarbeitung für das Auslagern zu zEnterprise Application Assist Processor (zAAP) infrage, falls dies verfügbar ist. Das Auslagern der Verarbeitung kann Kostenvorteile mit sich bringen.
- Wenn Sie den Nicht-Java-Parser von z/OS Connect for CICS verwenden, kann sich dies positiv auf die Leistung und den Durchsatz einiger Workloads auswirken. Weitere Informationen dazu, für welche Workloads diese Vorteile realisiert werden können, finden Sie in der Dokumentation zur Leistung, die mit diesem

Release verfügbar gemacht wird. Selbst wenn Sie den Nicht-Java-Parser verwenden, kommt für den Großteil der z/OS Connect for CICS-Infrastrukturverarbeitung die Auslagerung nach zAAP infrage.

- Wenn Sie den von CICS bereitgestellten Terminal-Handler DFHPIJT verwenden, kann sich dies positiv auf die Leistung und den Durchsatz einiger Workloads auswirken. Wenn Sie diese Methode zur Verarbeitung von JSON-Anforderungen verwenden, kommt für keinen Teil der Verarbeitung ein Auslagern nach zAAP infrage.

CICS als Service-Provider für JSON-Anforderungen unter Verwendung von z/OS Connect for CICS:

Damit CICS einen Service für einen externen JSON-Client bereitstellen kann, muss es die Anforderung in z/OS Connect for CICS empfangen, die JSON-Nachricht umsetzen und sie an das Zielanwendungsprogramm übergeben. Die Antwort der Anwendung wird über denselben Mechanismus an den JSON-Client zurückgegeben.

Abb. 17 zeigt eine Beispielkonfiguration der Architektur und Ressourcen, die erforderlich sind, um eine Anforderung von einem externen JSON-Client zu verarbeiten, wenn CICS ein Service-Provider ist, der z/OS Connect for CICS verwendet.

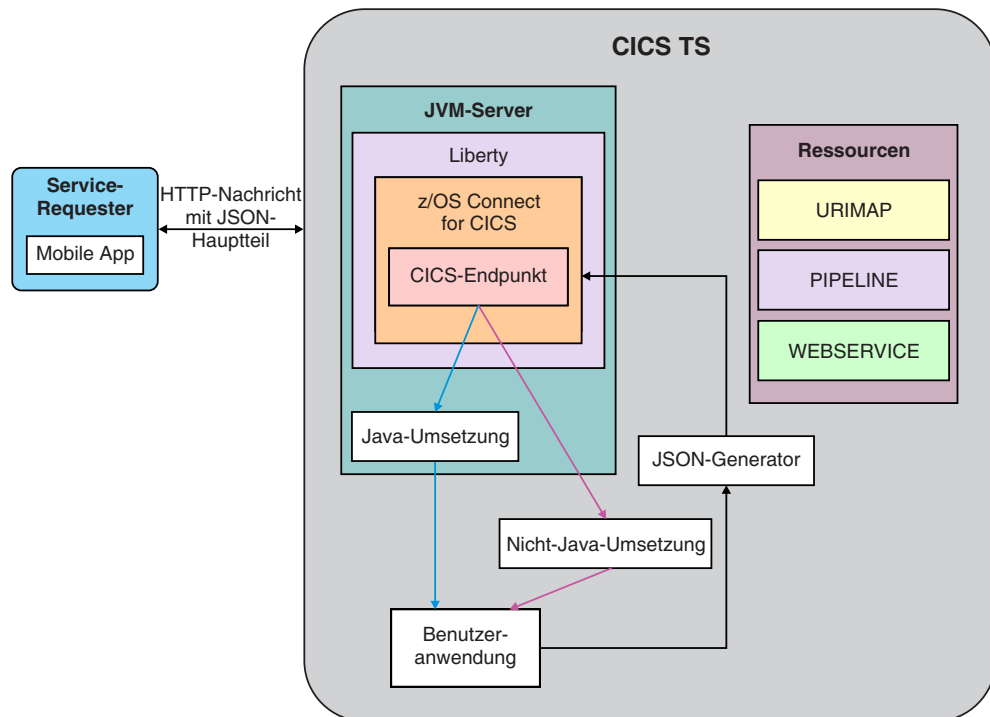


Abbildung 17. Architektur und Ressourcen für einen JSON-Service-Provider, der z/OS Connect for CICS verwendet

JSON-Anforderung mit z/OS Connect for CICS verarbeiten

Um eine Anforderung zu verarbeiten, führt CICS die folgenden Operationen aus:

1. Empfangen der Anforderung vom Service-Requester.
WebSphere Liberty empfängt die Anforderung und übergibt sie an z/OS Connect for CICS.
2. CICS löst die URIMAP-Ressource für die Anforderung auf, indem die URIMAP-Definitionen gescannt werden, deren Attribut USAGE auf JVMSERVER gesetzt ist. URIMAP gibt die verwendete WEBSERVICE-Ressource an. Es wird eine neue CICS-Task gestartet, um die Anforderung mithilfe der Transaktions-ID aus der URIMAP zu verarbeiten. Standardmäßig wird die Transaktion CPIH verwendet.

Die WEBSERVICE-Ressource steuert die Verarbeitung, die CICS ausführt. Insbesondere wird die WSBIND-Datei, auf die von der WEBSERVICE-Ressource verwiesen wird, für Datenumsetzungen zwischen JSON und strukturierten Anwendungsdaten verwendet. WSBIND-Dateien für JSON-Web-Services werden mithilfe der Dienstprogramme DFHLS2JS und DFHJS2LS generiert.

Anmerkung: Laufzeitvalidierung von JSON-Daten gegen ein Schema wird nicht unterstützt. Der Wert des Attributs VALIDATION einer WEBSERVICE-Ressource, die mit JSON-Nutzdaten verwendet wird, wird ignoriert. Informationen zu geltenden Einschränkungen finden Sie unter JSON web service restrictions.

3. z/OS Connect for CICS verarbeitet die Anforderung konfigurationsentsprechend. Wenn z/OS Connect for CICS für die Verwendung von globalen Interceptors konfiguriert ist, werden die Interceptors während dieser Verarbeitung ausgeführt.
4. Der CICS-Endpunkt erhält die Steuerung. Die JSON-Nutzdaten werden in strukturierte Anwendungsdaten umgesetzt, entsprechend der Konfigurationsoption, die in der Pipelinekonfigurationsdatei angegeben ist. Es gibt zwei Optionen für die Durchführung der Umsetzung:
 - Eine Java-Umsetzung wird im JVM-Server ausgeführt. Diese Verarbeitung kommt für eine Auslagerung auf einen zAAP-Prozessor infrage, wenn einer verfügbar ist.
 - Eine Nicht-Java-Umsetzung wird außerhalb des JVM-Servers ausgeführt und kann Leistungs- und Durchsatzvorteile für bestimmte Workloads mit sich bringen. Weitere Informationen dazu, für welche Workloads diese Vorteile realisiert werden können, finden Sie in der Dokumentation zur Leistung, die mit diesem Release verfügbar gemacht wird.

Die Zuordnung erfolgt entsprechend den Informationen in der WSBIND-Datei. Die Ausgabe der Umsetzung ist in beiden Fällen äquivalent.

5. CICS stellt eine Verknüpfung mit dem Anwendungsprogramm her und übergibt die umgesetzten Daten. Das Programm verarbeitet diese Eingabe und gibt eine Antwort an den JSON-Generator zurück.
6. Der JSON-Generator generiert eine JSON-Antwortnachricht unter Verwendung der Ausgabe aus Schritt 5. Diese Nachricht wird an den Service-Requester via z/OS Connect for CICS zurückgesendet.

z/OS Connect for CICS stellt auch eine REST-konforme Schnittstelle bereit, die die REST-konformen Standardmethoden unterstützt:

POST
PUT

GET
DELETE
HEAD

Bei Bedarf verwendet diese Schnittstelle die Umsetzung und den Generator, für die die Pipeline konfiguriert ist, auf dieselbe Weise wie reguläre JSON-Anforderungen. Weitere Informationen zu REST-konformen JSON-Web-Services finden Sie unter Concepts of RESTful JSON web services.

Ein Großteil der z/OS Connect for CICS-Infrastrukturverarbeitung kommt für die Auslagerung nach zEnterprise Application Assist Processor (zAAP) infrage.

CICS als Service-Provider für JSON-Anforderungen unter Verwendung von CICS-Java-Pipelines:

Damit CICS einen Service für einen externen JSON-Client bereitstellt, muss es die Anforderung empfangen und sie über eine Pipeline an das Zielanwendungsprogramm übergeben. Die Antwort der Anwendung wird über dieselbe Pipeline an den JSON-Client zurückgegeben. Die JSON-Umsetzung wird mithilfe von Java im JVM-Server durchgeführt.

Abb. 18 zeigt eine Beispielkonfiguration der Architektur und Ressourcen, die erforderlich sind, um eine Anforderung von einem externen JSON-Client zu verarbeiten, wenn CICS ein Service-Provider ist, der eine Java-Pipeline verarbeitet. Die Pipelineverarbeitung für eine JSON-Anforderung entspricht der Art, wie CICS eine SOAP-Anforderung in einer Java-Pipeline verarbeitet. Weitere Informationen finden Sie unter Java-based SOAP pipelines.

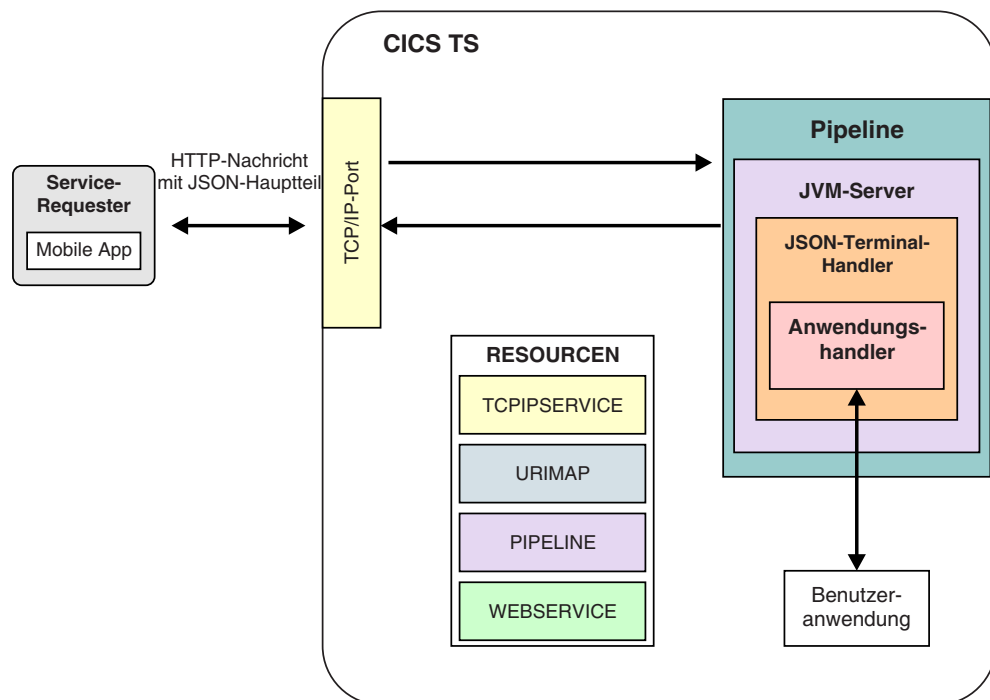


Abbildung 18. Architektur und Ressourcen für einen JSON-Service-Provider, der eine CICS Java-Pipeline verwendet

Verarbeitung einer JSON-Anforderung

Um eine Anforderung zu verarbeiten, führt CICS die folgenden Operationen aus:

1. Empfangen der Anforderung vom Service-Requester.
Die TCPIPService-Resource gibt einen Port für eingehende Anforderungen an. Dieser Port wird von der von CICS bereitgestellten Socket-Listener-Task überwacht.
2. Überprüfen der Anforderung und Extrahieren der Inhalte, die für das Zielanwendungsprogramm relevant sind.
Wenn die Anforderungsnachricht am entsprechenden Port empfangen wird, werden die URIMAP-Ressourcendefinitionen nach einer URIMAP-Ressource durchsucht, deren USAGE-Attribut auf PIPELINE und deren PATH-Attribut auf den in der Anforderung gesendeten URI gesetzt ist. Wenn eine entsprechende URIMAP-Definition gefunden wird, werden die PIPELINE- und WEBSERVICE-Definitionen aus den PIPELINE- und WEBSERVICE-Attributen der URIMAP-Definition verwendet. Das Attribut TRANSACTION der URIMAP-Definition bestimmt den Namen der Transaktion, die zur Verarbeitung der Pipeline angehängt werden soll. Standardmäßig ist das die Transaktion CPIH. Die URIMAP-Definition gibt auch die zu verwendenden PIPELINE- und WEBSERVICE-Ressourcen an.
Diese PIPELINE- und WEBSERVICE-Ressourcen steuern die Verarbeitung, die CICS ausführt. Insbesondere wird die WSBIND-Datei, auf die von der WEBSERVICE-Ressource verwiesen wird, für Datenumsetzungen zwischen JSON und Sprachstrukturen verwendet. WSBIND-Dateien für JSON-Web-Services werden mithilfe der Dienstprogramme DFHLS2JS und DFHJS2LS generiert.
Anmerkung: Laufzeitvalidierung von JSON-Daten gegen ein Schema wird nicht unterstützt. Der Wert des Attributs VALIDATION einer WEBSERVICE-Ressource, die mit JSON-Nutzdaten verwendet wird, wird ignoriert. Informationen zu geltenden Einschränkungen finden Sie unter JSON web service restrictions.
3. Die Pipelineverarbeitung startet und die Anforderung durchläuft alle definierten Handler. Es wird nicht erwartet, dass die Handler, die derzeit von CICS für SOAP-Web-Services implementiert werden, für JSON-Web-Services relevant sind.
4. Am Ende der Pipeline wird der JSON-Terminal-Handler aufgerufen. Dieser Terminal-Handler ist ein Java-Programm, das eine Schnittstelle mit der Axis2-Pipeline bildet. Der Terminal-Handler führt die erforderliche Axis2-Konfiguration aus und startet anschließend die Axis2-Engine mit dem HTTP-Anforderungshauptteil. In der Axis2-Pipeline wird der JSON-Hauptteil (falls vorhanden) geparkt und ein Java-Objektmodell, das den Inhalt darstellt, wird erstellt. CICS ruft dann den Anwendungshandler auf. Die Hauptrolle des Anwendungshandlers besteht darin, die Java-Objektmodelldarstellung der Anforderung Anwendungsdaten zuzuordnen. Diese Zuordnung wird mithilfe der Beschreibung der Sprachstruktur in der WSBIND-Datei durchgeführt.
5. Aufrufen des Anwendungsprogramms, wobei aus der Anforderung extrahierte Daten übergeben werden.
Dann stellt der Anwendungshandler eine Verknüpfung zum Anwendungsprogramm her. Das Programm verarbeitet diese Eingabe und gibt eine Antwort an den Anwendungshandler zurück.
6. Erstellen einer Antwort unter Verwendung der Daten, die vom Anwendungsprogramm zurückgegeben werden, und senden Sie sie an den Service-Requester.

Der Anwendungshandler und die Nachrichtenhandler konvertieren die Antwortnachricht, die von der Service-Provider-Anwendung empfangen wurde, in eine Nachricht im Format der ursprünglichen Anforderung. Diese Nachricht wird an den Service-Requester zurückgesendet.

Ein Teil der Verarbeitung in der Pipeline kommt für die Auslagerung in zEnterprise Application Assist Processor (zAAP) infrage.

CICS-Infrastruktur für SOAP-Service-Provider erstellen

Zur Erstellung der CICS-Infrastruktur für einen SOAP-Service-Provider müssen Sie eine Pipelinekonfigurationsdatei sowie eine Reihe von CICS-Ressourcen erstellen.

Before you begin

Wenn Sie eine Java-Pipeline verwenden möchten, stellen Sie sicher, dass eine JVM-SERVER-Ressource vorhanden ist, für die die Option `JAVA_PIPELINE=YES` im JVM-Profil angegeben ist.

Ein JVM-Server kann die SOAP-Verarbeitung für viele Java-Pipelines steuern.

About this task

Sie können die PIPELINE-Ressource in einer lokalen CICS-Region mithilfe von CICS- oder CICSplex SM-Funktionen definieren oder Sie können CICS Explorer verwenden, um die PIPELINE-Ressource entweder in einer lokalen CICS-Region oder in einem CICS-Bundle zu definieren. Wenn Sie CICS Explorer verwenden, um eine PIPELINE-Ressource in einem CICS-Bundle zu definieren, erstellen Sie auch die Pipelinekonfigurationsdatei und packen sie in das CICS-Bundle, sodass Sie diese Datei nicht separat verwalten müssen. PROGRAM-Ressourcen und WEBSERVICE-Ressourcen können auch in CICS-Bundles definiert werden. Wenn Sie eine WEBSERVICE-Ressource in einem CICS-Bundle definieren, können Sie eine Web-Service-Bindungsdatei und ein WSDL-Dokument oder eine WSDL-Archivdatei importieren und diese in das Bundle einschließen. Sie können auch URIMAP-Definitionen zur Unterstützung des Web-Service erstellen und diese in ein Bundle packen. Weitere Hilfe zur Verwendung von CICS Explorer zum Erstellen und Bearbeiten von Ressourcen in CICS-Bundles finden Sie unter *Working with bundles in the CICS Explorer product documentation*.

Procedure

1. Definieren Sie die Transportinfrastruktur.
 - a. Wenn Sie den WebSphere MQ-Transport verwenden, müssen Sie eine oder mehrere lokale Warteschlangen definieren, die Eingabenachrichten bis zu ihrer Verarbeitung speichern. Und Sie müssen einen Auslöseprozess definieren, der die CICS-Transaktion angibt, die die Eingabenachrichten verarbeitet wird.
 - 1) Ausführliche Informationen finden Sie unter *Configuring CICS to use the WebSphere MQ transport*.
 - b. Wenn Sie den HTTP-Transport verwenden, müssen Sie eine TCPIPSERVICE-Ressource definieren, die den Port angibt, an dem eingehende Anforderungen empfangen werden.
 - 1) Ausführliche Informationen finden Sie unter *CICS resources for web services*.
2. Optional: Wiederholen Sie diesen Schritt für jede einzelne erforderliche Transportkonfiguration.

3. Definieren Sie die Nachrichtenhandler und Programme zur Headerverarbeitung, die Sie in die Pipelinekonfigurationsdatei einschließen möchten, um eingehende Web-Service-Anforderungen und ihre Antworten zu verarbeiten. CICS stellt die folgenden Handler und Programme zur Headerverarbeitung bereit.
 - a. SOAP message handlers für die Verarbeitung von SOAP 1.1- oder 1.2-Nachrichten. Sie können nur eine Ebene von SOAP in einer Service-Provider-Pipeline verarbeiten.
 - b. MTOM handler zur Verarbeitung von MIME-Multipart/Related-Nachrichten, die den MTOM/XOP-Spezifikationen entsprechen.
 - c. Support for securing web services zur Verarbeitung von sicheren Web-Service-Nachrichten.
 - d. Support for Web Services Transactions zur Verarbeitung von atomaren Transaktionsnachrichten.
4. Optional: Wenn Sie Ihre eigene Verarbeitung in der Pipeline durchführen möchten, müssen Sie einen Nachrichtenhandler oder ein Programm zur Headerverarbeitung erstellen. Ausführliche Informationen finden Sie unter Message handlers. Wenn Sie sich entscheiden, benutzerdefinierte Nachrichtenhandlerprogramme zu erstellen, müssen Sie diese threadsicher machen, um die Leistung zu optimieren.
5. Erstellen Sie eine XML-Pipelinekonfigurationsdatei, die Ihre Nachrichtenhandler, Ihre Programme zur Headerverarbeitung und Ihre Anwendungshandler enthält.
 - a. CICS stellt außerdem zwei grundlegende Beispiele für Pipelinekonfigurationsdateien im Providermodus bereit, `basicsoap11provider.xml` und `basicsoap11javaprovider.xml`.
 - b. Sie können diese Beispiele nach Bedarf bearbeiten oder zusätzliche Nachrichtenhandler hinzufügen. Sie finden sie in der Bibliothek `/usr/lpp/cicsts/cicsts55/samples/pipelines` (wobei `/usr/lpp/cicsts/cicsts55` das Standardinstallationsverzeichnis für CICS-Dateien unter z/OS UNIX ist).
 - c. Weitere Informationen zu den verfügbaren Optionen in der Pipelinekonfigurationsdatei finden Sie unter Pipeline configuration files.
6. Kopieren Sie die Pipelinekonfigurationsdatei in ein geeignetes Verzeichnis unter z/OS UNIX.
7. Ändern Sie die Pipelinekonfigurationsdateiberechtigungen so, dass die CICS-Region die Datei lesen kann.
8. Wiederholen Sie die Schritte 5 bis 7 für jede erforderliche Pipelinekonfiguration.
9. Erstellen Sie eine PIPELINE-Ressource.
 - a. Die PIPELINE-Ressource definiert die Position der Pipelinekonfigurationsdatei. Sie gibt auch ein *Abholverzeichnis* an, bei dem es sich um das z/OS UNIX-Verzeichnis handelt, das die Web-Service-Bindungsdateien und optional die WSDL enthält.
 - b. Wiederholen Sie diesen Schritt für jede Pipelinekonfiguration.
 - a. Wenn Sie eine PIPELINE-Ressource erstellen, liest CICS alle Dateien im angegebenen Abholverzeichnis und erstellt die WEBSERVICE-Ressource und die URIMAP-Ressource dynamisch.
10. Falls Sie keine automatisch installierten PROGRAM-Definitionen verwenden, erstellen Sie eine PROGRAM-Ressource für jedes Programm, das in der Pipeline ausgeführt wird. Dazu gehört das Zielanwendungsprogramm, das normalerweise unter der Transaktion CPIH ausgeführt wird. Die Transaktion wird

mit dem Attribut TASKDATALOC(ANY) definiert. Wenn Sie eine Programmverbindung verwenden, müssen Sie daher die Option AMODE(31) angeben.

Results

Ihr CICS-System enthält jetzt die für die einzelnen Service-Provider erforderliche Infrastruktur.

What to do next

Sie können die Konfiguration bei Bedarf erweitern, entweder um zusätzliche Transportinfrastruktur zu definieren oder um zusätzliche Pipelines zu erstellen.

CICS-Infrastruktur für SOAP-Service-Requester erstellen

Zur Erstellung der CICS-Infrastruktur für einen SOAP-Service-Requester müssen Sie eine Pipelinekonfigurationsdatei sowie eine Reihe von CICS-Ressourcen erstellen.

Before you begin

Wenn Sie eine Java-Pipeline verwenden möchten, stellen Sie sicher, dass eine JVM-SERVER-Ressource vorhanden ist, für die die Option JAVA_PIPELINE=YES im JVM-Profil angegeben ist. Siehe JVMSERVER resources.

Ein JVM-Server kann die SOAP-Verarbeitung für viele Java-Pipelines steuern.

About this task

Sie können die PIPELINE-Ressource in einer lokalen CICS-Region mithilfe von CICS- oder CICSplex SM-Funktionen definieren oder Sie können CICS Explorer verwenden, um die PIPELINE-Ressource entweder in einer lokalen CICS-Region oder in einem CICS-Bundle zu definieren. Wenn Sie CICS Explorer verwenden, um eine PIPELINE-Ressource in einem CICS-Bundle zu definieren, erstellen Sie auch die Pipelinekonfigurationsdatei und packen sie in das CICS-Bundle, sodass Sie diese Datei nicht separat verwalten müssen. PROGRAM-, WEBSERVICE- und URIMAP-Ressourcen können auch in CICS-Bundles definiert werden. Wenn Sie eine WEBSERVICE-Ressource in einem CICS-Bundle definieren, können Sie eine Web-Service-Bindungsdatei und ein WSDL-Dokument oder eine WSDL-Archivdatei importieren und diese in das Bundle packen. Und für einen Service-Provider können Sie eine PROGRAM-Definition hinzufügen. Sie können auch URIMAP-Definitionen zur Unterstützung des Web-Service erstellen und diese in ein Bundle packen. Weitere Hilfe zur Verwendung von CICS Explorer zum Erstellen und Bearbeiten von Ressourcen in CICS-Bundles finden Sie unter Working with bundles in the CICS Explorer product documentation.

Procedure

1. Definieren Sie die Nachrichtenhandler und Programme zur Headerverarbeitung, die Sie in die Pipelinekonfigurationsdatei einschließen möchten, um eingehende Web-Service-Anforderungen und ihre Antworten zu verarbeiten. CICS stellt die folgenden Handler und Programme zur Headerverarbeitung bereit.
 - a. SOAP-Nachrichtenhandler zur Verarbeitung von SOAP 1.1- oder 1.2-Nachrichten. Sie können nur eine Version von SOAP in einer Service-Requester-Pipeline unterstützen.
 - b. MTOM-Handler zur Verarbeitung von MIME-Multipart/Related-Nachrichten, die den MTOM/XOP-Spezifikationen entsprechen.

- c. Sicherheitshandler zur Verarbeitung von sicheren Web-Service-Nachrichten.
 - d. WS-AT-Programm zur Headerverarbeitung zur Verarbeitung von atomaren Transaktionsnachrichten.
2. Optional: Wenn Sie Ihre eigene Verarbeitung in der Pipeline durchführen möchten, müssen Sie einen Nachrichtenhandler oder ein Programm zur Headerverarbeitung erstellen. Ausführliche Informationen finden Sie unter „Nachrichtenhandler“ auf Seite 151. Wenn Sie sich entscheiden, benutzerdefinierte Nachrichtenhandlerprogramme zu erstellen, müssen Sie diese thread-sicher machen, um die Leistung zu optimieren.
 3. Erstellen Sie eine XML-Pipelinekonfigurationsdatei, die Ihre Nachrichtenhandler und Programme zur Headerverarbeitung enthält. CICS stellt außerdem zwei grundlegende Beispiele für Pipelinekonfigurationsdateien im Requestermodus bereit, `basicsoap11provider.xml` und `basicsoap11javaprovider.xml`, die Sie nach Bedarf kopieren und bearbeiten können. Sie finden diese Beispiele in der Bibliothek `/usr/lpp/cicsts/cicsts55/samples/pipelines` (wobei `/usr/lpp/cicsts/cicsts55` das Standardinstallationsverzeichnis für CICS-Dateien unter z/OS UNIX ist). Weitere Informationen zu den verfügbaren Optionen in der Pipelinekonfigurationsdatei finden Sie unter „Pipelinekonfigurationsdateien“ auf Seite 95.
 4. Kopieren Sie die Pipelinekonfigurationsdatei in ein geeignetes Verzeichnis unter z/OS UNIX.
 5. Ändern Sie die Pipelinekonfigurationsdateiberechtigungen so, dass die CICS-Region die Datei lesen kann.
 6. Wiederholen Sie die Schritte 3 bis 5 für jede erforderliche Pipelinekonfiguration.
 7. Erstellen Sie eine PIPELINE-Ressource. Siehe PIPELINE resources. Die PIPELINE-Ressource definiert die Position der Pipelinekonfigurationsdatei. Sie gibt auch ein *Abholverzeichnis* an, bei dem es sich um das z/OS UNIX-Verzeichnis handelt, das die Web-Service-Bindungsdateien und optional die WSDL enthält. Sie können auch ein Zeitlimit in Sekunden angeben, das bestimmt, wie lange CICS auf eine Antwort von Web-Service-Providern wartet. Wiederholen Sie diesen Schritt für jede Pipelinekonfigurationsdatei. Wenn Sie eine PIPELINE-Ressource erstellen, liest CICS alle Dateien im angegebenen Abholverzeichnis und erstellt die WEBSERVICE-Ressourcen dynamisch (siehe WEBSERVICE resources).
 8. Falls Sie keine automatisch installierten PROGRAM-Definitionen verwenden, erstellen Sie eine PROGRAM-Ressource für jedes Programm, das in der Pipeline ausgeführt wird. Siehe PROGRAM resources. Dazu gehört das Service-Requester-Anwendungsprogramm, das normalerweise unter der Transaktion CPIH ausgeführt wird. Die Transaktion wird mit dem Attribut `TASKDATALOC(ANY)` definiert. Wenn Sie eine Programmverbindung verwenden, müssen Sie daher die Option `AMODE(31)` angeben.
 9. Optional: Erstellen Sie eine URIMAP-Ressource (siehe URIMAP resources) für Clientanforderungen an jeden URI, den Ihre Service-Requester verwenden, um Anforderungen zu erstellen, wie in den Anweisungen unter Creating a URIMAP resource for CICS as a HTTP client beschrieben. Sie können den URI direkt im Befehl **INVOKE SERVICE** in Ihren Programmen angeben, statt eine URIMAP-Ressource zu verwenden. Aber wenn Sie eine URIMAP-Ressource verwenden, müssen Sie Ihre Anwendungen nicht erneut kompilieren, wenn sich der URI eines Service-Providers ändert. Mit einer URIMAP-Ressource können Sie auch Verbindungspooling implementieren, wobei CICS die Clientverbindung nach der Verwendung geöffnet hält, sodass sie von der Anwendung für nachfolgende Anforderungen bzw. von einer anderen Anwendung, die denselben Service aufruft, wiederverwendet werden kann.

Results

Ihr CICS-System enthält jetzt die für die einzelnen Service-Requester erforderliche Infrastruktur.

What to do next

Sie können die Konfiguration bei Bedarf erweitern, wenn Sie zusätzliche Pipelines erstellen.

CICS-Infrastruktur für JSON-Service-Provider erstellen

Zur Erstellung der CICS-Infrastruktur für einen JSON-Service-Provider müssen Sie eine Pipelinekonfigurationsdatei sowie eine Reihe von CICS-Ressourcen erstellen.

Before you begin

Für die Implementierung von JSON-Services in CICS stehen eine Reihe verschiedener Technologien zur Verfügung. Die funktionsreichste Option ist z/OS Connect. Diese Task beschreibt die alternativen Optionen.

Wenn Sie z/OS Connect nicht verwenden möchten, nutzen Sie CICS als Service-Provider für JSON-Anforderungen oder verwenden Sie die verknüpfbare Schnittstelle für die Umsetzung von JSON. Definieren und installieren Sie zunächst eine JVMSERVER-Ressource mit einem JVM-Profil, das die Option **JAVA_PIPELINE=YES** angibt. Es wird eine JVMSERVER-Beispielressourcendefinition namens DFHAXIS in der Gruppe DFH\$AXIS bereitgestellt.

Anmerkung: Bei der hier beschriebenen Infrastruktur wird davon ausgegangen, dass Sie nicht z/OS Connect for CICS verwenden, um eine Verbindung mit Ihrem JSON-Service-Provider herzustellen, und entsprechend wird Java-Parsing innerhalb des JVM-Servers verwendet, um die JSON-Nachrichten zu parsen. Wenn Sie ein anderes JSON-Parsing als das von Java verwenden möchten, müssen Sie z/OS Connect for CICS für die Verbindung mit dem JSON-Web-Service verwenden. Weitere Informationen zum Einrichten von z/OS Connect for CICS finden Sie unter *Configuring z/OS Connect for CICS*.

Procedure

1. Definieren Sie die Transportinfrastruktur.
Definieren Sie eine TCPIPSERVICE-Ressource, die den Port definiert, an dem eingehende Anforderungen empfangen werden. Ausführliche Informationen finden Sie unter CICS-Ressourcen für Web-Services.
2. Definieren Sie die Nachrichtenhandler, die Sie in die Pipelinekonfigurationsdatei einschließen möchten, um eingehende Web-Service-Anforderungen und ihre Antworten zu verarbeiten.
Wenn Sie Ihre eigene Verarbeitung in der Pipeline durchführen möchten, müssen Sie einen Nachrichtenhandler erstellen. Ausführliche Informationen finden Sie unter Nachrichtenhandler. Wenn Sie sich entscheiden, benutzerdefinierte Nachrichtenhandlerprogramme zu erstellen, müssen Sie diese threadsicher machen, um die Leistung zu optimieren.
3. Erstellen Sie eine XML-Pipelinekonfigurationsdatei, die Ihre Nachrichtenhandler, Ihre Programme zur Headerverarbeitung und Ihre Anwendungshandler enthält. CICS stellt ein grundlegendes Beispiel für eine Pipelinekonfigurationsdatei im Providermodus bereit: `jsonjavaprovider.xml`. Sie können dieses Beispiel nach Bedarf bearbeiten, um zusätzliche Nachrichtenhandler hinzuzufügen.

Sie finden dieses Beispiel im Verzeichnis `/usr/lpp/cicsts/cicsts55/samples/pipelines`, wobei `/usr/lpp/cicsts/cicsts55` das Standardinstallationsverzeichnis für CICS-Dateien unter z/OS UNIX ist. Weitere Informationen zu den verfügbaren Optionen in der Pipelinekonfigurationsdatei finden Sie unter „In Service-Provider- und Service-Requester-Pipelines verwendete Elemente“ auf Seite 113.

4. Kopieren Sie die Pipelinekonfigurationsdatei in ein geeignetes Verzeichnis unter z/OS UNIX.
5. Ändern Sie die Pipelinekonfigurationsdateiberechtigungen so, dass die CICS-Region die Datei lesen kann.
6. Erstellen Sie eine PIPELINE-Ressource. Die PIPELINE-Ressource definiert die Position der Pipelinekonfigurationsdatei. Sie gibt auch ein *Abholverzeichnis* an, bei dem es sich um das z/OS UNIX-Verzeichnis handelt, das die Web-Service-Bindungsdateien enthält. Wiederholen Sie diesen Schritt für jede Pipelinekonfiguration. Wenn Sie eine PIPELINE-Ressource erstellen oder einen Pipeline-Scan (PIPELINE SCAN) ausführen, liest CICS die `.wsbind`-Dateien im angegebenen Abholverzeichnis und erstellt dynamisch passende WEBSERVICE- und URI-MAP-Ressourcen.
7. Falls Sie keine automatisch installierten PROGRAM-Definitionen verwenden, erstellen Sie eine PROGRAM-Ressource für jedes Programm, das in der Pipeline ausgeführt wird. Dazu gehört das Zielanwendungsprogramm, das normalerweise unter der Transaktion CPIH ausgeführt wird. Die Transaktion wird mit dem Attribut `TASKDATALOC(ANY)` definiert. Wenn Sie eine Programmverbindung verwenden, müssen Sie daher die Option `AMODE(31)` angeben.

Results

Sie haben die für die einzelnen Service-Provider erforderliche Infrastruktur erstellt und können jetzt diese Ressourcen in Ihrem CICS-System installieren.

What to do next

Installieren Sie die Ressourcen. Sie können die Konfiguration bei Bedarf erweitern, entweder um zusätzliche Transportinfrastruktur zu definieren oder um zusätzliche Pipelines zu erstellen.

CICS-Infrastruktur für einen Nicht-Java-JSON-Service-Provider erstellen

Sie können eine Nicht-Java-Umgebung für die Verarbeitung von JSON-Anforderungen einrichten. Zur Erstellung der CICS-Infrastruktur für einen Nicht-Java-JSON-Service-Provider müssen Sie eine Pipelinekonfigurationsdatei sowie eine Reihe von CICS-Ressourcen erstellen.

Procedure

1. Definieren Sie die Transportinfrastruktur.

Definieren Sie eine TCPIPSERVICE-Ressource, die den Port definiert, an dem eingehende Anforderungen empfangen werden. Ausführliche Informationen finden Sie unter CICS-Ressourcen für Web-Services.

2. Definieren Sie die Nachrichtenhandler, die Sie in die Pipelinekonfigurationsdatei einschließen möchten, um eingehende Web-Service-Anforderungen und ihre Antworten zu verarbeiten.

Wenn Sie Ihre eigene Verarbeitung in der Pipeline durchführen möchten, müssen Sie einen Nachrichtenhandler erstellen. Ausführliche Informationen finden

Sie unter Nachrichtenhandler. Wenn Sie sich entscheiden, benutzerdefinierte Nachrichtenhandlerprogramme zu erstellen, müssen Sie diese threadsicher machen, um die Leistung zu optimieren.

3. Erstellen Sie eine XML-Pipelinekonfigurationsdatei, die Ihre Nachrichtenhandler enthält.

In der Konfigurationsdatei müssen Sie das Terminal-Handlerprogramm DFHPIJT in einem `<terminal_handler>`-Element angeben, wie unter Abb. 19 gezeigt. DFHPIJT ist das von CICS bereitgestellte JSON-Handlerprogramm, das die Nicht-Java-Verarbeitung von JSON-Nachrichten aktiviert.

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

Abbildung 19. Terminal-Handler DFHPIJT für Nicht-Java-Verarbeitung von JSON-Nachrichten

Anmerkung: Wenn Sie DFHPIJT als Terminal-Handler verwenden, definieren Sie keinen Anwendungshandler in der Pipelinekonfigurationsdatei, d. h. die Pipelinekonfigurationsdatei sollte kein `<apphandler>`-Element enthalten. Wenn ein Anwendungshandler angegeben ist, wird er nicht aufgerufen.

Weitere Informationen zu den verfügbaren Optionen in der Pipelinekonfigurationsdatei finden Sie unter „In Service-Provider- und Service-Requester-Pipelines verwendete Elemente“ auf Seite 113.

4. Kopieren Sie die Pipelinekonfigurationsdatei in ein geeignetes Verzeichnis unter z/OS UNIX.
5. Ändern Sie die Pipelinekonfigurationsdateiberechtigungen so, dass die CICS-Region die Datei lesen kann.
6. Erstellen Sie eine PIPELINE-Ressource. Die PIPELINE-Ressource definiert die Position der Pipelinekonfigurationsdatei. Sie gibt auch ein *Abholverzeichnis* an, bei dem es sich um das z/OS UNIX-Verzeichnis handelt, das die Web-Service-Bindungsdateien enthält. Wiederholen Sie diesen Schritt für jede Pipelinekonfiguration. Wenn Sie eine PIPELINE-Ressource erstellen oder einen Pipeline-Scan (PIPELINE SCAN) ausführen, liest CICS die `.wsbind`-Dateien im angegebenen Abholverzeichnis und erstellt dynamisch passende WEBSERVICE- und URI-MAP-Ressourcen.
7. Falls Sie keine automatisch installierten PROGRAM-Definitionen verwenden, erstellen Sie eine PROGRAM-Ressource für jedes Programm, das in der Pipeline ausgeführt wird. Dazu gehört das Zielanwendungsprogramm, das normalerweise unter der Transaktion CPIH ausgeführt wird. Die Transaktion wird mit dem Attribut `TASKDATALOC(ANY)` definiert. Wenn Sie eine Programmverbindung verwenden, müssen Sie daher die Option `AMODE(31)` angeben.

Results

Sie haben die für die einzelnen Service-Provider erforderliche Infrastruktur erstellt und können jetzt diese Ressourcen in Ihrem CICS-System installieren.

What to do next

Installieren Sie die Ressourcen. Sie können die Konfiguration bei Bedarf erweitern, entweder um zusätzliche Transportinfrastruktur zu definieren oder um zusätzliche Pipelines zu erstellen.

z/OS Connect for CICS konfigurieren

z/OS Connect in CICS konfigurieren

z/OS Connect ist die primäre Technologie von IBM für die Implementierung von JSON-Services und APIs für CICS.

z/OS Connect kann in einer eigenständigen Konfiguration konfiguriert oder auf einem Liberty-JVM-Server in CICS gehostet werden. Informationen zu der eigenständigen Konfiguration finden Sie unter z/OS Connect Enterprise Edition V3.0 product documentation.

Diese Informationen beschreiben, wie z/OS Connect in einem CICS-Adressraum konfiguriert wird.

Es gibt zwei Hauptversionen von z/OS Connect: z/OS Connect for CICS 1.0 und das weiter entwickelte z/OS Connect Enterprise Edition.

Wählen Sie die Task aus, die Ihren Anforderungen entspricht.

z/OS Connect for CICS 1.0 konfigurieren:

z/OS Connect for CICS 1.0 wird als Teil von CICS Transaction Server verteilt. Sie müssen einen JVM-Server konfigurieren sowie die Pipelinekonfiguration und Ressourcen für z/OS Connect einrichten, um JSON-Services implementieren zu können. Diese Erstkonfiguration ist ein einmaliger Vorgang.

Before you begin

Verfügen Sie bereits über einen WebSphere Liberty-JVM-Server, der in CICS konfiguriert ist? Es ist zwar möglich, z/OS Connect und andere nicht zugehörige Services in derselben WebSphere Liberty-Umgebung zu hosten, trotzdem wird empfohlen, einen separaten JVM-Server nur für z/OS Connect zu konfigurieren.

Sie können z/OS Connect for CICS 1.0 in einer eigenen CICS-Region oder in einer Gruppe von CICS-Regionen hosten und den Distributed Program Link-Mechanismus (Verbindung zu verteilten Programmen) verwenden, um CICS-Programme in den CICS-Regionen mit eigenen Anwendungen zu hosten.

Procedure

1. Erstellen Sie einen JVMSERVER und konfigurieren Sie ihn für die Unterstützung von WebSphere Liberty. Weitere Informationen zum Erstellen eines WebSphere Liberty-JVMSERVER finden Sie unter Configuring a Liberty JVM server.
2. Konfigurieren Sie WebSphere Liberty für Ihre Sicherheitsanforderungen. Standardmäßig erwartet WebSphere Liberty, dass clientzertifizierte SSL-Zertifikate verwendet werden. Um HTTP-Basisauthentifizierung zu aktivieren, fügen Sie die folgende Konfigurationsoption zur Datei server.xml hinzu:

```
<!-- Failover zu HTTP-Basisauthentifizierung zulassen -->  
<webAppSecurity allowFailOverToBasicAuth="true"/>
```

Sie müssen außerdem Benutzern von z/OS Connect die Sicherheitsrolle **zosConnectAccess** erteilen. Weitere Informationen zur WebSphere Liberty-Sicherheit finden Sie unter Configuring security for a Liberty JVM server. Weitere Informationen zur z/OS Connect-Sicherheit finden Sie unter Sicherheit für z/OS Connect.

3. Aktualisieren Sie die Liste `<featureManager>` in der Datei `server.xml` für die WebSphere Liberty-Umgebung, sodass sie die Funktion `<feature>cicsts:zosConnect-1.0</feature>` enthält, wie im folgenden Beispiel gezeigt:

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

4. Definieren Sie den z/OS Connect for CICS 1.0-Service-Controller, indem Sie die folgende Anweisung zur Datei `server.xml` hinzufügen:

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

5. Installieren Sie den JVMSERVER. Prüfen Sie die generierte Datei `messages.log` auf Fehler oder Warnnachrichten. Dieses Protokoll enthält die Nachrichten, die von WebSphere Liberty Server generiert werden, einschließlich Nachrichten, die von z/OS Connect for CICS 1.0 zurückgegeben werden, z. B. die folgenden:

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

6. Erstellen Sie eine XML-Pipelinekonfigurationsdatei. Sie finden ein Beispiel für die Pipelinekonfigurationsdatei `jsonzosconnectprovider.xml` im Verzeichnis `/usr/lpp/cicsts/cicsts55/samples/pipelines/` (wobei `/usr/lpp/cicsts/cicsts55` das Standardinstallationsverzeichnis für CICS-Dateien unter z/OS UNIX ist). Sie müssen entscheiden, ob Sie die JSON mithilfe von Java auf dem Liberty-JVM-Server parsen möchten (dies ist der Standard) oder ob der Nicht-Java-JSON-Parser verwendet werden soll:

- Um die JSON mithilfe von Java auf dem Liberty-JVM-Server zu parsen, können Sie das Beispiel für die Pipelinekonfigurationsdatei verwenden, müssen aber DFHWLP im Element `<jvmserver>` durch den Namen Ihres JVMSERVER aus Schritt 1 ersetzen.
- Um die JSON mithilfe des Nicht-Java-Parsers zu parsen, ändern Sie die Beispielkonfigurationsdatei, indem Sie das Attribut `java_parser="no"` an das Element `<provider_pipeline_json>` anhängen, wie im folgenden Beispiel gezeigt:

```
<?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>
```

Ersetzen Sie DFHWLP durch den Namen des JVMSERVER, den Sie am Anfang dieser Prozedur erstellt haben.

7. Kopieren Sie die Pipelinekonfigurationsdatei in ein passendes Verzeichnis in zFS und stellen Sie sicher, dass die Dateiberechtigungen zulassen, dass die CICS-Region die Datei liest. Weitere Informationen finden Sie unter Pipelinekonfigurationsdateien.
8. Erstellen Sie eine PIPELINE-Ressource. Die PIPELINE-Ressource definiert die Position der Pipelinekonfigurationsdatei im Attribut `CONFIGFILE`.
9. Optional: Erstellen Sie eine URIMAP-Standardressource für z/OS Connect. URIMAP-Ressourcen werden verwendet, um eine TRANSACTION und eine Standard-Benutzer-ID zu z/OS Connect-Arbeit zuzuordnen. Mindestens eine URIMAP-Ressource kann verwendet werden, um eine Standardrichtlinie für z/OS Connect zu konfigurieren. Eine URIMAP-Beispielkonfiguration und weitere Informationen zu Konfigurationsoptionen finden Sie unter Configuring permissions for z/OS Connect Services and APIs:

Anmerkung:

z/OS Connect führt eine zusätzliche Authentifizierung für einzelne HTTP-Anforderungen durch, sodass die Anwendungstasks, die in CICS ausgeführt werden, typischerweise einer spezifischeren Benutzer-ID zugeordnet werden als es die ursprüngliche Benutzer-ID aus der URIMAP ist. Die ursprüngliche Benutzer-ID ist nur wirksam, bis eine benutzerspezifische Authentifizierung in z/OS Connect stattfindet.

Results

Ihre z/OS Connect for CICS 1.0-Instanz wird konfiguriert. Sie können die Basis-konfiguration testen, indem Sie diese URL in einen Web-Browser eingeben: `https://hostname:portnummer/zosConnect/services`. Dabei ist *hostname* die IP-Adresse oder der Hostname des Systems, auf dem die CICS-Region, die z/OS Connect for CICS 1.0 hostet, ausgeführt wird. Und *portnummer* ist der HTTPS-Port (**httpsPort**), der im Element `<httpEndpoint>` der Datei `server.xml` angegeben ist. Der Web-Browser zeigt eine Liste installierter Services an. Da noch keine Services installiert wurden, ist die Liste leer.

Wenn Sie die Antwort HTTP 403 AuthorizationFailed statt der erwarteten Serviceliste empfangen, überprüfen Sie die Sicherheitskonfiguration aus Schritt 2. Es ist wahrscheinlich, dass der authentifizierte Benutzer nicht für die Verwendung von z/OS Connect autorisiert ist.

What to do next

Sie sind jetzt bereit, JSON-Web-Services in z/OS Connect for CICS 1.0 zu implementieren.

z/OS Connect for CICS für einen CICS-JSON-Web-Service konfigurieren:

Nach der Erstkonfiguration von z/OS Connect for CICS können Sie die Software für einen CICS JSON-Web-Service konfigurieren. JSON-Web-Services werden in z/OS Connect for CICS auf ähnliche Weise implementiert wie in anderen CICS PIPELINE-Umgebungen.

Before you begin

Sie müssen die Basiskonfiguration von z/OS Connect abschließen, bevor Sie diese Task starten. Informationen zu z/OS Connect for CICS 1.0 finden Sie unter „z/OS Connect for CICS 1.0 konfigurieren“ auf Seite 77. Informationen zu z/OS Connect Enterprise Edition finden Sie unter „z/OS Connect Enterprise Edition konfigurieren“ auf Seite 84. Sie benötigen außerdem eine JSON-WSBind-Datei für jeden Service, den Sie implementieren möchten. Diese Bindungsdateien können mithilfe des CICS-JSON-Assistenten (den Dienstprogrammen **DFHLS2JS** und **DFHJS2LS**) generiert werden. Weitere Informationen zu diesen Dienstprogrammen finden Sie unter The CICS JSON assistant.

About this task

Services können in z/OS Connect als verwaltete z/OS Connect-Services oder als verwaltete CICS-Services implementiert werden. In diesem Thema wird die Implementierung dieser Services als verwaltete CICS Services erläutert. Durch die Implementierung von Services als verwaltete CICS-Services gibt es WEBSERVICE-Ressourcen für jeden Service. Dieser Implementierungsmechanismus ist kompatibel

mit älterer JSON-Web-Service-Technologie in CICS, bei der Services in z/OS Connect ähnlich wie in anderen CICS-PIPELINE-Umgebungen implementiert werden, darunter SOAP-Web-Services.

Wenn Sie z/OS Connect Enterprise Edition V3 verwenden, können Sie unter Umständen bessere Ergebnisse erzielen, indem Sie die Services als verwaltete z/OS Connect-Ressourcen implementieren, wie unter z/OS Connect Enterprise Edition V3.0 product documentation beschrieben.

Anmerkung: Diese Task findet Anwendung, wenn Sie z/OS Connect Enterprise Edition V1.0 oder den CICS-Service-Provider verwenden, der mit CICS TS bereitgestellt wird. Informationen zur Verwendung des CICS-Service-Providers, der mit z/OS Connect Enterprise Edition v3.0 bereitgestellt wird, finden Sie unter Automated service archive management in z/OS Connect Enterprise Edition V3.0 product documentation.

Procedure

- Installieren Sie eine WEBSERVICE-Ressource, die die WSBIND-Datei der entsprechenden PIPELINE-Ressource zuordnet. Wählen Sie einen aussagekräftigen Namen für den WEBSERVICE, da dieser Name auch als Name des Service in z/OS Connect verwendet wird. Optional können Sie den Befehl **PERFORM PIPELINE SCAN** verwenden, um WEBSERVICE-Ressourcen zu installieren. Welche Methode auch immer Sie verwenden möchten, stellen Sie sicher, dass Sie die folgenden Informationen zu URIMAPs berücksichtigen:

Der URI, unter dem der Service in z/OS Connect verfügbar ist, stammt von einer der folgenden Positionen:

- Die Standardnamenskonvention, die von z/OS Connect verwendet wird.
- Der URI, der in der WSBIND-Datei gespeichert wird, wenn die zugeordnete WEBSERVICE-Ressource mithilfe des Befehls **PERFORM PIPELINE SCAN** installiert wird.
- Die servicespezifische Konfiguration in der Liberty-Datei `server.xml`, wenn der Konfigurationsparameter **invokeURI** verwendet wird.
- Nur z/OS Connect Enterprise Edition: Die Anwendungsarchivdatei (`.aar`-Datei) für eine API, die den Service einbindet.

Wenn Sie sich auf die Standardnamenskonvention verlassen, wird der Service typischerweise über den folgenden URI verfügbar gemacht:

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

In diesem Beispiel bezieht sich `<servicename>` auf den Namen des Service (genauer gesagt, den Namen der WEBSERVICE-Ressource) und `<hostname>` sowie `<port>` werden der Konfiguration des Liberty-Servers entnommen.

- Optional: Installieren Sie eine URIMAP-Ressource für den z/OS Connect-Service. Eine URIMAP-Ressource wird von CICS verwendet, um die Arbeit für den Service einer bestimmten Transaktions-ID in CICS und einer ursprünglichen Benutzer-ID zuzuordnen. Mit einer einzelnen URIMAP können Sie einen oder mehrere Service konfigurieren. Sie können auch eine Standard-URIMAP-Ressource definieren, wenn Sie CICS for z/OS Connect konfigurieren. Wenn keine URIMAP vorhanden ist, um den URI für einen z/OS Connect-Service abzugleichen, werden die Tasks der Anwendung unter der Transaktion CJS ausgeführt und die CICS-Standardbenutzer-ID (typischerweise CICSUSER) wird als ursprüngliche Benutzer-ID verwendet. Weitere Informationen zur Rolle der ursprünglichen Benutzer-ID finden Sie unter Configuring permissions for z/OS Connect Services and APIs.

Wenn Sie eine URIMAP-Ressource erstellen, stellen Sie sicher, dass die zugeordnete Benutzer-ID berechtigt ist, die angegebene Transaktion auszuführen.

Sie können auch eine URIMAP verwenden, um den URI einer bestimmten WEBSERVICE-Ressource zuzuordnen. Wenn eine URIMAP eng an einen WEBSERVICE gebunden ist, wird der Ziel-WEBSERVICE für alle HTTP-Anforderungen verwendet, die mit der URIMAP übereinstimmen. Wenn kein WEBSERVICE angegeben ist, wird der WEBSERVICE basierend auf dem Namen des z/OS Connect-Service ausgewählt. Wenn das WEBSERVICE-Attribut der URIMAP leer gelassen wird, führt z/OS Connect den Abgleich selbst aus, wodurch z/OS Connect Enterprise Edition verschiedene Services unterschiedlichen HTTP-Methoden in einer API zuordnen kann.

- Optional: Aktualisieren Sie die Datei `server.xml`. Manche Verwendungsmuster können Änderungen in der Liberty-Serverkonfigurationsdatei `server.xml` erforderlich machen. In der Regel ist es nicht erforderlich, servicespezifische Änderungen in der Datei `server.xml` vorzunehmen, es sei denn, Sie benötigen eine spezielle Verarbeitung für einen Service. Sie können beispielsweise entscheiden, eine bestimmte Gruppe von z/OS Connect-Interceptors einem Service in der Datei `server.xml` zuzuordnen, oder einen Service mithilfe eines URI zugänglich machen, der nicht mit der z/OS Connect-Standardnamenskonvention übereinstimmt.

Führen Sie diese Schritte aus, um einen Service in der Datei `server.xml` zu definieren:

1. Stellen Sie sicher, dass der gewünschte Servicename nicht mit dem Namen einer WEBSERVICE-Ressource in CICS übereinstimmt. CICS fügt automatisch Konfigurationsinformationen in z/OS Connect ein und wenn ein explizit definierter Service und ein CICS-WEBSERVICE denselben Namen haben, ist das resultierende Verhalten unvorhersehbar.
2. Legen Sie einen passenden Wert für das Attribut **invokeURI** in der Datei `server.xml` fest, das mit dem URI in der URIMAP übereinstimmt.
3. Binden Sie den Service an das **CICSEndpointService**-Element `serviceRef`.
4. Stellen Sie sicher, dass eine URIMAP vorhanden ist, um den URI für den Service präzise mit dem WEBSERVICE abzugleichen, der verwendet werden soll. Wenn in **Schritt 2** ein Attribut **invokeURI** festgelegt wurde, muss die URIMAP diesen URI abgleichen. Andernfalls muss die URIMAP den z/OS Connect-Standard für die URI-Namenskonvention verwenden.

Das folgende Beispiel zeigt eine explizite z/OS Connect for CICS 1.0-Servicedeclaration in der Datei `server.xml`:

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

Das folgende Beispiel zeigt die äquivalente Deklaration für z/OS Connect Enterprise Edition in der Datei `server.xml`:

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

In beiden Beispielen wird der URI `/json/myCustomService` dem z/OS Connect for CICS-Empfängerprogramm, `CICSService1`, zugeordnet.

Results

Sie können die Konfiguration für z/OS Connect for CICS testen, indem Sie diese URL in einen Web-Browser eingeben: `https://hostname:portnummer/zosConnect/`

services. *hostname* ist die IP-Adresse oder der Name der CICS-Region, die z/OS Connect for CICS hostet. *portnummer* ist der HTTPS-Port (**httpsPort**), der im Abschnitt <httpEndpoint> der Konfigurationsdatei *server.xml* konfiguriert ist. Der Web-Browser zeigt eine Liste der installierten Services an.

Der Service kann jetzt von einem JSON-Client aufgerufen werden, der dieselben Werte für *hostname* und *portnummer* verwendet.

Für jeden installierten Service gibt es einen Eintrag in der Liste *zosConnect/services*, z. B.:

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

In diesem Beispiel wurde die zugeordnete WEBSERVICE-Ressource "EXAMPLE" genannt und diese Servicedefinition wurde dynamisch von CICS erstellt, als die WEBSERVICE-Ressource EXAMPLE installiert wurde. Sie können einen Web-Browser verwenden, um <https://hostname:portnummer/zosConnect/services/EXAMPLE> aufzurufen. Es wird ein Dokument mit weiteren Details über den Service zurückgegeben, ähnlich dem folgenden:

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

In diesem Beispiel ist "address" der URI, unter dem der Service verfügbar gemacht wird. Dieser URI kann aus Informationen in einer WSBind-Datei, aus einem Attribut **invokeURI** oder aus der z/OS Connect-Standardnamenskonvention abgeleitet sein.

Wenn eine Anforderung am Liberty-JVM-Server eintrifft, wird sie dem z/OS Connect for CICS-Empfänger zugeordnet, der die Informationen aus der Konfigurationsdatei *server.xml* verwendet. Eine neue CICS-Task beginnt, diese Arbeit auszuführen, und wird einer spezifischen WEBSERVICE-Ressource zugeordnet, die die Informationen aus der URIMAP-Ressource verwendet. Der Datenumsetzungsprozess findet auf dem Liberty-JVM-Server statt und das CICS-Zielfprogramm ist wie in der WSBind-Datei angegeben angehängt.

Von JSON-Web-Services zu z/OS Connect migrieren:

JSON-Services können aus einer älteren JSON-Web-Service-Technologie in CICS oder sogar aus dem noch älteren Feature-Pack für mobile Erweiterungen erneut in z/OS Connect implementiert werden. Eine solche erneute Implementierung setzt voraus, dass z/OS Connect im Kompatibilitätsmodus konfiguriert ist. Dieser Modus ist sowohl in z/OS Connect for CICS 1.0 als auch in z/OS Connect Enterprise Edition verfügbar.

About this task

In dieser Task bezieht sich *z/OS Connect* sowohl auf *z/OS Connect for CICS 1.0* als auch auf *z/OS Connect Enterprise Edition*.

Die erneute Implementierung eines JSON-Web-Service aus Java Pipelines for JSON (wie mit CICS Transaction Server Feature Pack for Mobile Extensions V1 verwendet) in *z/OS Connect* ist ein einfacher Prozess. Die *WSBind*-Dateien, die für die Java Pipelines for JSON und für *z/OS Connect* verwendet werden, werden mithilfe derselben Tools (*DFHLS2JS* und *DFHJS2LS*) erzeugt und sind vollständig kompatibel miteinander. Wenn Sie diese Funktionalität in CICS erkunden, können Sie Beispiele für JSON-Web-Services in den IBM Redbooks: *Implementing IBM CICS JSON Web Services for Mobile Applications* finden.

Die *z/OS Connect*-Umgebung unterstützt die Verwendung von SSL zwischen der Clientanwendung und CICS. Wenn Ihre vorhandene Java Pipelines for JSON-Umgebung SSL nicht verwendet, umfasst die Konvertierung in *z/OS Connect* einen zusätzlichen Schritt.

Procedure

1. Erstellen Sie die erforderliche *z/OS Connect*-Infrastruktur, indem Sie die Anweisungen aus *Configuring z/OS Connect for CICS* verwenden. Als Teil dieser Konfiguration wählen Sie eine *SSL-TCP/IP-Portnummer* aus, an der der *Libertry-JVM-Server* für eingehende Verbindungen empfangsbereit ist. Sie haben zwei Möglichkeiten:
 - a. Wählen Sie eine andere Portnummer aus als die von *TCPIP SERVICE* für die Java Pipelines for JSON verwendete. Diese Option hat den Vorteil, dass beide Umgebungen gleichzeitig auf verschiedenen *TCP/IP-Ports* installiert werden können. Das bedeutet, dass Clientprogramme mit dem neuen JSON-Service als Ziel aktualisiert werden müssen. Wenn die alte Umgebung *SSL* nicht verwendet hat, erfordert die Konvertierung in *z/OS Connect* Änderungen an dem *URI*, sodass diese Option geeigneter ist.
 - b. Wählen Sie dieselbe Portnummer aus wie die von *TCPIP SERVICE* für die Java Pipelines for JSON verwendete. Die Portnummern in den *URIs*, die von den Clientprogrammen verwendet werden, müssen nicht geändert werden, aber die beiden Umgebungen können nicht gleichzeitig installiert werden. Der *URI* muss möglicherweise aus anderen Gründen geändert werden, z. B. der Wechsel von *HTTP* zu *HTTPS* bei der Aktivierung von *SSL*.
2. Implementieren Sie die *WSBind*-Dateien in *z/OS Connect*. Ihre vorhandenen *WSBind*-Dateien sind umfassend kompatibel mit *z/OS Connect*. Befolgen Sie die Schritte zur Implementierung eines neuen JSON-Web-Service in *z/OS Connect*, wie unter *z/OS Connect* für einen CICS-JSON-Web-Service konfigurieren beschrieben. CICS lässt nicht zu, dass zwei *WEBSERVICE*-Ressourcen mit demselben Namen installiert werden. Sie haben daher zwei Möglichkeiten:
 - a. Verwerfen Sie den ursprünglichen *WEBSERVICE*, damit der neue mit demselben Namen installiert werden kann.
 - b. Benennen Sie den neuen *WEBSERVICE* um, um einen Konflikt zu vermeiden.
3. Wenn Sie die Verarbeitung der Java Pipelines for JSON unter Verwendung von *PIPELINE-Handlerprogrammen* angepasst haben, überlegen Sie, ob diese Anpassungen noch erforderlich sind. Falls nicht, erstellen Sie *z/OS Connect-Interceptor-Programme* mit äquivalenten Funktionen und implementieren Sie diese als globale *Interceptors*. Weitere Informationen zu *z/OS Connect-Interceptors* finden Sie unter *Defining z/OS Connect interceptors*.

Results

Sie sind jetzt bereit, Ihre neuen z/OS Connect-JSON-Web-Services zu testen. Wenn Sie diese Services mithilfe derselben Portnummer wie in Java Pipelines for JSON implementiert haben, sind keine Änderungen am Client erforderlich (es sei denn, die Sicherheitskonfiguration hat sich geändert, z. B. bei der Aktivierung von SSL). Wenn Sie die Portnummer oder den URI für den Service geändert haben, muss der Client geändert werden.

z/OS Connect Enterprise Edition konfigurieren:

z/OS Connect Enterprise Edition ist ein separat erhältliches Produkt; es wird nicht als Teil von CICS TS bereitgestellt. Es ergänzt die Basisservices von z/OS Connect for CICS 1.0 um zusätzliche Funktionalitäten und kann verwendet werden, um die älteren JSON-Web-Service-basierten Services von CICS sowie die Standardimplementierungsartefakte von z/OS Connect wie SAR- und AAR-Dateien zu hosten. Sie müssen eine JVMSERVER-Ressource konfigurieren, bevor Sie JSON-Services implementieren können. Sie können auch eine PIPELINE-Ressource konfigurieren. Diese Erstkonfiguration ist nur einmal erforderlich.

Before you begin

Installieren Sie die z/OS Connect Enterprise Edition-Laufzeit. Befolgen Sie die Anweisungen in der z/OS Connect Enterprise Edition V3.0 product documentation. Wenn Sie z/OS Connect Enterprise Edition dafür vorbereiten, eingebettet in CICS ausgeführt zu werden, müssen Sie keine separate Liberty-Serverinstanz erstellen. Sie müssen sicherstellen, dass die Dateisystemkomponenten unter zFS verfügbar sind. Die Installation des API-Toolkits, das mit z/OS Connect Enterprise Edition bereitgestellt wird, ist nicht Teil dieser Task.

About this task

Wenn Sie z/OS Connect for CICS 1.0 zuvor installiert haben, verfügen Sie möglicherweise bereits über eine geeignete Konfiguration und können einige Schritte in der folgenden Task überspringen. Welche Schritte Sie überspringen können, ist in der Prozedur angegeben.

Procedure

1. Erstellen Sie eine JVMSERVER-Ressource und konfigurieren Sie sie für die Unterstützung eines WebSphere Liberty-Servers. Weitere Informationen zum Erstellen eines WebSphere Liberty-JVMSERVER finden Sie unter *Configuring a Liberty JVM server*.
Wenn Sie z/OS Connect for CICS 1.0 bereits installiert haben, können Sie diesen Schritt überspringen.
2. Fügen Sie ZCEE_INSTALL_DIR zu Ihren JVM-Serveroptionen hinzu. Weitere Informationen zu der Option und ein Beispiel finden Sie unter *Symbols used in the JVM profile*.
3. Konfigurieren Sie WebSphere Liberty für Ihre Sicherheitsanforderungen. Standardmäßig wird die Verwendung von clientzertifizierten SSL-Zertifikaten verwendet. Um HTTP-Basisauthentifizierung zu aktivieren, fügen Sie die folgende Konfigurationsoption zur Datei `server.xml` hinzu:

```
<!-- Failover zu HTTP-Basisauthentifizierung zulassen -->  
<webAppSecurity allowFailOverToBasicAuth="true"/>
```

Sie müssen außerdem Benutzern von z/OS Connect die Sicherheitsrolle `zosConnectAccess` erteilen. Weitere Informationen zur WebSphere Liberty-Si-

cherheit finden Sie unter Configuring security for a Liberty JVM server. Weitere Informationen zur z/OS Connect-Sicherheit finden Sie unter Sicherheit für z/OS Connect.

Wenn Sie z/OS Connect for CICS 1.0 bereits installiert haben, können Sie diesen Schritt überspringen.

What to do next

Befolgen Sie die Schritte für Ihre spezifische Version von z/OS Connect Enterprise Edition, wie in den folgenden Tasks beschrieben.

z/OS Connect for CICS für einen CICS-JSON-Web-Service konfigurieren:

Nach der Erstkonfiguration von z/OS Connect for CICS können Sie die Software für einen CICS JSON-Web-Service konfigurieren. JSON-Web-Services werden in z/OS Connect for CICS auf ähnliche Weise implementiert wie in anderen CICS PIPELINE-Umgebungen.

Before you begin

Sie müssen die Basiskonfiguration von z/OS Connect abschließen, bevor Sie diese Task starten. Informationen zu z/OS Connect for CICS 1.0 finden Sie unter „z/OS Connect for CICS 1.0 konfigurieren“ auf Seite 77. Informationen zu z/OS Connect Enterprise Edition finden Sie unter „z/OS Connect Enterprise Edition konfigurieren“ auf Seite 84. Sie benötigen außerdem eine JSON-WSBind-Datei für jeden Service, den Sie implementieren möchten. Diese Bindungsdateien können mithilfe des CICS-JSON-Assistenten (den Dienstprogrammen **DFHLS2JS** und **DFHJS2LS**) generiert werden. Weitere Informationen zu diesen Dienstprogrammen finden Sie unter The CICS JSON assistant.

About this task

Services können in z/OS Connect als verwaltete z/OS Connect-Services oder als verwaltete CICS-Services implementiert werden. In diesem Thema wird die Implementierung dieser Services als verwaltete CICS Services erläutert. Durch die Implementierung von Services als verwaltete CICS-Services gibt es WEBSERVICE-Ressourcen für jeden Service. Dieser Implementierungsmechanismus ist kompatibel mit älterer JSON-Web-Service-Technologie in CICS, bei der Services in z/OS Connect ähnlich wie in anderen CICS-PIPELINE-Umgebungen implementiert werden, darunter SOAP-Web-Services.

Wenn Sie z/OS Connect Enterprise Edition V3 verwenden, können Sie unter Umständen bessere Ergebnisse erzielen, indem Sie die Services als verwaltete z/OS Connect-Ressourcen implementieren, wie unter z/OS Connect Enterprise Edition V3.0 product documentation beschrieben.

Anmerkung: Diese Task findet Anwendung, wenn Sie z/OS Connect Enterprise Edition V1.0 oder den CICS-Service-Provider verwenden, der mit CICS TS bereitgestellt wird. Informationen zur Verwendung des CICS-Service-Providers, der mit z/OS Connect Enterprise Edition v3.0 bereitgestellt wird, finden Sie unter Automated service archive management in z/OS Connect Enterprise Edition V3.0 product documentation.

Procedure

- Installieren Sie eine WEBSERVICE-Ressource, die die WSBind-Datei der entsprechenden PIPELINE-Ressource zuordnet. Wählen Sie einen aussagekräftigen Na-

men für den WEBSERVICE, da dieser Name auch als Name des Service in z/OS Connect verwendet wird. Optional können Sie den Befehl **PERFORM PIPELINE SCAN** verwenden, um WEBSERVICE-Ressourcen zu installieren. Welche Methode auch immer Sie verwenden möchten, stellen Sie sicher, dass Sie die folgenden Informationen zu URIMAPs berücksichtigen:

Der URI, unter dem der Service in z/OS Connect verfügbar ist, stammt von einer der folgenden Positionen:

- Die Standardnamenskonvention, die von z/OS Connect verwendet wird.
- Der URI, der in der WSBIND-Datei gespeichert wird, wenn die zugeordnete WEBSERVICE-Ressource mithilfe des Befehls **PERFORM PIPELINE SCAN** installiert wird.
- Die servicespezifische Konfiguration in der Liberty-Datei `server.xml`, wenn der Konfigurationsparameter **invokeURI** verwendet wird.
- Nur z/OS Connect Enterprise Edition: Die Anwendungsarchivdatei (.aar-Datei) für eine API, die den Service einbindet.

Wenn Sie sich auf die Standardnamenskonvention verlassen, wird der Service typischerweise über den folgenden URI verfügbar gemacht:

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

In diesem Beispiel bezieht sich `<servicename>` auf den Namen des Service (genauer gesagt, den Namen der WEBSERVICE-Ressource) und `<hostname>` sowie `<port>` werden der Konfiguration des Liberty-Servers entnommen.

- Optional: Installieren Sie eine URIMAP-Ressource für den z/OS Connect-Service. Eine URIMAP-Ressource wird von CICS verwendet, um die Arbeit für den Service einer bestimmten Transaktions-ID in CICS und einer ursprünglichen Benutzer-ID zuzuordnen. Mit einer einzelnen URIMAP können Sie einen oder mehrere Service konfigurieren. Sie können auch eine Standard-URIMAP-Ressource definieren, wenn Sie CICS for z/OS Connect konfigurieren. Wenn keine URIMAP vorhanden ist, um den URI für einen z/OS Connect-Service abzugleichen, werden die Tasks der Anwendung unter der Transaktion CJSA ausgeführt und die CICS-Standardbenutzer-ID (typischerweise CICSUSER) wird als ursprüngliche Benutzer-ID verwendet. Weitere Informationen zur Rolle der ursprünglichen Benutzer-ID finden Sie unter *Configuring permissions for z/OS Connect Services and APIs*.

Wenn Sie eine URIMAP-Ressource erstellen, stellen Sie sicher, dass die zugeordnete Benutzer-ID berechtigt ist, die angegebene Transaktion auszuführen.

Sie können auch eine URIMAP verwenden, um den URI einer bestimmten WEBSERVICE-Ressource zuzuordnen. Wenn eine URIMAP eng an einen WEBSERVICE gebunden ist, wird der Ziel-WEBSERVICE für alle HTTP-Anforderungen verwendet, die mit der URIMAP übereinstimmen. Wenn kein WEBSERVICE angegeben ist, wird der WEBSERVICE basierend auf dem Namen des z/OS Connect-Service ausgewählt. Wenn das WEBSERVICE-Attribut der URIMAP leer gelassen wird, führt z/OS Connect den Abgleich selbst aus, wodurch z/OS Connect Enterprise Edition verschiedene Services unterschiedlichen HTTP-Methoden in einer API zuordnen kann.

- Optional: Aktualisieren Sie die Datei `server.xml`. Manche Verwendungsmuster können Änderungen in der Liberty-Serverkonfigurationsdatei `server.xml` erforderlich machen. In der Regel ist es nicht erforderlich, servicespezifische Änderungen in der Datei `server.xml` vorzunehmen, es sei denn, Sie benötigen eine spezielle Verarbeitung für einen Service. Sie können beispielsweise entscheiden, eine bestimmte Gruppe von z/OS Connect-Interceptors einem Service in der Da-

tei `server.xml` zuzuordnen, oder einen Service mithilfe eines URI zugänglich machen, der nicht mit der z/OS Connect-Standardnamenskonvention übereinstimmt.

Führen Sie diese Schritte aus, um einen Service in der Datei `server.xml` zu definieren:

1. Stellen Sie sicher, dass der gewünschte Servicename nicht mit dem Namen einer WEBSERVICE-Ressource in CICS übereinstimmt. CICS fügt automatisch Konfigurationsinformationen in z/OS Connect ein und wenn ein explizit definierter Service und ein CICS-WEBSERVICE denselben Namen haben, ist das resultierende Verhalten unvorhersehbar.
2. Legen Sie einen passenden Wert für das Attribut **invokeURI** in der Datei `server.xml` fest, das mit dem URI in der URIMAP übereinstimmt.
3. Binden Sie den Service an das **CICSEndpointService**-Element `serviceRef`.
4. Stellen Sie sicher, dass eine URIMAP vorhanden ist, um den URI für den Service präzise mit dem WEBSERVICE abzugleichen, der verwendet werden soll. Wenn in **Schritt 2** ein Attribut **invokeURI** festgelegt wurde, muss die URIMAP diesen URI abgleichen. Andernfalls muss die URIMAP den z/OS Connect-Standard für die URI-Namenskonvention verwenden.

Das folgende Beispiel zeigt eine explizite z/OS Connect for CICS 1.0-Servicedeclaration in der Datei `server.xml`:

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

Das folgende Beispiel zeigt die äquivalente Deklaration für z/OS Connect Enterprise Edition in der Datei `server.xml`:

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

In beiden Beispielen wird der URI `/json/myCustomService` dem z/OS Connect for CICS-Empfängerprogramm, `CICSService1`, zugeordnet.

Results

Sie können die Konfiguration für z/OS Connect for CICS testen, indem Sie diese URL in einen Web-Browser eingeben: `https://hostname:portnummer/zosConnect/services`. *hostname* ist die IP-Adresse oder der Name der CICS-Region, die z/OS Connect for CICS hostet. *portnummer* ist der HTTPS-Port (**httpsPort**), der im Abschnitt `<httpEndpoint>` der Konfigurationsdatei `server.xml` konfiguriert ist. Der Web-Browser zeigt eine Liste der installierten Services an.

Der Service kann jetzt von einem JSON-Client aufgerufen werden, der dieselben Werte für *hostname* und *portnummer* verwendet.

Für jeden installierten Service gibt es einen Eintrag in der Liste `zosConnect/services`, z. B.:

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

In diesem Beispiel wurde die zugeordnete WEBSERVICE-Ressource "EXAMPLE" genannt und diese Servicedefinition wurde dynamisch von CICS erstellt, als die WEBSERVICE-Ressource EXAMPLE installiert wurde. Sie können einen Web-Browser verwenden, um `https://hostname:portnumber/zosConnect/services/EXAMPLE` aufzurufen. Es wird ein Dokument mit weiteren Details über den Service zurückgegeben, ähnlich dem folgenden:

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

In diesem Beispiel ist "address" der URI, unter dem der Service verfügbar gemacht wird. Dieser URI kann aus Informationen in einer WSBind-Datei, aus einem Attribut **invokeURI** oder aus der z/OS Connect-Standardnamenskonvention abgeleitet sein.

Wenn eine Anforderung am Liberty-JVM-Server eintrifft, wird sie dem z/OS Connect for CICS-Empfänger zugeordnet, der die Informationen aus der Konfigurationsdatei `server.xml` verwendet. Eine neue CICS-Task beginnt, diese Arbeit auszuführen, und wird einer spezifischen WEBSERVICE-Ressource zugeordnet, die die Informationen aus der URIMAP-Ressource verwendet. Der Datenumsetzungsprozess findet auf dem Liberty-JVM-Server statt und das CICS-Zielprogramm ist wie in der WSBind-Datei angegeben angehängt.

Von JSON-Web-Services zu z/OS Connect migrieren:

JSON-Services können aus einer älteren JSON-Web-Service-Technologie in CICS oder sogar aus dem noch älteren Feature-Pack für mobile Erweiterungen erneut in z/OS Connect implementiert werden. Eine solche erneute Implementierung setzt voraus, dass z/OS Connect im Kompatibilitätsmodus konfiguriert ist. Dieser Modus ist sowohl in z/OS Connect for CICS 1.0 als auch in z/OS Connect Enterprise Edition verfügbar.

About this task

In dieser Task bezieht sich *z/OS Connect* sowohl auf z/OS Connect for CICS 1.0 als auch auf z/OS Connect Enterprise Edition.

Die erneute Implementierung eines JSON-Web-Service aus Java Pipelines for JSON (wie mit CICS Transaction Server Feature Pack for Mobile Extensions V1 verwendet) in z/OS Connect ist ein einfacher Prozess. Die WSBind-Dateien, die für die Java Pipelines for JSON und für z/OS Connect verwendet werden, werden mithilfe derselben Tools (DFHLS2JS und DFHJS2LS) erzeugt und sind vollständig kompatibel miteinander. Wenn Sie diese Funktionalität in CICS erkunden, können Sie Beispiele für JSON-Web-Services in den IBM Redbooks: *Implementing IBM CICS JSON Web Services for Mobile Applications* finden.

Die z/OS Connect-Umgebung unterstützt die Verwendung von SSL zwischen der Clientanwendung und CICS. Wenn Ihre vorhandene Java Pipelines for JSON-Um-

gebung SSL nicht verwendet, umfasst die Konvertierung in z/OS Connect einen zusätzlichen Schritt.

Procedure

1. Erstellen Sie die erforderliche z/OS Connect-Infrastruktur, indem Sie die Anweisungen aus Configuring z/OS Connect for CICS verwenden. Als Teil dieser Konfiguration wählen Sie eine SSL-TCP/IP-Portnummer aus, an der der Liberty-JVM-Server für eingehende Verbindungen empfangsbereit ist. Sie haben zwei Möglichkeiten:
 - a. Wählen Sie eine andere Portnummer aus als die von TCPIP SERVICE für die Java Pipelines for JSON verwendete. Diese Option hat den Vorteil, dass beide Umgebungen gleichzeitig auf verschiedenen TCP/IP-Ports installiert werden können. Das bedeutet, dass Clientprogramme mit dem neuen JSON-Service als Ziel aktualisiert werden müssen. Wenn die alte Umgebung SSL nicht verwendet hat, erfordert die Konvertierung in z/OS Connect Änderungen an dem URI, sodass diese Option geeigneter ist.
 - b. Wählen Sie dieselbe Portnummer aus wie die von TCPIP SERVICE für die Java Pipelines for JSON verwendete. Die Portnummern in den URIs, die von den Clientprogrammen verwendet werden, müssen nicht geändert werden, aber die beiden Umgebungen können nicht gleichzeitig installiert werden. Der URI muss möglicherweise aus anderen Gründen geändert werden, z. B. der Wechsel von HTTP zu HTTPS bei der Aktivierung von SSL.
2. Implementieren Sie die WSBIND-Dateien in z/OS Connect. Ihre vorhandenen WSBIND-Dateien sind umfassend kompatibel mit z/OS Connect. Befolgen Sie die Schritte zur Implementierung eines neuen JSON-Web-Service in z/OS Connect, wie unter z/OS Connect für einen CICS-JSON-Web-Service konfigurieren beschrieben. CICS lässt nicht zu, dass zwei WEBSERVICE-Ressourcen mit demselben Namen installiert werden. Sie haben daher zwei Möglichkeiten:
 - a. Verwerfen Sie den ursprünglichen WEBSERVICE, damit der neue mit demselben Namen installiert werden kann.
 - b. Benennen Sie den neuen WEBSERVICE um, um einen Konflikt zu vermeiden.
3. Wenn Sie die Verarbeitung der Java Pipelines for JSON unter Verwendung von PIPELINE-Handlerprogrammen angepasst haben, überlegen Sie, ob diese Anpassungen noch erforderlich sind. Falls nicht, erstellen Sie z/OS Connect-Interceptor-Programme mit äquivalenten Funktionen und implementieren Sie diese als globale Interceptors. Weitere Informationen zu z/OS Connect-Interceptors finden Sie unter Defining z/OS Connect interceptors.

Results

Sie sind jetzt bereit, Ihre neuen z/OS Connect-JSON-Web-Services zu testen. Wenn Sie diese Services mithilfe derselben Portnummer wie in Java Pipelines for JSON implementiert haben, sind keine Änderungen am Client erforderlich (es sei denn, die Sicherheitskonfiguration hat sich geändert, z. B. bei der Aktivierung von SSL). Wenn Sie die Portnummer oder den URI für den Service geändert haben, muss der Client geändert werden.

APIs aus z/OS Connect Enterprise Edition verwenden:

Eine primäre Funktion von z/OS Connect Enterprise Edition ist die Fähigkeit, JSON-APIs aus einem oder mehreren JSON-Services zusammenzusetzen. Diese Funktionalität gibt es nicht in z/OS Connect for CICS 1.0. Die optimale API-Entwicklungserfahrung erhalten Sie, indem Sie APIs und Services in z/OS Connect

Enterprise Edition v3.0 implementieren, statt ältere JSON-Web-Service-Technologie in CICS im Kompatibilitätsmodus auszuführen. Weitere Informationen finden Sie in der z/OS Connect Enterprise Edition-Dokumentation.

Details zum API-Toolkit finden Sie unter z/OS Connect Enterprise Edition V3.0 product documentation.

Es gibt einige geringfügige Unterschiede bei der Implementierung von APIs in z/OS Connect Enterprise Edition für CICS, verglichen mit der Implementierung von APIs in einer eigenständigen Installation von z/OS Connect Enterprise Edition. Die folgenden Überlegungen gelten für die Implementierung von APIs in CICS:

- Sie können keine vorhandene WSBIND-Datei für die Verwendung mit einer API umfunktionieren. Die Eingabe des API-Editors muss eine SAR-Datei (Service Archive Resource) umfassen.

Wenn Sie den CICS-Service-Provider, der von CICS TS bereitgestellt wird, verwenden, wird die WSBIND-Datei mit den Assistenten BAQLS2JS oder BAQJS2LS erstellt, die mit z/OS Connect Enterprise Edition verteilt werden. Wenn Sie APIs erstellen möchten, beginnen Sie, indem Sie die erforderlichen WSBIND- und SAR-Dateien mithilfe der Assistenten generieren, die mit z/OS Connect Enterprise Edition bereitgestellt werden.

Sie müssen den Parameter **SERVICE-NAME** in BAQLS2JS oder BAQJS2LS festlegen, um den Namen der WEBSERVICE-Ressource abzugleichen, die in CICS verwendet werden wird. CICS verlässt sich auf die 1:1-Übereinstimmung des Namens des Service in z/OS Connect und des Namens des WEBSERVICE in CICS.

Wenn Sie den CICS-Service-Provider verwenden, der mit z/OS Connect Enterprise Edition bereitgestellt wird, wird eine SAR-Datei entweder mit dem Service-Editor im z/OS Connect Enterprise Edition V3-API-Toolkit oder mit dem Build-Toolkit, das mit z/OS Connect Enterprise Edition verteilt wird, erstellt. Sie müssen diese Tools verwenden, um die erforderlichen SAR-Dateien zu generieren, bevor Sie APIs erstellen können.

- Eine WEBSERVICE-Ressourcendefinition ist erforderlich in CICS, die mit dem Namen des Service übereinstimmt.

Wenn Sie den CICS-Service-Provider verwenden, der von CICS TS bereitgestellt wird, bindet der WEBSERVICE die WSBIND-Datei für den Service ein.

Wenn Sie den CICS-Service-Provider verwenden, der mit z/OS Connect EE V3 bereitgestellt wird, ist keine WEBSERVICE-Ressourcendefinition erforderlich.

- Eine URIMAP-Ressourcendefinition kann auch in CICS implementiert werden. Wenn eine URIMAP verwendet wird, ordnet CICS die Arbeit für die API automatisch einer Transaktions-ID zu und legt eine passende ursprüngliche Benutzer-ID fest.
- Der API-Editor erzeugt eine Anwendungsarchivdatei (AAR) als Ausgabe. Diese muss mithilfe der in z/OS Connect EE verfügbaren Implementierungsmechanismen auf dem z/OS Connect EE-Server implementiert werden. Standardmäßig werden APIs im Verzeichnis /resources/zosconnect/apis im Konfigurationsverzeichnis des Servers implementiert. Wenn dieses Verzeichnis noch nicht vorhanden ist, müssen Sie es erstellen. Oder Sie können ein alternatives Verzeichnis für die APIs erstellen und dieses im Element `zosconnect_apis` in `server.xml` angeben.

Wenn Sie den Überlegungen oben folgen, werden die API und ihre Komponentenservices in CICS implementiert. CICS wird mit passenden URIMAP- und WEBSERVICE-Ressourcen konfiguriert und die AAR-Datei wird im Arbeitsbereich der Li-

berty-Konfiguration implementiert. Die Liberty-Hauptkonfigurationsdatei, `server.xml`, enthält einen Eintrag für jeden implementierten Komponentenservice.

z/OS Connect Enterprise Edition V3.0 für von z/OS Connect verwaltete Services konfigurieren:

z/OS Connect Enterprise Edition ist ein separat erhältliches Produkt; es wird nicht als Teil von CICS TS bereitgestellt. Zunächst müssen Sie die Laufzeitkomponente von z/OS Connect Enterprise Edition installieren und eine zFS-Datei konfigurieren, sodass CICS sie lokalisieren kann. Anschließend müssen Sie einen JVM-Server konfigurieren sowie die Pipelinekonfiguration und Ressourcen für z/OS Connect einrichten, um JSON-Services und -APIs implementieren zu können. Diese Erstkonfiguration ist eine einmalige Aktivität, die es z/OS Connect Enterprise Edition erlaubt, eingebettet auf dem Liberty-Server ausgeführt zu werden, der mit CICS TS bereitgestellt wird.

Before you begin

Installieren Sie die z/OS Connect Enterprise Edition-Laufzeit. Befolgen Sie die Anweisungen in der z/OS Connect Enterprise Edition V3.0 product documentation. Wenn Sie z/OS Connect Enterprise Edition dafür vorbereiten, eingebettet in CICS ausgeführt zu werden, müssen Sie keine separate Liberty-Serverinstanz erstellen. Sie müssen mindestens die Dateisystemkomponenten unter zFS installieren. Die Installation des API-Toolkits, das mit z/OS Connect Enterprise Edition bereitgestellt wird, ist nicht Teil dieser Task.

Haben Sie z/OS Connect for CICS 1.0 bereits installiert? Wenn ja, müssen Sie einige Schritte in dieser Task nicht wiederholen. Welche Schritte Sie überspringen können, ist in der Liste angegeben.

Procedure

1. Stellen Sie sicher, dass alle Schritte unter „z/OS Connect Enterprise Edition konfigurieren“ auf Seite 84 ausgeführt wurden.
2. Aktualisieren Sie die `<featureManager>`-Liste in der Datei `server.xml`, um die Funktion `zosconnect:cicsService-1.0` hinzuzufügen. Beispiel:

```
<featureManager>
  <feature>cicsts:core-1.0</feature>
  <feature>zosconnect:cicsService-1.0</feature>
</featureManager>
```

Anmerkung: Die CICS-Service-Provider, die von CICS TS und z/OS Connect Enterprise Edition v3.0 bereitgestellt werden, sind gegenseitig ausschließend. Wenn Sie ein Upgrade von z/OS Connect for CICS 1.0 auf z/OS Connect Enterprise Edition durchführen, müssen Sie den Funktionscode ändern.

3. Definieren Sie die Elemente für den CICS-Service-Provider. Wenn Sie z/OS Connect for CICS 1.0 installiert haben, können Sie diesen Schritt überspringen, weil Ihre Konfiguration wahrscheinlich bereits geeignet ist.

Konfigurieren Sie die folgenden Elemente in `server.xml`:

- a. Definieren Sie entweder eine lokale CICS-Verbindung oder eine ferne IPIC-Verbindung. Die folgende Beispieldefinition gilt für eine lokale CICS-Verbindung:

```
<zosconnect_cicsLocalConnection id="eciTest"/>
```

Die folgende Beispieldefinition gilt für eine ferne IPIC-Verbindung:

```
<zosconnect_cicsIPICConnection
  id="eciTest"
  host="cicshost.company.com"
  port="1111"/>
```

- b. Aktivieren Sie die Implementierung von Services mithilfe von SAR-Dateien:
Beispiel:

```
<zosconnect_services
  pollingRate="5s"
  updateTrigger="polled">
```

- c. Konfigurieren Sie `zosconnect_zosConnectManager`, um die Sicherheit zu inaktivieren.

Weitere Informationen zum Konfigurieren des CICS-Service-Providers, der mit z/OS Connect Enterprise Edition bereitgestellt wird, finden Sie unter CICS-Service-Provider verwenden.

4. Installieren Sie den JVMSERVER. Prüfen Sie die generierte Datei `messages.log` auf Fehler oder Warnnachrichten. Dieses Protokoll enthält die Nachrichten, die von dem WebSphere Liberty-Server generiert werden, darunter Nachrichten, die von z/OS Connect Enterprise Edition ausgegeben werden, z. B.:

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

Results

Ihre z/OS Connect Enterprise Edition-Instanz wird konfiguriert. Sie können die Basiskonfiguration für z/OS Connect testen, indem Sie diese URL in einen Web-Browser eingeben: `https://hostname:portnumber/zosConnect/services`. Dabei ist *hostname* die IP-Adresse oder der Hostname des Systems, auf dem die CICS-Region, die z/OS Connect hostet, ausgeführt wird. Und *portnummer* ist der HTTPS-Port (**httpsPort**), der im Element `<httpEndpoint>` der Datei `server.xml` angegeben ist. Der Web-Browser zeigt eine Liste installierter Services an. Da noch keine Services installiert wurden, ist die Liste leer.

Wenn Sie die Antwort HTTP 403 AuthorizationFailed statt der erwarteten Serviceliste empfangen, überprüfen Sie den Sicherheitskonfigurationsschritt in „z/OS Connect Enterprise Edition konfigurieren“ auf Seite 84. Der authentifizierte Benutzer ist möglicherweise nicht berechtigt, z/OS Connect zu verwenden.

What to do next

Sie sind jetzt bereit, JSON-Services oder -APIs in z/OS Connect Enterprise Edition zu implementieren. Weitere Informationen zum Implementieren von Services finden Sie unter Exposing z/OS assets as REST APIs in z/OS Connect Enterprise Edition V3.0 product documentation. Weitere Informationen zum Implementieren von APIs finden Sie unter „APIs aus z/OS Connect Enterprise Edition verwenden“ auf Seite 89.

z/OS Connect Enterprise Edition für von CICS verwaltete Services konfigurieren:

z/OS Connect Enterprise Edition ist ein separat erhältliches Produkt; es wird nicht als Teil von CICS TS bereitgestellt. Zunächst müssen Sie die Laufzeitkomponente von z/OS Connect Enterprise Edition installieren und eine zFS-Datei konfigurieren, sodass CICS sie lokalisieren kann. Anschließend müssen Sie einen JVM-Server konfigurieren sowie die Pipelinekonfiguration und Ressourcen für z/OS Connect einrichten, um JSON-Services und -APIs implementieren zu können. Diese Erstkonfi-

guration ist eine einmalige Aktivität, die es z/OS Connect Enterprise Edition erlaubt, eingebettet auf dem Liberty-Server ausgeführt zu werden, der mit CICS TS bereitgestellt wird.

Before you begin

Installieren Sie die z/OS Connect Enterprise Edition-Laufzeit. Befolgen Sie die Anweisungen in der z/OS Connect Enterprise Edition V2.0 product documentation. Wenn Sie z/OS Connect Enterprise Edition dafür vorbereiten, eingebettet in CICS ausgeführt zu werden, müssen Sie keine separate Liberty-Serverinstanz erstellen. Sie müssen mindestens die Dateisystemkomponenten unter zFS installieren. Die Installation des API-Toolkits, das mit z/OS Connect Enterprise Edition bereitgestellt wird, ist nicht Teil dieser Task.

Haben Sie z/OS Connect for CICS 1.0 bereits installiert? Wenn ja, müssen Sie einige Schritte in dieser Task nicht wiederholen. Welche Schritte Sie überspringen können, ist in der Liste angegeben.

Procedure

1. Stellen Sie sicher, dass alle Schritte unter „z/OS Connect Enterprise Edition konfigurieren“ auf Seite 84 ausgeführt wurden.
2. Aktualisieren Sie die <featureManager>-Liste in der Datei server.xml, um die Funktion cicsts:zosConnect-2.0 hinzuzufügen. Beispiel:

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

Anmerkung: Die Funktionen z/OS Connect for CICS 1.0 und z/OS Connect Enterprise Edition sind gegenseitig ausschließend. Wenn Sie ein Upgrade von z/OS Connect for CICS 1.0 auf z/OS Connect Enterprise Edition durchführen, müssen Sie die Funktion ändern.

3. Definieren Sie den z/OS Connect-Service-Controller. Wenn Sie den CICS-Service-Provider verwenden, der mit CICS TS bereitgestellt wird, fügen Sie die folgende Anweisung zur Datei server.xml hinzu:

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

Wenn Sie z/OS Connect for CICS 1.0 installiert haben, können Sie diesen Schritt überspringen, weil Ihre Konfiguration wahrscheinlich bereits geeignet ist.

Wenn Sie den mit z/OS Connect Enterprise Edition V2 bereitgestellten CICS-Service-Provider verwenden, konfigurieren Sie die folgenden Elemente in server.xml:

- a. Definieren Sie entweder eine lokale CICS-Verbindung oder eine ferne IPIC-Verbindung. Die folgende Beispieldefinition gilt für eine lokale CICS-Verbindung:

```
<zosconnect_cicsLocalConnection id="eciTest"/>
```

Die folgende Beispieldefinition gilt für eine ferne IPIC-Verbindung:

```
<zosconnect_cicsIPICConnection
  id="eciTest"
  host="cicshost.company.com"
  port="1111"/>
```

- b. Aktivieren Sie die Implementierung von Services mithilfe von SAR-Dateien:
Beispiel:

```
<zoscconnect_services  
  pollingRate="5s"  
  updateTrigger="polled">
```

Weitere Informationen zur Konfiguration des CICS-Service-Providers für z/OS Connect Enterprise Edition finden Sie unter Using the CICS service provider in z/OS Connect Enterprise Edition V3.0 product documentation.

4. Installieren Sie den JVMSERVER. Prüfen Sie die generierte Datei `messages.log` auf Fehler oder Warnnachrichten. Dieses Protokoll enthält die Nachrichten, die von dem WebSphere Liberty-Server generiert werden, darunter Nachrichten, die von z/OS Connect Enterprise Edition ausgegeben werden, z. B.:

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

5. Erstellen Sie eine XML-Pipelinekonfigurationsdatei. Sie finden ein Beispiel für die Pipelinekonfigurationsdatei `jsonzosconnectprovider.xml` im Verzeichnis `/usr/lpp/cicsts/cicsts55/samples/pipelines/` (wobei `/usr/lpp/cicsts/cicsts55` das Standardinstallationsverzeichnis für CICS-Dateien unter z/OS UNIX ist). Entscheiden Sie, ob Sie die JSON mithilfe von Java auf dem Liberty-JVM-Server parsen möchten (Standard) oder ob der Nicht-Java-JSON-Parser verwendet werden soll.

- Um die JSON mithilfe von Java auf dem Liberty-JVM-Server zu parsen, können Sie das Beispiel für die Pipelinekonfigurationsdatei verwenden, müssen aber DFHWLP im Element `<jvmserver>` durch den Namen Ihres JVMSERVER aus Schritt 2 ersetzen.
- Um die JSON mithilfe des Nicht-Java-Parsers zu parsen, ändern Sie die Beispielkonfigurationsdatei, indem Sie das Attribut `java_parser="no"` an das Element `<provider_pipeline_json>` anhängen, wie im folgenden Beispiel gezeigt:

```
<?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>
```

Ersetzen Sie DFHWLP durch den Namen des JVMSERVER, den Sie am Anfang dieser Prozedur erstellt haben.

Wenn Sie z/OS Connect for CICS 1.0 installiert haben, können Sie diesen Schritt überspringen, weil Ihre Konfiguration wahrscheinlich bereits geeignet ist.

6. Kopieren Sie die Pipelinekonfigurationsdatei in ein passendes Verzeichnis in zFS und stellen Sie sicher, dass die Dateiberechtigungen zulassen, dass die CICS-Region die Datei liest. Weitere Informationen finden Sie unter Pipelinekonfigurationsdateien.

Wenn Sie z/OS Connect for CICS 1.0 installiert haben, können Sie diesen Schritt überspringen, weil Ihre Konfiguration wahrscheinlich bereits geeignet ist.

7. Erstellen Sie eine PIPELINE-Ressource. Die PIPELINE-Ressource definiert die Position der Pipelinekonfigurationsdatei. Versuchen Sie nicht, SCAN-Mechanismen zu verwenden, um Web-Services in dieser Pipeline zu installieren.

Wenn Sie z/OS Connect for CICS 1.0 installiert haben, können Sie diesen Schritt überspringen, weil Ihre Konfiguration wahrscheinlich bereits geeignet ist.

8. Optional: Erstellen Sie eine URIMAP-Standardressource für z/OS Connect. URIMAP-Ressourcen werden verwendet, um eine TRANSACTION und eine

Standard-Benutzer-ID zu z/OS Connect-Arbeit zuzuordnen. Mindestens eine URIMAP-Ressource kann verwendet werden, um eine Standardrichtlinie für z/OS Connect zu konfigurieren. Ein Beispiel für eine URIMAP-Konfiguration und weitere Informationen zu Konfigurationsoptionen finden Sie unter Berechtigungen für z/OS Connect-Services und -APIs konfigurieren.

Anmerkung:

z/OS Connect führt eine zusätzliche Authentifizierung für einzelne HTTP-Anforderungen durch, sodass die Anwendungstasks, die in CICS ausgeführt werden, typischerweise einer spezifischeren Benutzer-ID zugeordnet werden, als es die ursprüngliche Benutzer-ID aus der URIMAP ist.

Diese ursprüngliche Benutzer-ID ist nur wirksam, bis eine benutzerspezifische Authentifizierung in z/OS Connect stattfindet. Verwenden Sie eine spezifische Benutzer-ID wie ZOSCUSER in der URIMAP, statt sich darauf zu verlassen, dass die Standardbenutzer-ID in CICS (in der Regel "CICSUSER") verwendet wird, und erteilen Sie ihr die Berechtigungen, die Zieltransaktionen zu verwenden.

Results

Ihre z/OS Connect Enterprise Edition-Instanz wird konfiguriert. Sie können die Basiskonfiguration für z/OS Connect testen, indem Sie diese URL in einen Web-Browser eingeben: `https://hostname:portnumber/zosConnect/services`. Dabei ist *hostname* die IP-Adresse oder der Hostname des Systems, auf dem die CICS-Region, die z/OS Connect hostet, ausgeführt wird. Und *portnummer* ist der HTTPS-Port (**httpsPort**), der im Element `<httpEndpoint>` der Datei `server.xml` angegeben ist. Der Web-Browser zeigt eine Liste installierter Services an. Da noch keine Services installiert wurden, ist die Liste leer.

Wenn Sie die Antwort HTTP 403 AuthorizationFailed statt der erwarteten Serviceliste empfangen, überprüfen Sie den Sicherheitskonfigurationsschritt in „z/OS Connect Enterprise Edition konfigurieren“ auf Seite 84. Der authentifizierte Benutzer ist möglicherweise nicht berechtigt, z/OS Connect zu verwenden.

What to do next

Sie sind jetzt bereit, JSON-Services oder -APIs in z/OS Connect Enterprise Edition zu implementieren. Weitere Informationen zum Implementieren von Services finden Sie unter *Configuring z/OS Connect for a CICS JSON web service*. Weitere Informationen zum Implementieren von APIs finden Sie unter „APIs aus z/OS Connect Enterprise Edition verwenden“ auf Seite 89.

Pipelinekonfigurationsdateien

Die Konfiguration einer Pipeline zur Verarbeitung einer Web-Service-Anforderung wird in einem XML-Dokument angegeben, das als *Pipelinekonfigurationsdatei* bezeichnet wird.

Die Pipelinekonfigurationsdatei wird im z/OS UNIX System Services-Dateisystem gespeichert und ihr Name wird im Attribut CONFIGFILE einer PIPELINE-Ressourcendefinition angegeben. Verwenden Sie einen geeigneten XML-Editor oder Texteditor für die Arbeit mit Ihren Pipelinekonfigurationsdateien. Die XML-Schemas für die Pipelinekonfigurationsdateien befinden sich im Verzeichnis `/usr/lpp/cicsts/cicsts55/schemas/pipeline/` (wobei `/usr/lpp/cicsts/cicsts55` das

Standardinstallationsverzeichnis für CICS-Dateien unter z/OS UNIX ist). Wenn Sie mit Konfigurationsdateien arbeiten, stellen Sie sicher, dass die Zeichensatzcodierung UTF-8 ist. Wenn Sie eine vorhandene Konfigurationsdatei importieren, die in EBCDIC codiert ist, wird sie automatisch in UTF-8 konvertiert.

Wenn CICS eine Web-Service-Anforderung verarbeitet, verwendet es eine Pipeline von einem oder mehreren Nachrichtenhandlern, um die Anforderung zu verarbeiten. Eine Pipeline wird so konfiguriert, dass sie Aspekte der Ausführungsumgebung bereitstellt, die für verschiedene Kategorien von Anwendungen gelten, z. B. Unterstützung für Web Service Security und Web-Service-Transaktionen. Typischerweise benötigt eine CICS-Region mit einer großen Anzahl von Service-Provider- oder Service-Requester-Anwendungen diverse Pipelinekonfigurationen. Wenn jedoch unterschiedliche Anwendungen dieselben Anforderungen haben, können sie dieselbe Pipelinekonfiguration verwenden.

Anmerkung: Wenn Sie CICS Explorer verwenden, um eine neue PIPELINE-Konfigurationsdatei als Teil eines Bundles zu erstellen, darf keine Konfigurationsdatei mit demselben Namen im Stammverzeichnis des Bundles vorhanden sein.

Es gibt zwei Arten von Pipelinekonfigurationen: Eine beschreibt die Konfiguration einer Service-Provider-Pipeline, die andere beschreibt eine Service-Requester-Pipeline. Jede hat ihr eigenes Schema und ein eigenes Stammelement.

Pipeline	Schema	Stammelement
Service-Provider	Provider.xsd	<provider_pipeline>
Service-Requester	Requester.xsd	<requester_pipeline>

Obwohl viele XML-Elemente in beiden Arten von Pipelinekonfigurationen verwendet werden, sind andere nur in der einen oder der anderen gültig, deshalb können Sie nicht dieselbe Konfigurationsdatei für die Provider- und die Requester-Pipeline verwenden.

Restriction: Namensbereichsqualifizierte Elementnamen werden in der Pipelinekonfigurationsdatei nicht unterstützt.

Die Elemente <provider_pipeline> und <requester_pipeline> haben die folgenden direkt untergeordneten Elemente:

- Ein Element <service>, das die Nachrichtenhandler angibt, die für die einzelnen Anforderungen aufgerufen werden. Dieses Element ist obligatorisch im Element <provider_pipeline> und optional im Element <requester_pipeline>.
- Ein optionales Element <transport>, das die zur Laufzeit ausgewählten Nachrichtenhandler angibt, basierend auf den Ressourcen, die für den Nachrichtentransport verwendet werden.
- Nur für <provider_pipeline>: ein optionales Element <apphandler>, das über einen Kanal angehängte Anwendungshandler angibt.
- Nur für <provider_pipeline>: ein optionales Element <apphandler_class>, das einen Axis2-Anwendungshandler angibt.
- Ein optionales Element <service_parameter_list>, das die Parameter angibt, die für die Nachrichtenhandler in der Pipeline verfügbar sind.

Bestimmten Elementen können Attribute zugeordnet sein. Attributwerte müssen in Anführungszeichen gesetzt sein, um ein gültiges XML-Dokument zu erzeugen.

Der Pipelinekonfigurationsdatei ist eine PIPELINE-Ressource zugeordnet. Zu den Attributen zählt CONFIGFILE, das den Namen der Pipelinekonfigurationsdatei in

z/OS UNIX angibt. Wenn Sie eine PIPELINE-Definition installieren, liest CICS die erforderlichen Informationen, um die Pipeline aus der Datei konfigurieren zu können.

CICS stellt Beispielkonfigurationsdateien bereit, die Sie als Grundlage für die Entwicklung Ihrer eigenen Konfigurationsdateien verwenden können. Sie werden in der Bibliothek `/usr/lpp/cicsts/cicsts55/samples/pipelines` bereitgestellt.

basicsoap11provider.xml

Eine Service-Provider-Pipelinedefinition, die das SOAP 1.1-Protokoll für eine Pipeline verwendet, die Java nicht unterstützt. Die Pipeline verwendet den Nachrichtenhandler `<cics_soap_1.1_handler>` und wird verwendet, wenn die CICS-Anwendung mit dem CICS-Web-Service-Assistenten implementiert wurde.

basicsoap11requester.xml

Eine Service-Requester-Pipelinedefinition, die das SOAP 1.1-Protokoll für eine Pipeline verwendet, die Java nicht unterstützt. Die Pipeline verwendet den Nachrichtenhandler `<cics_soap_1.1_handler>` und wird verwendet, wenn die CICS-Anwendung mit dem CICS-Web-Service-Assistenten implementiert wurde.

basicsoap11javaprovider.xml

Eine Service-Provider-Pipelinedefinition, die das SOAP 1.1-Protokoll für eine Pipeline verwendet, die Java unterstützt. Die Pipeline verwendet den Nachrichtenhandler `<cics_soap_1.1_handler_java>` und wird verwendet, wenn die Anwendung mit dem CICS-Web-Service-Assistenten implementiert wurde. Diese Konfiguration enthält das Element `<jvmserver>`. Dieser Nachrichtenhandler muss so bearbeitet werden, dass er den passenden JVM-Server angibt, bevor die Konfiguration verwendet werden kann.

basicsoap11javarequester.xml

Eine Service-Requester-Pipelinedefinition, die das SOAP 1.1-Protokoll für eine Pipeline verwendet, die Java unterstützt. Die Pipeline verwendet den Nachrichtenhandler `<cics_soap_1.1_handler_java>` und wird verwendet, wenn die Anwendung mit dem CICS-Web-Service-Assistenten implementiert wurde. Diese Konfiguration enthält das Element `<jvmserver>`. Dieser Nachrichtenhandler muss so bearbeitet werden, dass er den passenden JVM-Server angibt, bevor die Konfiguration verwendet werden kann.

jsonjavaprovider.xml

Eine Service-Provider-Pipelinedefinition, die das JSON-Nachrichtenformat für eine Pipeline verwendet, die Java unterstützt. Die Pipeline verwendet den Nachrichtenhandler `<cics_json_handler_java>` und wird verwendet, wenn die CICS-Anwendung mit dem CICS-JSON-Assistenten implementiert wurde. Diese Konfiguration enthält das Element `<jvmserver>`. Dieser Nachrichtenhandler muss so bearbeitet werden, dass er den passenden JVM-Server angibt, bevor die Konfiguration verwendet werden kann.

jsonzosconnectprovider.xml

Eine Pipelinedefinition für einen JSON-Web-Service, der in einer PIPELINE implementiert ist, die für z/OS Connect for CICS konfiguriert ist. Die Pipeline verwendet den Nachrichtenhandler `<provider_pipeline_json>`. Diese Konfiguration enthält das Element `<jvmserver>`. Dieser Nachrichtenhandler muss so bearbeitet werden, dass er den passenden JVM-Server angibt, bevor die Konfiguration verwendet werden kann.

kerberosprovider.xml

Eine Service-Provider-Pipelinedefinition, die Konfigurationsinformationen für die Kerberos-Unterstützung zu `basicsoap11provider.xml` hinzufügt.

samlprovider.xml

Eine Service-Provider-Pipelinedefinition, die Konfigurationsinformationen für die SAML-Unterstützung zu `basicsoap11provider.xml` hinzufügt.

samlrequester.xml

Eine Service-Requester-Pipelinedefinition, die Konfigurationsinformationen für die SAML-Unterstützung zu `basicsoap11requester.xml` hinzufügt.

propagatesamlprovider.xml

Eine Service-Provider-Pipelinedefinition, die Konfigurationsinformationen für die SAML-Unterstützung mit Weitergabe von SAML-Informationen über eine CICS-Transaktion zu `basicsoap11provider.xml` hinzufügt.

propagatesamlrequester.xml

Eine Service-Requester-Pipelinedefinition, die Konfigurationsinformationen für die SAML-Unterstützung mit Weitergabe von SAML-Informationen über eine CICS-Transaktion zu `basicsoap11requester.xml` hinzufügt.

wsatprovider.xml

Eine Pipelinedefinition, die Konfigurationsinformationen für Web-Service-Transaktionen zu `basicsoap11provider.xml` hinzufügt.

wsatrequester.xml

Eine Pipelinedefinition, die Konfigurationsinformationen für Web-Service-Transaktionen zu `basicsoap11requester.xml` hinzufügt.

Beispiel für eine Provider-Pipelinekonfigurationsdatei (JSON-Anwendungshandler)

Dies ist ein einfaches Beispiel für eine Konfigurationsdatei für eine Service-Provider-Pipeline, die das Element `<cics_json_handler_java>` verwendet:

```
<?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>
```

Die Pipeline enthält nur einen Nachrichtenhandler. Der Handler stellt eine Verknüpfung zum Programm DFHJSON her.

- Das Element `<provider_pipeline>` ist das Stammelement der Pipelinekonfigurationsdatei für eine Service-Provider-Pipeline.
- Das Element `<service>` gibt die Nachrichtenhandler an, die für die einzelnen Anforderungen aufgerufen werden. In diesem Beispiel gibt es nur einen Nachrichtenhandler.
- Das Element `<terminal_handler>` enthält die Definition des Terminal-Nachrichtenhändlers der Pipeline.
- Das Element `<cics_json_handler_java>` gibt an, dass die Pipeline eine Java-basierte Pipeline und der Service-Handler der Pipeline ein Nachrichtenhandler ist, der JSON-Nachrichten unterstützt.

- Das Element <apphandler> gibt den Namen des Anwendungshandlers an, mit dem der Terminal-Handler standardmäßig eine Verknüpfung herstellt. In diesem Fall heißt das Programm DFHJSON. Dabei handelt es sich um das von CICS bereitgestellte Programm für Anwendungen, die mit dem CICS-JSON-Assistenten implementiert werden.

Beispiel für eine Provider-Pipelinekonfigurationsdatei (über einen Kanal angehängter Anwendungshandler)

Dies ist ein einfaches Beispiel für eine Konfigurationsdatei für eine Service-Provider-Pipeline, die das Element <cics_soap_1.1_handler> verwendet:

```
<?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>
```

Die Pipeline enthält nur einen Nachrichtenhandler. Der Handler stellt eine Verknüpfung zum Programm DFHPITP her.

- Das Element <provider_pipeline> ist das Stammelement der Pipelinekonfigurationsdatei für eine Service-Provider-Pipeline.
- Das Element <service> gibt die Nachrichtenhandler an, die für die einzelnen Anforderungen aufgerufen werden. In diesem Beispiel gibt es nur einen Nachrichtenhandler.
- Das Element <terminal_handler> enthält die Definition des Terminal-Nachrichtenhändlers der Pipeline.
- Das Element <cics_soap_1.1_handler> gibt an, dass die Pipeline keine Java-basierte Pipeline ist und dass der Terminal-Handler der Pipeline ein Nachrichtenhandler ist, der SOAP 1.1-Nachrichten unterstützt.
- Das Element <apphandler> gibt den Namen des Anwendungshandlers an, mit dem der Terminal-Handler standardmäßig eine Verknüpfung herstellt. In diesem Fall heißt das Programm DFHPITP. Dabei handelt es sich um das von CICS bereitgestellte Programm für Anwendungen, die mit dem CICS-Web-Service-Assistenten implementiert werden.

Beispiel für eine Provider-Pipelinekonfigurationsdatei (Axis2-Anwendungshandler)

Dies ist ein einfaches Beispiel für eine Konfigurationsdatei für eine Service-Provider-Pipeline, die das Element <cics_soap_1.1_handler_java> verwendet:

```
<?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>
```

Die Pipeline enthält nur einen Nachrichtenhandler. Der Handler stellt eine Verknüpfung zum Programm DFHPITP her.

- Das Element `<provider_pipeline>` ist das Stammelement der Pipelinekonfigurationsdatei für eine Service-Provider-Pipeline.
- Das Element `<service>` gibt die Nachrichtenhandler an, die für die einzelnen Anforderungen aufgerufen werden. In diesem Beispiel gibt es nur einen Nachrichtenhandler.
- Das Element `<terminal_handler>` enthält die Definition des Terminal-Nachrichtenhandlers der Pipeline.
- Das Element `<cics_soap_1.1_handler_java>` gibt an, dass die Pipeline eine Java-basierte Pipeline ist und dass der Service-Handler der Pipeline ein Nachrichtenhandler ist, der SOAP 1.1-Nachrichten unterstützt.
- Das Element `<apphandler_class>` gibt den bereitgestellten Axis2-Anwendungshandler an.

Beispiel für eine Requester-Pipelinekonfigurationsdatei

Dies ist ein einfaches Beispiel für eine Konfigurationsdatei für eine Service-Requester-Pipeline, die das Element `<cics_soap_1.2_handler_java>` mit Axis2-MTOM/XOP-Unterstützung verwendet:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<requester_pipeline
  xmlns="http://www.ibm.com/software/htp/cics/pipeline">
  <service>
    <service_handler_list>
      <cics_soap_1.2_handler_java>
        <jvmserver>JVMSESV1</jvmserver>
        <mtom>
      </cics_soap_1.2_handler_java>
    </service_handler_list>
  </service>
</requester_pipeline>
```

Die Pipeline enthält nur einen Nachrichtenhandler.

- Das Element `<requester_pipeline>` ist das Stammelement der Pipelinekonfigurationsdatei für eine Service-Requester-Pipeline.
- Das Element `<service>` gibt die Nachrichtenhandler an, die für die einzelnen Anforderungen aufgerufen werden. In diesem Beispiel gibt es nur einen Nachrichtenhandler.
- `<service_handler_list>` gibt eine Liste von Nachrichtenhandlern an, die für die einzelnen Anforderungen aufgerufen werden.
- Das Element `<cics_soap_1.2_handler_java>` gibt an, dass die Pipeline Java unterstützt und dass der Service-Handler der Pipeline ein Nachrichtenhandler ist, der SOAP 1.2-Nachrichten unterstützt.
- Das Element `<jvmserver>` gibt den zu verwendenden JVM-Server an.
- Das Element `<mtom/>` gibt an, dass ausgehende XOP-Dokumente in MTOM-Nachrichten gepackt und gesendet werden. Standardmäßig werden eingehende MTOM-Nachrichten für Java-basierte Pipelines akzeptiert und entpackt.

Beispiel für eine Provider-Pipelinekonfigurationsdatei für einen z/OS Connect for CICS-JSON-Web-Service

Dies ist ein einfaches Beispiel für eine Konfigurationsdatei für eine Service-Provider-Pipeline, die das Element `<provider_pipeline_json>` verwendet. Da ein Attribut `java_parser="NO"` bereitgestellt wird, verwendet sie einen Nicht-Java-JSON-Parser:

```
<?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>
```

Das Element `<provider_pipeline_json>` unterscheidet sich vom Element `<provider_pipeline>` insofern, als dass keine Handlerprogramme definiert werden können.

- Das Element `<provider_pipeline_json>` ist das Stammelement der Pipelinekonfigurationsdatei für eine JSON-Web-Service-Provider-Pipeline von z/OS Connect for CICS.
- Das Attribut `java_parser="NO"` gibt an, dass der Nicht-Java-JSON-Parser verwendet wird.
- Das Element `<jvmserver>` gibt den zu verwendenden JVM-Server an.

Anmerkung: Der Versuch, eine `<provider_pipeline_json>`-Pipeline auf andere Weise zu starten als mit z/OS Connect for CICS, führt zu einem Fehler.

Beispiel für eine Provider-Pipelinekonfigurationsdatei für Nicht-Java-JSON-Parsing

```
<?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>
```

Die Pipeline enthält nur einen Nachrichtenhandler.

- Das Element `<provider_pipeline>` ist das Stammelement der Pipelinekonfigurationsdatei für eine Service-Provider-Pipeline.
- Das Element `<service>` gibt die Nachrichtenhandler an, die für die einzelnen Anforderungen aufgerufen werden. In diesem Beispiel gibt es nur einen Nachrichtenhandler.
- Das Element `<terminal_handler>` enthält die Definition des Terminal-Nachrichtenhändlers der Pipeline.
- Das Element `<handler>` gibt Details zum Handler an.
- Das Element `<program>` gibt das aufzurufende Programm an. DFHPIJT ist der von CICS bereitgestellte Handler für Nicht-Java-JSON-Verarbeitung.

Transportbezogene Handler

In der Konfigurationsdatei für die einzelnen Pipelines können Sie mehr als einen Satz von Nachrichtenhandlern angeben. Zur Laufzeit wählt CICS basierend auf den Ressourcen, die für den Nachrichtentransport verwendet werden, die Nachrichtenhandler aus, die aufgerufen werden.

In einem Service-Provider und einem Service-Requester können Sie angeben, dass manche Nachrichtenhandler nur aufgerufen werden sollen, wenn ein bestimmter Transport (HTTP oder WebSphere MQ) verwendet wird. Nehmen wir beispielsweise an, dass Sie für Ihre Mitarbeiter einen Web-Service verfügbar machen. Mitarbeiter, die an einer Unternehmenslokation arbeiten, greifen auf den Service mithilfe von WebSphere MQ-Transport in einem sicheren internen Netz zu. Aber Mitarbeiter, die an einer Lokation eines Geschäftspartners arbeiten, greifen mithilfe des HT-

TP-Transports über das Internet auf den Service zu. In dieser Situation ist es empfehlenswert, aufgrund der Vertraulichkeit der Informationen Nachrichtenhandler einzusetzen, um Teile der Nachricht zu verschlüsseln, wenn der HTTP-Transport verwendet wird.

In einem Service-Provider werden eingehende Nachrichten einer benannten Ressource zugeordnet (TCPIPService für den HTTP-Transport, QUEUE für den MQ-Transport). Sie können angeben, dass manche Nachrichtenhandler nur dann aufgerufen werden sollen, wenn eine bestimmte Ressource für eine eingehende Anforderung verwendet wird.

Dafür werden die Nachrichtenhandler in zwei unterschiedlichen Teilen der Pipelinekonfigurationsdatei angegeben:

Serviceabschnitt

Gibt die Nachrichtenhandler an, die jedes Mal aufgerufen werden, wenn die Pipeline ausgeführt wird.

Transportabschnitt

Gibt die Nachrichtenhandler an, die unter Umständen aufgerufen werden, abhängig von den verwendeten Transportressourcen.

Remember: Zur Laufzeit kann ein Nachrichtenhandler die Ausführung der Pipeline begrenzen. Selbst wenn CICS basierend auf dem Inhalt in der Pipelinekonfigurationsdatei entscheidet, dass ein bestimmter Nachrichtenhandler aufgerufen werden soll, kann diese Entscheidung also von einem früheren Nachrichtenhandler außer Kraft gesetzt werden.

Die Nachrichtenhandler, die im Transportabschnitt angegeben sind (die *transportbezogenen Handler*), werden in unterschiedlichen Listen organisiert. Zur Laufzeit wählt CICS die Handler in einer der beiden Listen zur Ausführung aus, abhängig davon, welche Transportressourcen verwendet werden. Wenn mehr als eine Liste mit den verwendeten Transportressourcen übereinstimmt, verwendet CICS die selektivste Liste. Die Listen, die sowohl in Service-Provider- als auch in Service-Requester-Pipelines verwendet werden, sind folgende:

<default_transport_handler_list>

Dies ist die am wenigsten selektive Liste von transportbezogenen Handlern. Die in dieser Liste angegebenen Handler werden aufgerufen, wenn keine der folgenden Listen mit den verwendeten Transportressourcen übereinstimmt.

<default_http_transport_handler_list>

In einer Service-Requester-Pipeline werden die Handler in dieser Liste aufgerufen, wenn der HTTP-Transport verwendet wird.

In einer Service-Provider-Pipeline werden die Handler in dieser Liste aufgerufen, wenn der HTTP-Transport verwendet wird, und kein Element `<named_transport_entry>` TCPIPService für die TCP/IP-Verbindung nennt.

<default_mq_transport_handler_list>

In einer Service-Requester-Pipeline werden die Handler in dieser Liste aufgerufen, wenn der WebSphere MQ-Transport verwendet wird.

In einer Service-Provider-Pipeline werden die Handler in dieser Liste aufgerufen, wenn der WebSphere MQ-Transport verwendet wird, und kein `<named_transport_entry>`-Element die Nachrichtenwarteschlange nennt, in der eingehende Nachrichten empfangen werden.

Die folgende Liste von Nachrichtenhndlern wird nur in der Konfigurationsdatei für eine Service-Provider-Pipeline verwendet:

<named_transport_entry>

Neben einer Liste von Hndlern gibt <named_transport_entry> den Namen einer Ressource und den Transporttyp an.

- Für den HTTP-Transport werden die Handler in dieser Liste aufgerufen, wenn der Ressourcename mit dem Namen von TCIPSERVICE für die eingehende TCP/IP-Verbindung übereinstimmt.
- Für den WebSphere MQ-Transport werden die Handler in dieser Liste aufgerufen, wenn der Ressourcename mit dem Namen der Nachrichtenwarteschlange übereinstimmt, die die eingehende Nachricht empfängt.

Beispiel

Dies ist ein Beispiel für ein Element <transport> aus der Pipelinekonfigurationsdatei für eine Service-Provider-Pipeline:

```
<transport>

  <!-- HANDLER1 und HANDLER2 sind die Standardtransporthandler -->
  <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 überschreibt die Standardeinstellungen für MQ-Transport -->
  <default_mq_transport_handler_list>
    <handler><program>HANDLER3</program><handler_parameter_list/></handler>
  </default_mq_transport_handler_list>

  <!-- HANDLER4 überschreibt die Standardwerte für HTTP-Transport mit TCIPSERVICE(WS00) -->
  <named_transport_entry type="http">
    <name>WS00</name>
    <transport_handler_list>
      <handler><program>HANDLER4</program><handler_parameter_list/></handler>
    </transport_handler_list>
  </named_transport_entry>

</transport>
```

Die Auswirkungen dieser Definition sind folgende:

- <default_mq_transport_handler_list> stellt sicher, dass Nachrichten, die MQ-Transport verwenden, von dem Handler HANDLER3 verarbeitet werden.
- <named_transport_entry> stellt sicher, dass Nachrichten, die die TCP/IP-Verbindung verwenden, die TCIPSERVICE(WS00) zugeordnet ist, vom Handler HANDLER4 verarbeitet werden.
- <default_transport_handler_list> stellt sicher, dass alle verbleibenden Nachrichten, d. h. solche, die den HTTP-Transport verwenden, aber nicht TCIPSERVICE(WS00), von den Hndlern HANDLER1 und HANDLER2 verarbeitet werden.

Remember: Alle im Serviceabschnitt der Pipelinedefinition angegebenen Handler werden zusätzlich zu solchen aufgerufen, die im Transportabschnitt angegeben sind.

Pipelinedefinition für einen Service-Provider

Die Nachrichtenhandler sind in einem XML-Dokument definiert, das unter z/OS UNIX gespeichert ist. Der Name der Datei, die das Dokument enthält, wird im Attribut CFGFILE einer PIPELINE-Definition angegeben.

Das Stammelement des Pipelinekonfigurationsdokuments ist das Element `<provider_pipeline>`. Die übergeordnete Struktur des Dokuments ist in Abb. 20 auf Seite 105 dargestellt.

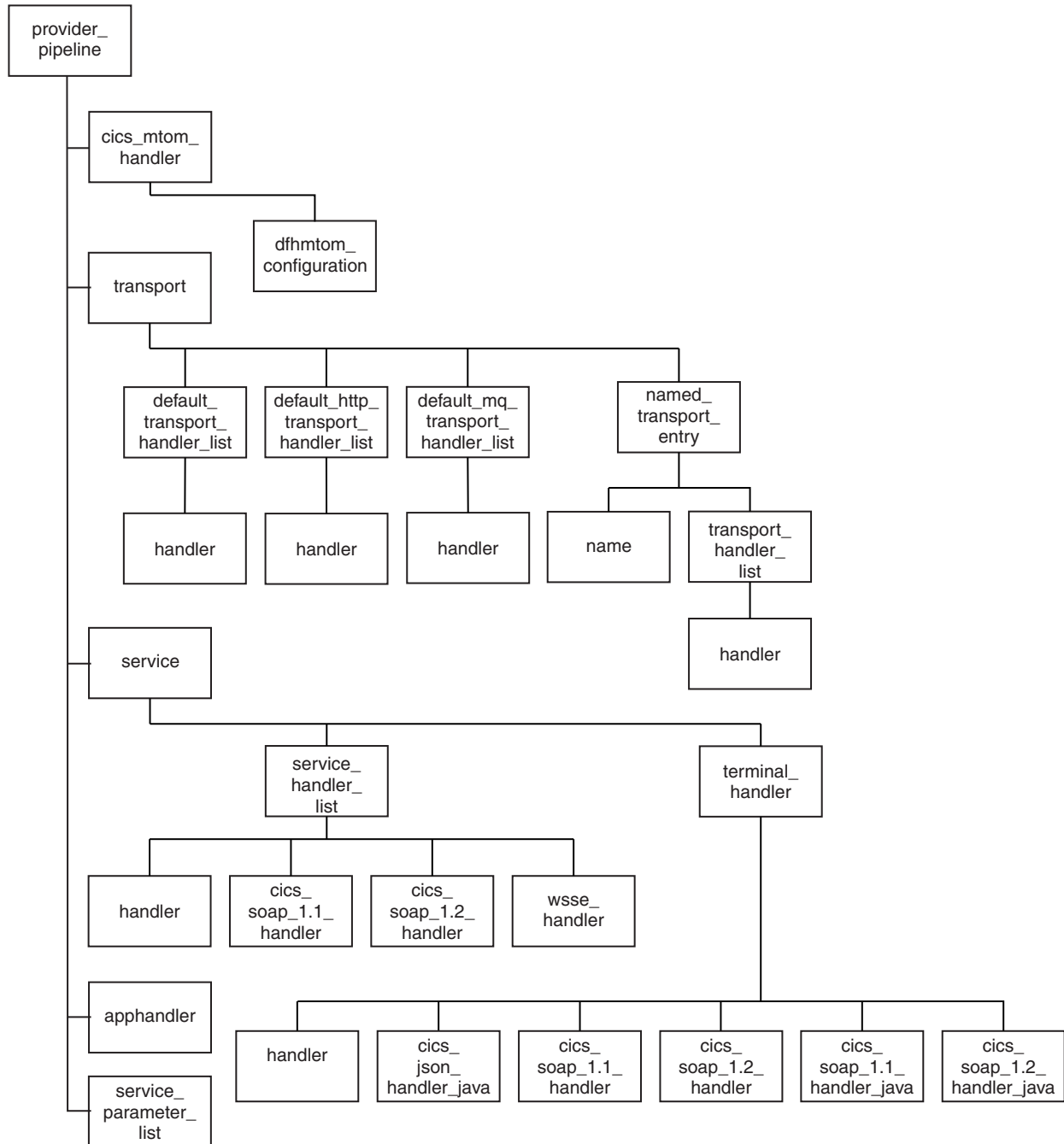


Abbildung 20. Struktur der Pipelinedefinition für einen Service-Provider.

Anmerkung: Zu Vereinfachung der Abbildung sind die untergeordneten Elemente der Elemente <handler>, <cics_json_handler_java>, <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> und <cics_soap_1.2_handler_java> nicht dargestellt.

Pipelinedefinition für einen Service-Requester

Die Nachrichtenhändler sind in einem XML-Dokument definiert, das unter z/OS UNIX gespeichert ist. Der Name der Datei, die das Dokument enthält, wird im Attribut CFGFILE einer PIPELINE-Definition angegeben.

Das Stammelement des Pipelinekonfigurationsdokuments ist das Element `<requester_pipeline>`. Die übergeordnete Struktur des Dokuments ist in Abb. 21 dargestellt.

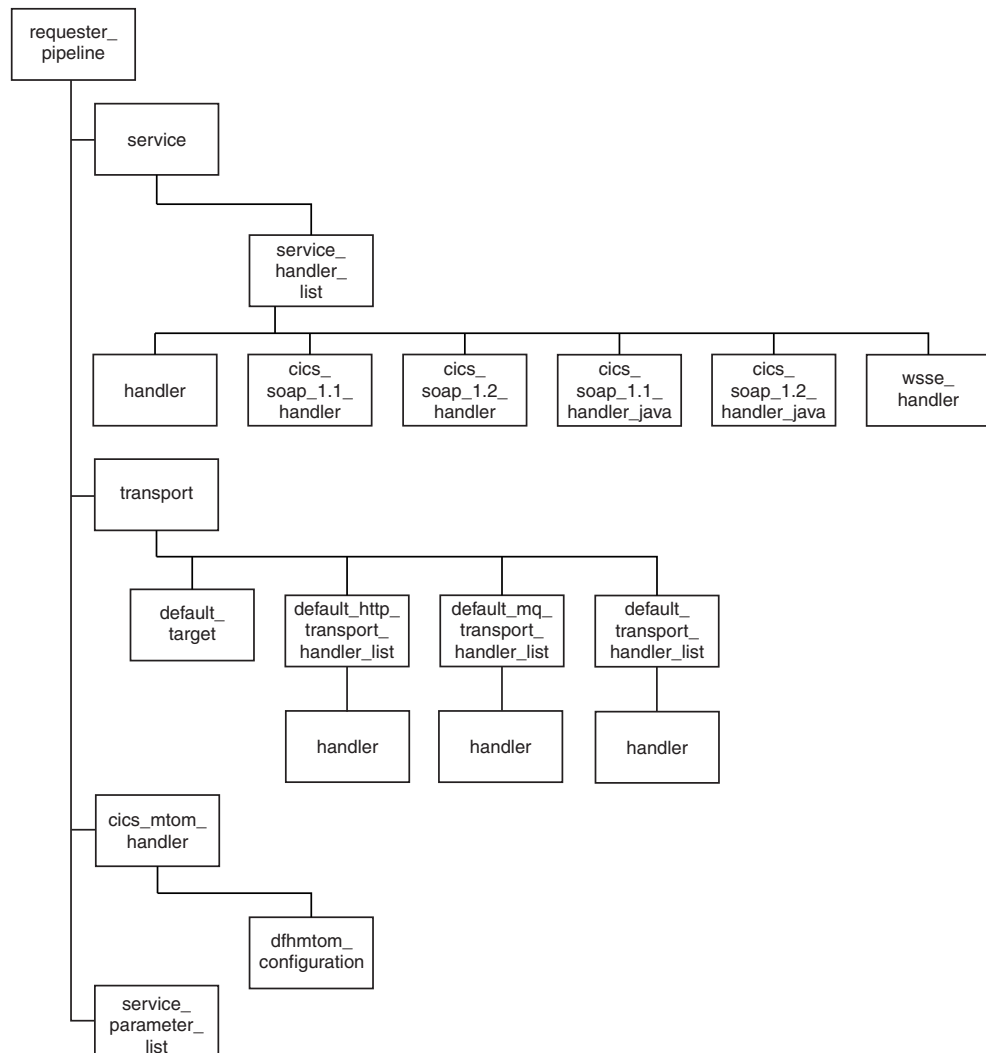


Abbildung 21. Struktur der Pipelinedefinition für einen Service-Requester.

Anmerkung: Zur Vereinfachung der Abbildung sind die untergeordneten Elemente der Elemente `<handler>`, `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` und `<cics_soap_1.2_handler_java>` nicht dargestellt.

Elemente, die nur in Service-Providern verwendet werden

Einige XML-Elemente in einer Pipelinekonfigurationsdatei können nur in Service-Provider-Pipelines verwendet werden.

Anwendungshandler:

Ein Anwendungshandler ist ein CICS-Programm, mit dem der Terminal-Handler einer SOAP-Service-Provider-Pipeline zur Laufzeit verknüpft wird.

Anwendungshandler werden in Pipelines im Providermodus verwendet, in denen der Terminal-Handler einer der bereitgestellten SOAP-Nachrichtenhandler ist. Dies

ist der Fall, wenn das `<terminal_handler>`-Element ein `<cics_soap_1.1_handler>`-, `<cics_soap_1.2_handler>`-, `<cics_soap_1.1_handler_java>`- oder `<cics_soap_1.2_handler_java>`-Element enthält.

Der Anwendungshandler ist für die Verarbeitung des Hauptteils einer SOAP-Anforderung und für die Generierung einer Antwort unter Verwendung der zurückgegebenen Daten verantwortlich. Der Anwendungshandler kann andere Programme aufrufen, um diese Verarbeitung abzuschließen. Typischerweise agiert der Anwendungshandler als vielseitig einsetzbare Darstellungsebene für eine oder mehrere Geschäftsanwendungen. Er ist verantwortlich für das Zuordnen von XML zu einem Formular, das eine Anwendung verwenden kann, das Anhängen dieser Anwendung und anschließend das Generieren einer Antwort unter Verwendung der zurückgegebenen Daten.

Ein Anwendungshandler kann auf zwei Arten von CICS angehängt werden. Die typische Vorgehensweise bezieht einen Kanal und Steuercontainer ein, die andere Methode verwendet Java-Bindungen für Axis2.

Über einen Kanal angehängte Anwendungshandler werden im `<apphandler>`-Element des `<provider_pipeline>`-Elements angegeben. Zur Laufzeit wird der Container DFHWS-APPHANDLER mit den Inhalten von `<apphandler>` gefüllt. Allerdings kann der Container DFHWS-APPHANDLER dynamisch von einem beliebigen anderen Nachrichtenhandler aktualisiert werden. Deshalb kann sich das Programm, das zur Laufzeit verknüpft wird, von dem im `<apphandler>`-Element angegebenen Programm unterscheiden. Die folgenden Anwendungshandler können im `<apphandler>`-Element oder im Container DFHWS-APPHANDLER angegeben werden:

- Der bereitgestellte, über einen Kanal angehängte SOAP-Anwendungshandler, DFHPITP. Weitere Informationen zu über einen Kanal angehängten Anwendungshandlern finden Sie unter „Über einen Kanal angehängte Anwendungshandler“ auf Seite 150.
- Ihr eigener, über einen Kanal angehängter Anwendungshandler. Dieser Anwendungshandler kann in anderen Sprachen als Java geschrieben werden. Weitere Informationen zu den Steuercontainern, die in Ihrem, über einen Kanal angehängten Anwendungshandler verwendet werden können, finden Sie unter „Steuercontainer“ auf Seite 166.
- Ihr eigener Java-Anwendungshandler für Java-basierte Pipelines, der die Java-Schnittstelle 'ApplicationHandler' implementiert und mithilfe von Axis2 MessageContext an die Pipeline angehängt wird. Weitere Informationen zur Java-Schnittstelle 'ApplicationHandler' finden Sie unter Interface ApplicationHandler.

Um einen Anwendungshandler mit Java-Bindungen für Axis2 zu verwenden, müssen Sie das `<apphandler_class>`-Element des `<provider_pipeline>`-Elements angeben. Axis2-Anwendungshandler erfordern auch, dass ein JVM-Server vorhanden ist, auf dem die Web-Service-Pipeline und der Anwendungshandler ausgeführt werden können, und dass der Terminal-Handler Ihrer Web-Service-Pipeline entweder der Nachrichtenhandler `<cics_soap_1.1_handler_java>` oder der Nachrichtenhandler `<cics_soap_1.2_handler_java>` ist. Um den bereitgestellten Axis2-Anwendungshandler zu verwenden, müssen Sie `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` in dem `<apphandler_class>`-Element angeben. Sie können jedoch auch Ihre eigene Axis2-Anwendungshandlerklasse angeben. Zur Laufzeit wird der Container DFHWS-APPHANCLAS mit den Inhalten von `<apphandler_class>` gefüllt.

Für Web-Service-Anwendungen, die mithilfe des CICS-Web-Service-Assistenten implementiert werden, müssen Sie entweder DFHPITP oder Ihren eigenen Anwen-

dungshandler angeben, der DFHPITP im <apphandler>-Element verwendet. Oder Sie geben `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` im <apphandler_class>-Element an. Weitere Informationen zum CICS-Web-Service-Assistenten finden Sie unter The CICS web services assistant.

Es ist auch möglich, Axis2-Anwendungen als Web-Services im Providermodus in CICS zu implementieren. Verwenden Sie dazu den Axis2-Stil der Web-Service-Implementierung. Weitere Informationen finden Sie unter Deploying a Java provider-mode web service in an Axis2 JVM server.

Pipelinekonfigurationselement <apphandler_class>:

Gibt an, dass der Terminal-Handler der Pipeline eine Verknüpfung zu einem Axis2-Anwendungshandler einrichtet.

Das <apphandler_class>-Element wird verwendet, um einen Axis2-Anwendungshandler anzugeben, wenn Ihr <terminal_handler>-Element ein <cics_json_handler_java>-, <cics_soap_1.1_handler_java>- oder <cics_soap_1.2_handler_java>-Element enthält. Um den bereitgestellten Axis2-Anwendungshandler zu verwenden, geben Sie `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` im <apphandler_class>-Element an. Wenn Sie die CICS-SOAP-Handler verwenden, können Sie auch Ihre eigene Axis2-Anwendungshandlerklasse angeben.

Alternativ können Sie das <apphandler>-Element in Ihrer Pipelinekonfigurationsdatei angeben, wenn Sie einen über einen Kanal angehängten Anwendungshandler verwenden möchten. Weitere Informationen finden Sie unter dem <apphandler>-Element. Sie dürfen jedoch nicht <apphandler_class>- und <apphandler>-Elemente in derselben Pipelinekonfigurationsdatei angeben.

Anmerkung: Sie dürfen das <apphandler>-Element nicht mit dem <cics_json_handler_java>-Element verwenden.

Sie dürfen das <apphandler_class>-Element nicht verwenden, wenn Ihr <terminal_handler>-Element entweder ein <cics_soap_1.1_handler>- oder ein <cics_soap_1.2_handler>-Element enthält.

Weitere Informationen zu Anwendungshandlern finden Sie unter „Anwendungshandler“ auf Seite 106.

Verwendet in:

- Service-Provider

Enthalten in:

- <provider_pipeline>-Element

Beispiel

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

Pipelinekonfigurationselement <named_transport_entry>:

Enthält eine Liste mit Handlern, die aufgerufen werden sollen, wenn eine benannte Transportressource von einem Service-Provider verwendet wird.

- Für den WebSphere MQ-Transport ist die benannte Ressource die lokale Eingabewarteschlange, in der die Anforderung empfangen wird.

- Für den HTTP-Transport ist die Ressource der TCP/IP-Service (TCPIPService), der den Port definiert, an dem die Anforderung empfangen wurde.

Verwendet in:

- Service-Provider

Enthalten in:

<transport>

Attribute:

Name	Beschreibung
type	Der Transportmechanismus, dem die benannte Ressource zugeordnet wird: wmq Die benannte Ressource ist eine Warteschlange. http Die benannte Ressource ist ein TCP/IP-Service.

Enthält:

1. Ein Element <name>, das den Namen der Ressource enthält.
2. Ein optionales Element <transport_handler_list>. Jedes Element <transport_handler_list> enthält mindestens ein Element <handler>.
Wenn Sie kein Element <transport_handler_list> codieren, sind die einzigen Nachrichtenhandler, die aufgerufen werden, wenn der benannte Transport verwendet wird, diejenigen, die im Element <service> angegeben sind.

Beispiel

```
<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>
```

In diesem Beispiel werden die angegebenen Nachrichtenhandler (HANDLER1 und HANDLER2) für Nachrichten aufgerufen, die im TCP/IP-Service namens PORT80 empfangen werden.

Pipelinekonfigurationselement <provider_pipeline>:

Gibt das Stammelement des XML-Dokuments an, das die Konfiguration der CICS-Pipeline für einen Web-Service-Provider beschreibt.

Verwendet in:

- Service-Provider

Enthält:

1. Optionales Element <cics_mtom_handler>
2. Optionales Element <transport>
3. Element <service>
4. Optionales Element <apphandler>
5. Optionales Element <apphandler_class>

6. Optionales Element `<service_parameter_list>`, das XML-Elemente enthält, die allen Nachrichtenhndlern in der Pipeline im Container DFH-SERVICEPLIST verfügbar gemacht werden

Beispiel

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

Pipelinekonfigurationselement `<provider_pipeline_json>`:

Gibt das Stammelement des XML-Dokuments an, das die Konfiguration der CICS-Pipeline eines JSON-Web-Service-Providers für z/OS Connect beschreibt.

Dieses Element unterscheidet sich vom Element `<provider_pipeline>` insofern, als dass keine Handlerprogramme definiert werden können. Diese Art von Pipeline dient als Container für die **WEBSERVICE**-Ressourcen, die von z/OS Connect verwendet werden. Der Versuch, eine Pipeline `<provider_pipeline_json>` auf andere Weise zu starten als mit z/OS Connect führt zu einem Fehler. Die resultierende **PIPELINE**-Ressource kann nicht als Ziel einer **USAGE(PIPELINE) URIMAP**-Ressource verwendet werden. Sie kann nur mit **USAGE(JVMSEVER) URIMAP**-Ressourcen verwendet werden.

Verwendet in:

- Service-Provider

Attribute:

java_parser=**{yes|no}**

Wählen Sie den Typ des JSON-Parsers aus, der für die Verarbeitung eingehender Nachrichten verwendet wird.

Mögliche Werte sind:

- | | |
|------------|---|
| yes | JSON-Parsing mithilfe von Java im JVM-Server. Dies ist die Standardeinstellung. |
| no | Nicht-Java-Parsing der JSON-Nachricht. |

Anmerkung: Das Attribut `java_parser` ist optional. Wenn Sie es nicht angeben, ist das Standardverhalten, die JSON-Nachricht mithilfe von Java im JVM-Server zu parsen. Dies entspricht dem Verhalten, wenn Sie `java_parser="yes"` angeben.

java_generator=**{yes|no}**

Auswählen des Typs von JSON-Generators, der zum Generieren von ausgehenden Nachrichten verwendet wird.

Mögliche Werte sind:

- | | |
|------------|--|
| yes | Durchführen der JSON-Generierung mithilfe von Java im JVM-Server. |
| no | Durchführen einer Nicht-Java-Generierung der JSON-Nachricht. Dies ist die Standardeinstellung. |

Enthält:

- Ein Element `<jvmserver>`, das den Namen der JVMSEVER-Ressource enthält, in der z/OS Connect konfiguriert wird.

Beispiel für Java-Parsing

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

Beispiel für Nicht-Java-Parsing

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

Pipelinekonfigurationselement **<terminal_handler>**:

Enthält die Definition des Terminal-Nachrichtenhandlers der Service-Provider-Pipeline.

Verwendet in:

- Service-Provider

Enthalten in:

- Element **<service>**

Enthält:

Eines der folgenden Elemente:

```
<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>
```

Wenn Ihre Pipeline sowohl SOAP 1.1- als auch SOAP 1.2-Nachrichten verarbeiten soll, müssen Sie entweder **<cics_soap_1.2_handler>** oder **<cics_soap_1.2_handler_java>** verwenden.

Remember: In einem Service-Provider können Sie die Elemente **<cics_soap_1.1_handler>** und **<cics_soap_1.2_handler>** im Element **<service_handler_list>** und im Element **<terminal_handler>** angeben. In einem Service-Provider können Sie jedoch nur die Elemente **<cics_soap_1.1_handler_java>** und **<cics_soap_1.2_handler_java>** im Element **<terminal_handler>** angeben.

Beispiel

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

Beispiel: Nicht-Java-Verarbeitung von JSON-Nachrichten aktivieren

Um die Nicht-Java-Verarbeitung von JSON-Nachrichten zu aktivieren, geben Sie das Terminal-Handlerprogramm als DFHPIJT an:

```

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

```

Anmerkung: Wenn Sie DFHPIJT als Terminal-Handler verwenden, definieren Sie keinen Anwendungshandler in der Pipelinekonfigurationsdatei, d. h. die Pipelinekonfigurationsdatei sollte kein <apphandler>-Element enthalten. Wenn ein Anwendungshandler angegeben ist, wird er nicht aufgerufen.

Pipelinekonfigurationselement <transport_handler_list>:

Enthält eine Liste von Nachrichtenhandlern, die aufgerufen werden, wenn eine benannte Ressource verwendet wird.

- Für den MQ-Transport ist die benannte Ressource der Name der lokalen Eingabewarteschlange.
- Für den HTTP-Transport ist die Ressource der TCP/IP-Service (TCPIPSERVICE), der den Port definiert, an dem die Anforderung empfangen wurde.

Verwendet in:

- Service-Provider

Enthalten in:

- Element <named_transport_entry>

Enthält:

- Ein oder mehrere <handler>-Elemente

Beispiel

```

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

```

In Service-Requestern verwendete Elemente

Einige XML-Elemente in einer Pipelinekonfigurationsdatei können nur in Service-Requester-Pipelines verwendet werden.

Konfigurationselement <requester_pipeline>:

Das Stammelement des XML-Dokuments, das die Konfiguration einer Pipeline in einem Service-Requester beschreibt.

Verwendet in:

- Service-Requester

Enthält:

1. Optionales Element <service>
2. Optionales Element <transport>
3. Optionales Element <cics_mtom_handler>

- Optionales Element `<service_parameter_list>`, das XML-Elemente enthält, die den Nachrichtenhandlern im Container DFH-SERVICEPLIST verfügbar gemacht werden

Beispiel

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

In Service-Provider- und Service-Requester-Pipelines verwendete Elemente

Einige XML-Elemente, die in einer Pipelinekonfigurationsdatei verwendet werden, gelten für Service-Provider- und Service-Requester-Pipelines.

Pipelinekonfigurationselement `<addressing>`:

Gibt die Unterstützung für Web-Service-Adressierung in einer Java-basierten SOAP-Verarbeitung an.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

Element `<cics_soap_1.1_handler_java>`

Element `<cics_soap_1.2_handler_java>`

Enthält:

Ein Element `<namespace>`. In einem Service-Provider ist dieses Element optional. Das Element enthält eines der beiden WS-Addressing-Schemas, die von CICS unterstützt werden. Für eingehende Nachrichten unterstützt Axis2 beide Spezifikationen. Für ausgehende Nachrichten wird der in diesem Element angegebene Namensbereich verwendet. Wenn Sie dieses Element nicht angeben oder zwei Elemente haben, verwendet CICS dieselbe Spezifikation für die ausgehende und die eingehende Nachricht. In einem Service-Requester ist dieses Element erforderlich und Sie können nur einen Namensbereich für die ausgehende Nachricht angeben.

Dieses Beispiel zeigt die Konfiguration für eine Service-Provider-Pipeline, in der beide WS-Addressing-Spezifikationen unterstützt werden. CICS verwendet dieselbe Spezifikation für die ausgehende und die eingehende Nachricht. Sie können dieselben Ergebnisse abrufen, indem Sie ein leeres `<addressing>`-Element angeben.

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

Pipelinekonfigurationselement `<cics_json_handler_java>`:

Gibt die Attribute des Handlerprogramms für JSON-Nachrichten in Java-basierten JSON-Pipelines an.

Verwendet in:

- Service-Provider

Enthalten in:

The <service_handler_list> element
The <terminal_handler> element

Enthält:

1. Ein Element <jvmserver>.
2. Ein optionales Element <repository>.

Beispiel

Das folgende Beispiel zeigt die XML für den Java-basierten JSON-Handler und ihre verschachtelten Elemente:

```
<?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/cicsts55/lib/pipeline/repository</repository>
      </cics_json_handler_java>
    </terminal_handler>
  </service>
  <apphandler_class>com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler</apphandler_class>
</provider_pipeline>
```

Pipelinekonfigurationselement <cics_soap_1.1_handle>:

Gibt die Attribute des Handlerprogramms für SOAP 1.1-Nachrichten in Nicht-Java-Pipelines an.

Verwendet in:

- Service-Requester
- Service-Provider

Enthalten in:

<service_handler_list>-Element
<terminal_handler>-Element

Enthält:

Kein, ein oder mehrere <headerprogram>-Elemente Jedes <headerprogram>-Element enthält:

1. Ein <program_name>-Element, das den Namen des Programms zur Headerverarbeitung enthält
2. Ein <namespace>-Element, das mit dem folgenden <localname>-Element verwendet wird, um zu bestimmen, welche Headerblöcke in einer SOAP-Nachricht von dem Programm zur Headerverarbeitung verarbeitet werden sollen. Das <namespace>-Element enthält den URI (Uniform Resource Identifier) des Namensbereichs des Headerblocks.
3. Ein <localname>-Element, das mit dem führenden <namespace>-Element verwendet wird, um zu bestimmen, welche Headerblöcke in einer SOAP-Nachricht von dem Programm zur Headerverarbeitung verarbeitet werden sollen. Das <localname>-Element enthält den Elementnamen des Headerblocks.

Sehen Sie sich beispielsweise diesen Headerblock an:

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

- Der Name des Namensbereichs ist `http://mynamespace`.
- Der Elementname ist `myheaderblock`.

Damit das Headerprogramm mit diesem Headerblock übereinstimmt, codieren Sie die Elemente `<namespace>` und `<localname>` wie folgt:

```
<namespace>http://mynamespace</namespace>  
<localname>myheaderblock</localname>
```

Sie können einen Stern (*) in dem Element `<localname>` codieren, um anzugeben, dass alle Headerblöcke in dem Namensbereich, deren Namen mit einer gegebenen Zeichenfolge beginnen, verarbeitet werden sollen. Beispiel:

```
<namespace>http://mynamespace</namespace>  
<localname>myhead*</localname>
```

Wenn Sie den Stern im Element `<localname>` verwenden, kann ein Header in einer Nachricht mit mehr als einem `<headerprogram>`-Element übereinstimmen. Sehen Sie sich beispielsweise diesen Headerblock an:

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

stimmt mit allen folgenden `<headerprogram>`-Elementen überein:

```
<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>
```

Wenn dies der Fall ist, wird das Headerprogramm ausgeführt, das im `<headerprogram>`-Element angegeben ist, in dem der Elementname des Headerblocks sehr exakt angegeben ist. In diesem Beispiel ist dies HDRPROG3.

Wenn die SOAP-Nachricht mehr als einen Header enthält, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen, aber die Reihenfolge, in der die Header verarbeitet werden, ist nicht definiert.

Wenn Sie zwei oder mehr `<headerprogram>`-Elemente codieren, die dieselben Elemente `<namespace>` und `<localname>` enthalten, aber unterschiedliche Headerprogramme angeben, wird nur eines der Headerprogramme ausgeführt. Welches ausgeführt wird, ist jedoch nicht definiert.

4. Ein `<mandatory>`-Element, das einen booleschen XML-Wert (`true` oder `false`) enthält. Alternativ können Sie die Werte als 1 bzw. 0 codieren.

true

Während der Serviceanforderungsverarbeitung in einer Service-Provider-Pipeline und während der Serviceantwortverarbeitung in einer Service-Requester-Pipeline muss das Programm zur Headerverarbeitung mindestens

einmal aufgerufen werden, selbst wenn keiner der Header in den SOAP-Nachrichten mit den Elementen `<namespace>` und `<localname>` übereinstimmt:

- Wenn keiner der Header übereinstimmt, wird das Programm zur Headerverarbeitung einmal aufgerufen.
- Wenn Header übereinstimmen, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen.

Während der Serviceanforderungsverarbeitung in einer Service-Requester-Pipeline und während der Serviceantwortverarbeitung in einer Service-Provider-Pipeline wird das Programm zur Headerverarbeitung mindestens einmal aufgerufen, auch wenn die SOAP-Nachricht, die CICS erstellt, anfänglich keine Header enthält. Wenn Sie Ihrer Nachricht Header hinzufügen möchten, müssen Sie sicherstellen, dass mindestens ein Programm zur Headerverarbeitung aufgerufen wird, indem Sie `<mandatory>true</mandatory>` oder `<mandatory>1</mandatory>` angeben.

false

Das Programm zur Headerverarbeitung wird nur aufgerufen, wenn ein oder mehrere Header in den SOAP-Nachrichten mit den Elementen `<namespace>` und `<localname>` übereinstimmen:

- Wenn keiner der Header übereinstimmt, wird das Programm zur Headerverarbeitung nicht aufgerufen.
- Wenn Header übereinstimmen, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen.

Beispiel

```
<cics_soap_1.1_handler>
  <headerprogram>
    <program_name> ... </program_name>
    <namespace>...</namespace>
    <localname>...</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.1_handler>
```

Pipelinekonfigurationselement `<cics_soap_1.1_handler_java>`:

Gibt die Attribute des Handlerprogramms für SOAP 1.1-Nachrichten in Java-basierten SOAP-Pipelines an.

Verwendet in:

- Service-Requester
- Service-Provider

Enthalten in:

`<service_handler_list>`-Element
`<terminal_handler>`-Element

Enthält:

1. Ein Element `<jvmserver>`.
2. Ein optionales Element `<repository>`.
3. Ein optionales Element `<addressing>`. Wenn Sie Web-Service-Adressierung in Axis2 aktivieren, verwenden Sie nicht das Programm zur Headerverarbeitung von DFHWSADH.

4. Kein, ein oder mehrere <headerprogram>-Elemente. Jedes <headerprogram>-Element enthält:
 - a. Ein <program_name>-Element, das den Namen des Programms zur Headerverarbeitung enthält. Sie können Axis2-Handler in Java schreiben, um die SOAP-Header zu verarbeiten.
 - b. Ein <namespace>-Element, das mit dem folgenden <localname>-Element verwendet wird, um zu bestimmen, welche Headerblöcke in einer SOAP-Nachricht von dem Programm zur Headerverarbeitung verarbeitet werden sollen. Das <namespace>-Element enthält den URI (Uniform Resource Identifier) des Namensbereichs des Headerblocks.
 - c. Ein <localname>-Element, das mit dem führenden <namespace>-Element verwendet wird, um zu bestimmen, welche Headerblöcke in einer SOAP-Nachricht von dem Programm zur Headerverarbeitung verarbeitet werden sollen. Das <localname>-Element enthält den Elementnamen des Headerblocks.

Sehen Sie sich beispielsweise diesen Headerblock an:

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

Der Name des Namensbereichs ist http://mynamespace und der Elementname ist myheaderblock.

Damit das Headerprogramm mit diesem Headerblock übereinstimmt, codieren Sie die Elemente <namespace> und <localname> wie folgt:

```
<namespace>http://mynamespace</namespace>
<localname>myheaderblock</localname>
```

Sie können einen Stern (*) in dem Element <localname> codieren, um anzugeben, dass alle Headerblöcke in dem Namensbereich, deren Namen mit einer gegebenen Zeichenfolge beginnen, verarbeitet werden sollen. Beispiel:

```
<namespace>http://mynamespace</namespace>
<localname>myhead*</localname>
```

Wenn Sie den Stern im Element <localname> verwenden, kann ein Header in einer Nachricht mit mehr als einem <headerprogram>-Element übereinstimmen. Sehen Sie sich beispielsweise diesen Headerblock an:

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

stimmt mit allen folgenden <headerprogram>-Elementen überein:

```
<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>
```

Wenn dies der Fall ist, wird das Headerprogramm ausgeführt, das im <headerprogram>-Element angegeben ist, in dem der Elementname des Headerblocks sehr exakt angegeben ist. In diesem Beispiel ist dies HDRPROG3.

Wenn die SOAP-Nachricht mehr als einen Header enthält, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen, aber die Reihenfolge, in der die Header verarbeitet werden, ist nicht definiert.

Wenn Sie zwei oder mehr <headerprogram>-Elemente codieren, die dieselben Elemente <namespace> und <localname> enthalten, aber unterschiedliche Headerprogramme angeben, wird nur eines der Headerprogramme ausgeführt. Welches ausgeführt wird, ist jedoch nicht definiert.

- d. Ein <mandatory>-Element, das einen booleschen XML-Wert (true oder false) enthält. Alternativ können Sie die Werte als 1 bzw. 0 codieren.

true

Während der Serviceanforderungsverarbeitung in einer Service-Provider-Pipeline und während der Serviceantwortverarbeitung in einer Service-Requester-Pipeline muss das Programm zur Headerverarbeitung mindestens einmal aufgerufen werden, selbst wenn keiner der Header in den SOAP-Nachrichten mit den Elementen <namespace> und <localname> übereinstimmt:

- Wenn keiner der Header übereinstimmt, wird das Programm zur Headerverarbeitung einmal aufgerufen.
- Wenn Header übereinstimmen, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen.

Während der Serviceanforderungsverarbeitung in einer Service-Requester-Pipeline und während der Serviceantwortverarbeitung in einer Service-Provider-Pipeline wird das Programm zur Headerverarbeitung mindestens einmal aufgerufen, auch wenn die SOAP-Nachricht, die CICS erstellt, anfänglich keine Header enthält. Wenn Sie Ihrer Nachricht Header hinzufügen möchten, müssen Sie sicherstellen, dass mindestens ein Programm zur Headerverarbeitung aufgerufen wird, indem Sie <mandatory>true</mandatory> oder <mandatory>1</mandatory> angeben.

false

Das Programm zur Headerverarbeitung wird nur aufgerufen, wenn ein oder mehrere Header in den SOAP-Nachrichten mit den Elementen <namespace> und <localname> übereinstimmen:

- Wenn keiner der Header übereinstimmt, wird das Programm zur Headerverarbeitung nicht aufgerufen.
- Wenn Header übereinstimmen, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen.

Beispiel

Das folgende Beispiel zeigt die XML für den Java-basierten SOAP-Handler und ihre verschachtelten Elemente:

```
<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>
```

Pipelinekonfigurationselement `<cics_soap_1.2_handler>`:

Gibt die Attribute des Handlerprogramms für SOAP 1.2-Nachrichten in Nicht-Java-Pipelines an.

Verwendet in:

- Service-Requester
- Service-Provider

Enthalten in:

`<service_handler_list>`-Element
`<terminal_handler>`-Element

Enthält:

Kein, ein oder mehrere `<headerprogram>`-Elemente Jedes `<headerprogram>`-Element enthält:

1. Ein `<program_name>`-Element, das den Namen des Programms zur Headerverarbeitung enthält
2. Ein `<namespace>`-Element, das mit dem folgenden `<localname>`-Element verwendet wird, um zu bestimmen, welche Headerblöcke in einer SOAP-Nachricht von dem Programm zur Headerverarbeitung verarbeitet werden sollen. Das `<namespace>`-Element enthält den URI (Uniform Resource Identifier) des Namensbereichs des Headerblocks.
3. Ein `<localname>`-Element, das mit dem führenden `<namespace>`-Element verwendet wird, um zu bestimmen, welche Headerblöcke in einer SOAP-Nachricht von dem Programm zur Headerverarbeitung verarbeitet werden sollen. Das `<localname>`-Element enthält den Elementnamen des Headerblocks.

Sehen Sie sich beispielsweise diesen Headerblock an:

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

- Der Name des Namensbereichs ist `http://mynamespace`.
- Der Elementname ist `myheaderblock`.

Damit das Headerprogramm mit diesem Headerblock übereinstimmt, codieren Sie die Elemente `<namespace>` und `<localname>` wie folgt:

```
<namespace>http://mynamespace</namespace>  
<localname>myheaderblock</localname>
```

Sie können einen Stern (*) in dem Element `<localname>` codieren, um anzugeben, dass alle Headerblöcke in dem Namensbereich, deren Namen mit einer gegebenen Zeichenfolge beginnen, verarbeitet werden sollen. Beispiel:

```
<namespace>http://mynamespace</namespace>  
<localname>myhead*</localname>
```

Wenn Sie den Stern im Element `<localname>` verwenden, kann ein Header in einer Nachricht mit mehr als einem `<headerprogram>`-Element übereinstimmen. Sehen Sie sich beispielsweise diesen Headerblock an:

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

stimmt mit allen folgenden `<headerprogram>`-Elementen überein:

```
<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>

```

Wenn dies der Fall ist, wird das Headerprogramm ausgeführt, das im `<headerprogram>`-Element angegeben ist, in dem der Elementname des Headerblocks sehr exakt angegeben ist. In diesem Beispiel ist dies HDRPROG3.

Wenn die SOAP-Nachricht mehr als einen Header enthält, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen, aber die Reihenfolge, in der die Header verarbeitet werden, ist nicht definiert.

Wenn Sie zwei oder mehr `<headerprogram>`-Elemente codieren, die dieselben Elemente `<namespace>` und `<localname>` enthalten, aber unterschiedliche Headerprogramme angeben, wird nur eines der Headerprogramme ausgeführt. Welches ausgeführt wird, ist jedoch nicht definiert.

4. Ein `<mandatory>`-Element, das einen booleschen XML-Wert (true oder false) enthält. Alternativ können Sie die Werte als 1 bzw. 0 codieren.

true

Während der Serviceanforderungsverarbeitung in einer Service-Provider-Pipeline und während der Serviceantwortverarbeitung in einer Service-Requester-Pipeline muss das Programm zur Headerverarbeitung mindestens einmal aufgerufen werden, selbst wenn keiner der Header in den SOAP-Nachrichten mit den Elementen `<namespace>` und `<localname>` übereinstimmt:

- Wenn keiner der Header übereinstimmt, wird das Programm zur Headerverarbeitung einmal aufgerufen.
- Wenn Header übereinstimmen, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen.

Während der Serviceanforderungsverarbeitung in einer Service-Requester-Pipeline und während der Serviceantwortverarbeitung in einer Service-Provider-Pipeline wird das Programm zur Headerverarbeitung mindestens einmal aufgerufen, auch wenn die SOAP-Nachricht, die CICS erstellt, anfänglich keine Header enthält. Wenn Sie Ihrer Nachricht Header hinzufügen möchten, müssen Sie sicherstellen, dass mindestens ein Programm zur Headerverarbeitung aufgerufen wird, indem Sie `<mandatory>true</mandatory>` oder `<mandatory>1</mandatory>` angeben.

false

Das Programm zur Headerverarbeitung wird nur aufgerufen, wenn ein oder mehrere Header in den SOAP-Nachrichten mit den Elementen `<namespace>` und `<localname>` übereinstimmen:

- Wenn keiner der Header übereinstimmt, wird das Programm zur Headerverarbeitung nicht aufgerufen.
- Wenn Header übereinstimmen, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen.

Beispiel

```
<cics_soap_1.2_handler>
  <headerprogram>
    <program_name> ... </program_name>
    <namespace>...</namespace>
    <localname>...</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler>
```

Pipelinekonfigurationselement **<cics_soap_1.2_handler_java>**:

Gibt die Attribute des Handlerprogramms für SOAP 1.2-Nachrichten in Java-basierten SOAP-Pipelines an.

Verwendet in:

- Service-Requester
- Service-Provider

Enthalten in:

<service_handler_list>-Element
<terminal_handler>-Element

Enthält:

1. Ein Element <jvmserver>.
2. Ein optionales Element <repository>.
3. Ein optionales Element <addressing>. Wenn Sie Unterstützung für Web-Service-Adressierung in Axis2 aktivieren, verwenden Sie nicht die Programme zur Headerverarbeitung. Sie können Axis2-Handler in Java schreiben, um die SOAP-Header zu verarbeiten.
4. Kein, ein oder mehrere <headerprogram>-Elemente. Jedes <headerprogram>-Element enthält:
 - a. Ein <program_name>-Element, das den Namen des Programms zur Headerverarbeitung enthält
 - b. Ein <namespace>-Element, das mit dem folgenden <localname>-Element verwendet wird, um zu bestimmen, welche Headerblöcke in einer SOAP-Nachricht von dem Programm zur Headerverarbeitung verarbeitet werden sollen. Das <namespace>-Element enthält den URI (Uniform Resource Identifier) des Namensbereichs des Headerblocks.
 - c. Ein <localname>-Element, das mit dem führenden <namespace>-Element verwendet wird, um zu bestimmen, welche Headerblöcke in einer SOAP-Nachricht von dem Programm zur Headerverarbeitung verarbeitet werden sollen. Das <localname>-Element enthält den Elementnamen des Headerblocks.

Sehen Sie sich beispielsweise diesen Headerblock an:

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

Der Name des Namensbereichs ist http://mynamespace und der Elementname ist myheaderblock.

Damit das Headerprogramm mit diesem Headerblock übereinstimmt, codieren Sie die Elemente <namespace> und <localname> wie folgt:

```
<namespace>http://mynamespace</namespace>
<localname>myheaderblock</localname>
```

Sie können einen Stern (*) in dem Element `<localname>` codieren, um anzugeben, dass alle Headerblöcke in dem Namensbereich, deren Namen mit einer gegebenen Zeichenfolge beginnen, verarbeitet werden sollen. Beispiel:

```
<namespace>http://mynamespace</namespace>
<localname>myhead*</localname>
```

Wenn Sie den Stern im Element `<localname>` verwenden, kann ein Header in einer Nachricht mit mehr als einem `<headerprogram>`-Element übereinstimmen. Sehen Sie sich beispielsweise diesen Headerblock an:

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

stimmt mit allen folgenden `<headerprogram>`-Elementen überein:

```
<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>
```

Wenn dies der Fall ist, wird das Headerprogramm ausgeführt, das im `<headerprogram>`-Element angegeben ist, in dem der Elementname des Headerblocks sehr exakt angegeben ist. In diesem Beispiel ist dies HDRPROG3.

Wenn die SOAP-Nachricht mehr als einen Header enthält, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen, aber die Reihenfolge, in der die Header verarbeitet werden, ist nicht definiert.

Wenn Sie zwei oder mehr `<headerprogram>`-Elemente codieren, die dieselben Elemente `<namespace>` und `<localname>` enthalten, aber unterschiedliche Headerprogramme angeben, wird nur eines der Headerprogramme ausgeführt. Welches ausgeführt wird, ist jedoch nicht definiert.

- d. Ein `<mandatory>`-Element, das einen booleschen XML-Wert (true oder false) enthält. Alternativ können Sie die Werte als 1 bzw. 0 codieren.

true

Während der Serviceanforderungsverarbeitung in einer Service-Provider-Pipeline und während der Serviceantwortverarbeitung in einer Service-Requester-Pipeline muss das Programm zur Headerverarbeitung mindestens einmal aufgerufen werden, selbst wenn keiner der Header in den SOAP-Nachrichten mit den Elementen `<namespace>` und `<localname>` übereinstimmt:

- Wenn keiner der Header übereinstimmt, wird das Programm zur Headerverarbeitung einmal aufgerufen.
- Wenn Header übereinstimmen, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen.

Während der Serviceanforderungsverarbeitung in einer Service-Requester-Pipeline und während der Serviceantwortverarbeitung in einer Service-Provider-Pipeline wird das Programm zur Headerverarbeitung mindestens einmal aufgerufen, auch wenn die SOAP-Nachricht, die CICS erstellt, anfänglich keine Header enthält. Wenn Sie Ihrer Nachricht Header hinzufügen möchten, müssen Sie sicherstellen, dass mindestens ein Programm zur Headerverarbeitung aufgerufen wird, indem Sie `<mandatory>true</mandatory>` oder `<mandatory>1</mandatory>` angeben.

false

Das Programm zur Headerverarbeitung wird nur aufgerufen, wenn ein oder mehrere Header in den SOAP-Nachrichten mit den Elementen `<namespace>` und `<localname>` übereinstimmen:

- Wenn keiner der Header übereinstimmt, wird das Programm zur Headerverarbeitung nicht aufgerufen.
- Wenn Header übereinstimmen, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen.

Beispiel

Das folgende Beispiel zeigt die XML für den Java-basierten SOAP-Handler und ihre verschachtelten Elemente:

```
<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>
```

Pipelinekonfigurationselement `<default_http_transport_handler_list>`:

Gibt die Nachrichtenhandler an, die standardmäßig aufgerufen werden, wenn der HTTP-Transport verwendet wird.

In einem Service-Provider werden in dieser Liste angegebene Nachrichtenhandler nur aufgerufen, wenn die Liste mit den Handlern, die in einem `<named_transport_entry>`-Element definiert sind, weniger spezifisch ist.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

- `<transport>`-Element

Enthält:

- Ein oder mehrere `<handler>`-Elemente

Beispiel

```
<default_http_transport_handler_list>
  <handler>
    ...
  </handler>
```

```

    <handler>
    ...
  </handler>
</default_http_transport_handler_list>

```

Pipelinekonfigurationselement <default_mq_transport_handler_list>:

Gibt die Nachrichtenhandler an, die standardmäßig aufgerufen werden, wenn der WebSphere MQ-Transport verwendet wird.

In einem Service-Provider werden in dieser Liste angegebene Nachrichtenhandler nur aufgerufen, wenn die Liste mit den Handlern, die in einem <named_transport_entry>-Element definiert sind, weniger spezifisch ist.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

- <transport>-Element

Enthält:

- Ein oder mehrere <handler>-Elemente

Beispiel

```

<default_mq_transport_handler_list>
  <handler>
  ...
  </handler>
  <handler>
  ...
  </handler>
</default_mq_transport_handler_list>

```

Pipelinekonfigurationselement <default_transport_handler_list>:

Gibt die Nachrichtenhandler an, die standardmäßig aufgerufen werden, wenn ein beliebiger Transport verwendet wird.

In einem Service-Provider werden in dieser Liste angegebene Nachrichtenhandler aufgerufen, wenn die Liste mit den Handlern, die in einem der folgenden Elemente definiert sind, weniger spezifisch ist:

```

  <default_http_transport_handler_list>
  <default_mq_transport_handler_list>
  <named_transport_entry>

```

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

- <transport>-Element

Enthält:

- Ein oder mehrere <handler>-Elemente

Beispiel

```
<default_transport_handler_list>
  <handler>
    <program>HANDLER1</program>
    <handler_parameter_list/>
  </handler>
  <handler>
    <program>HANDLER2</program>
    <handler_parameter_list/>
  </handler>
</default_transport_handler_list>
```

Pipelinekonfigurationselement <handler>:

Gibt die Attribute eines Nachrichtenhandlerprogramms an.

Einige von CICS bereitgestellte Handlerprogramme setzen das Element <handler> nicht ein. Die von CICS bereitgestellten SOAP-Nachrichtenhandlerprogramme werden beispielsweise mithilfe der Elemente <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> und <cics_soap_1.2_handler_java> definiert.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

```
<default_transport_handler_list>
<transport_handler_list>
<service_handler_list>
<terminal_handler>
<default_http_transport_handler_list>
<default_mq_transport_handler_list>
```

Enthält:

1. Ein Element <program>, das den Namen des Handlerprogramms enthält.
2. Ein Element <handler_parameter_list>, das XML-Elemente enthält, die Nachrichtenhandlern im Container DFH-HANDLERPLIST zur Verfügung gestellt werden.

Beispiel

```
<?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>

```

In diesem Beispiel ist das Handlerprogramm MYPROG. Die Liste der Handlerparameter besteht aus einem einzelnen <output>-Element. Der Inhalt der Parameterliste ist MYPROG bekannt.

Pipelinekonfigurationselement <jvmserver>:

Gibt den Namen der JVMSERVER-Ressource an.

Dieses Element gibt den Namen der JVMSERVER-Ressource an, die die Anforderung verarbeiten wird. Wenn kein Wert angegeben wird, wird eine Fehlermeldung generiert, und die PIPELINE wird im Status DISABLED installiert.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

- The <cics_json_handler_java> element
- The <cics_soap_1.1_handler_java> element
- The <cics_soap_1.2_handler_java> element
- The <provider_pipeline_json> element

Beispiel

```
<jvmserver>JVMSEVER_NAME</jvmserver>
```

Pipelinekonfigurationselement <repository>:

Gibt den Verzeichnisnamen des Axis2-Repositorys an.

Dieses optionale Element gibt den Verzeichnisnamen des Axis2-Repositorys an. Wenn Sie diese Option verwenden, müssen Sie The <jvmserver> element zuvor in der Handler-XML angeben. Wenn das Element zu diesem Zeitpunkt nicht angegeben wird, wird das Beispielverzeichnis verwendet. Bei der Installation von CICS Transaction Server wird das Axis2-Beispielverzeichnis im Verzeichnis /usr/lpp/cicsts/cicsts55/lib/pipeline/repository installiert, wobei /usr/lpp/cicsts/cicsts55 das Standardinstallationsverzeichnis für CICS-Dateien unter z/OS UNIX ist.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

- The <cics_json_handler_java> element
- The <cics_soap_1.1_handler_java> element
- The <cics_soap_1.2_handler_java> element

Beispiel

```
<cics_soap_1.1_handler_java>  
<jvmserver>JVMSERV1</jvmserver>  
<repository>/lib/pipeline/repository</repository>  
</cics_soap_1.1_handler_java>
```

Pipelinekonfigurationselement <service>:

Gibt die Nachrichtenhandler an, die für die einzelnen Anforderungen aufgerufen werden.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

```
<provider_pipeline>  
<requester_pipeline>
```

Enthält:

1. Element <service_handler_list>
2. Element <terminal_handler> (nur in einem Service-Provider)

Beispiel

```
<service>  
  <service_handler_list>  
    ...  
  </service_handler_list>  
  <terminal_handler>  
    ...  
  </terminal_handler>  
</service>
```

Pipelinekonfigurationselement <service_handler_list>:

Gibt eine Liste von Nachrichtenhandlern an, die für die einzelnen Anforderungen aufgerufen werden.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

- <service>-Element

Enthält:

Mindestens eines der folgenden Elemente:

```
<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>
```

Sie bestimmen die Reihenfolge, in der die einzelnen Handler zur Laufzeit aufgerufen werden, indem Sie die Reihenfolge der Handlerelemente im Element `<service_handler_list>` angeben. Beispiel: Wenn Ihre Pipeline WS-Security unterstützt, bleiben verschlüsselte SOAP-Nachrichten verschlüsselt, bis das Element `<wsse_handler>` aufgerufen wird. Deshalb müssen Sie das Element `<wsse_handler>` vor jedem anderen Handlerprogramm angeben, das nicht verschlüsselte Nachrichten verarbeitet.

Das Element `<service_handler_list>` für einen Service-Provider darf nicht die Elemente `<cics_soap_1.1_handler_java>` und `<cics_soap_1.2_handler_java>` enthalten, da diese Elemente im Element `<terminal_handler>` für Java-basierte Pipelines angegeben werden müssen. Ein Service-Requester kann die Elemente `<cics_soap_1.1_handler_java>` und `<cics_soap_1.2_handler_java>` enthalten. Aber wenn diese verwendet werden, müssen sie im Element `<service_handler_list>` zuerst aufgelistet sein.

Wenn Ihre Pipeline sowohl SOAP 1.1- als auch SOAP 1.2-Nachrichten verarbeiten soll, müssen Sie entweder `<cics_soap_1.2_handler>` oder `<cics_soap_1.2_handler_java>` verwenden.

Sie können entweder einen SOAP 1.1- oder einen SOAP 1.2-Handler in einer Service-Requester-Pipeline verwenden, aber in diesem Fall unterstützt der SOAP 1.2-Handler keine SOAP 1.1-Nachrichten. Geben Sie den SOAP 1.1- oder SOAP 1.2-Handler nicht in der Pipeline an, wenn Ihre Service-Requester-Anwendungen vollständige SOAP-Envelopes im Container DFHREQUEST senden. Auf diese Weise vermeiden Sie doppelte SOAP-Nachrichtenheader in ausgehenden Nachrichten.

In einem Service-Provider können Sie den generischen Handler und die SOAP-Handler im Element `<terminal_handler>` und im Element `<service_handler_list>` angeben. Weitere Informationen zur Verarbeitung von SOAP-Headern finden Sie unter „Programme zur Headerverarbeitung“ auf Seite 160.

Beispiel

```
<service_handler_list>
  <wsse_handler>
    ...
  </wsse_handler>
  <cics_soap_1.1_handler_java>
    ...
  </cics_soap_1.1_handler_java>
  <handler>
    ...
  </handler>
</service_handler_list>
```

Pipelinekonfigurationselement `<service_parameter_list>`:

Gibt die XML-Elemente an, die für alle Nachrichtenhandler in der Pipeline im Container DFH-SERVICEPLIST verfügbar gemacht werden. Dies ist ein optionales Element.

Verwendet in:

- Service-Requester
- Service-Provider

Enthält:

- Wenn Sie WS-AT verwenden: ein Element `<registration_service_endpoint>`

- Wenn Sie in einem Service-Requester WS-AT verwenden: ein optionales Element `<new_tx_context_required/>`
- Optionale benutzerdefinierte Tags

Beispiel

```
<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>
```

Pipelinekonfigurationselement `<transport>`:

Gibt Handler an, die nur aufgerufen werden, wenn ein bestimmter Transport verwendet wird.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

```
<provider_pipeline>
<requester_pipeline>
```

Enthält:

In einem Service-Provider:

1. Ein optionales Element `<default_transport_handler_list>`
2. Ein optionales Element `<default_http_transport_handler_list>`
3. Ein optionales Element `<default_mq_transport_handler_list>`
4. Kein, ein oder mehrere `<named_transport_entry>`-Elemente

In einem Service-Requester:

1. Ein optionales Element `<default_target>`. `<default_target>` enthält einen URI, den CICS verwendet, um den Ziel-Web-Service zu lokalisieren, wenn die Service-Requester-Anwendung keinen URI bereitstellt. In vielen Fällen wird der URI des Ziels jedoch von der Service-Requester-Anwendung bereitgestellt und Ihre Angabe in `<default_target>` wird ignoriert. Die Service-Provider-Anwendungen, die mithilfe des CICS-Web-Service-Assistenten implementiert werden, rufen den URI jedoch in der Regel aus der Web-Service-Beschreibung ab.
2. Ein optionales Element `<default_http_transport_handler_list>`
3. Ein optionales Element `<default_mq_transport_handler_list>`
4. Ein optionales Element `<default_transport_handler_list>`

Beispiel

```
<transport>
  <default_transport_handler_list>
    ...
  </default_transport_handler_list>
</transport>
```

Pipelinekonfiguration für MTOM/XOP

CICS-SOAP-Pipelines können die Spezifikationen für Message Transmission Optimization Mechanism (MTOM) und XML-binary Optimized Packaging (XOP) unterstützen. Diese Spezifikationen definieren einen Mechanismus für das Senden und Empfangen von binären Anhängen mithilfe von SOAP, ohne den zusätzlichen Aufwand einer base64-Codierung. Für die Aktivierung der MTOM-Unterstützung müssen Sie Ihre Pipelines entsprechend konfigurieren.

Pipelinekonfigurationselement <mtom>:

Aktiviert MTOM/XOP-Unterstützung für Java-basierte Pipelines. Wenn dieses Element in Ihrer Pipelinekonfigurationsdatei definiert ist, wird MTOM-Unterstützung für alle eingehenden und ausgehenden Nachrichten aktiviert. Ist dieses Element jedoch nicht in der Pipelinekonfigurationsdatei angegeben, wird MTOM-Unterstützung nur für eingehende Nachrichten aktiviert.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

```
<cics_soap_1.1_handler_java>
<cics_soap_1.2_handler_java>
```

Sowohl in Provider- als auch in Requester-Pipeline-Konfigurationsdateien sollte das Element <mtom> nach dem optionalen Element <addressing> und vor dem optionalen Element <headerprogram> definiert sein.

Beispiel

Für eine Pipeline im Provider- oder Requestermodus können Sie Folgendes angeben:

```
<cics_soap_1.2_handler_java>
  <jvmserver>JVMSERV1</jvmserver>
  <addressing></addressing>
  <mtom></mtom>
  <headerprogram>
    <program_name>HDRPROG4</program_name>
    <namespace>http://mynamespace</namespace>
    <localname>myheaderblock</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler_java>
```

Pipelinekonfigurationselement <cics_mtom_handler>:

Aktiviert das bereitgestellte MTOM-Handlerprogramm für SOAP-Pipelines. Dieses Programm bietet Unterstützung für MTOM-MIME-Multipart/Related-Nachrichten, die XOP-Dokumente und binäre Anhänge enthalten. MTOM-Unterstützung wird

für alle eingehenden Nachrichten aktiviert, die in der Pipeline empfangen werden. Die MTOM-Unterstützung für ausgehende Nachrichten ist jedoch abhängig von weiteren Optionen nur bedingt aktiviert.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

```
<provider_pipeline>
<requester_pipeline>
```

In einer Provider-Pipeline-Konfigurationsdatei muss das `<cics_mtom_handler>`-Element vor dem `<transport>`-Element definiert werden. Zur Laufzeit muss das MTOM-Handlerprogramm die eingehende MTOM-Nachricht entpacken, bevor andere Handler, einschließlich des Transporthandlers, sie verarbeiten. Anschließend wird dieser als der letzte Handler für die Antwortnachricht aufgerufen, um eine MTOM-Nachricht zu packen und an den Web-Service-Requester zu senden.

In einer Requester-Pipeline-Konfigurationsdatei muss das `<cics_mtom_handler>`-Element nach dem `<transport>`-Element definiert werden. Zur Laufzeit wird die ausgehende Anforderungsnachricht erst in das MTOM-Format konvertiert, nachdem sie von allen anderen Handlern verarbeitet wurde. Anschließend wird der Transporthandler als erster Handler für die eingehende Antwortnachricht aufgerufen, um die MTOM-Nachricht zu entpacken, bevor andere Handler sie verarbeiten und an das anfordernde Programm zurückkehren.

Anmerkung: Sie dürfen dieses Handlerprogramm nicht mit Java-basierten Pipelines verwenden. Geben Sie für Java-basierte Pipelines das `<mtom>`-Element an.

Enthält:

```
<dfhmtom_configuration>-Element
```

Standardoptionen können mithilfe der Konfigurationsoptionen geändert werden, die im `<dfhmtom_configuration>`-Element angegeben sind. Wenn Sie die Standardoptionen nicht ändern möchten, können Sie ein leeres Element verwenden.

Beispiel

Für eine Pipeline im Providermodus können Sie Folgendes angeben:

```
<provider_pipeline>
  <cics_mtom_handler></cics_mtom_handler>
  <transport>
    ....
  </transport>
  <service>
    ....
  </service>
</provider_pipeline>
```

Pipelinekonfigurationselement `<dfhmtom_configuration>`:

Gibt die Konfigurationsinformationen für das bereitgestellte MTOM-Handlerprogramm für Pipelines an, das keine Unterstützung für Java bietet. Dieses Programm

bietet Unterstützung für MIME-Nachrichten, die XOP-Dokumente und binäre Anhänge enthalten. Wenn Sie keine Konfiguration für MTOM angeben, übernimmt CICS die Standardwerte.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

`<cics_mtom_handler>`

Attribute:

Name	Beschreibung
version	Eine ganze Zahl, die die Version der Konfigurationsinformationen angibt. Der einzige gültige Wert ist 1.

Enthält:

- Ein optionales Element `<mtom_options>`.
- Ein optionales Element `<xop_options>`.
- Ein optionales Element `<mime_options>`.

Beispiel

```
<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>
```

Pipelinekonfigurationselement `<mtom_options>`:

Gibt an, wann MTOM für ausgehende SOAP-Nachrichten für Pipelines verwendet werden soll, die Java nicht unterstützen.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

`<dfhmtom_configuration>`

Attribute:

Attribut	Beschreibung
send_mtom	<p>Gibt an, ob MTOM verwendet werden soll, um die ausgehende SOAP-Nachricht in eine MIME-Nachricht zu konvertieren:</p> <p>no MTOM wird nicht für ausgehende SOAP-Nachrichten verwendet.</p> <p>same Im Service-Providermodus wird MTOM für SOAP-Antwortnachrichten verwendet, wenn der Requester MTOM verwendet. Dies ist die Standardeinstellung in einer Service-Provider-Pipeline.</p> <p>Im Service-Requestermodus ist die Angabe dieses Werts gleichbedeutend mit der Angabe von <code>send_mtom="yes"</code>.</p> <p>yes MTOM wird für alle ausgehenden SOAP-Nachrichten verwendet. Dies ist die Standardeinstellung in einer Service-Requester-Pipeline.</p>
send_when_no_xop	<p>Gibt an, ob eine MTOM-Nachricht auch dann gesendet werden soll, wenn in der Nachricht keine binären Anhänge vorhanden sind.</p> <p>no MTOM wird nur verwendet, wenn mit der Nachricht binäre Anhänge gesendet werden.</p> <p>yes MTOM wird für alle ausgehenden SOAP-Nachrichten verwendet, auch wenn keine binären Anhänge in der Nachricht gesendet werden sollen. Dies ist die Standardeinstellung und sie wird primär als Indikator für das empfangende Programm verwendet, dass der Absender MTOM/XOP unterstützt.</p> <p>Dieses Attribut kann mit jedem send_mtom-Attributwert kombiniert werden, hat aber keine Auswirkung, wenn Sie <code>send_mtom="no"</code> angeben.</p>

Beispiel

```
<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>
```

In diesem Provider-Pipeline-Beispiel werden SOAP-Nachrichten nur dann in MTOM-Nachrichten konvertiert, wenn binäre Anhänge mit der Nachricht gesendet werden müssen und wenn der Service-Requester eine MTOM-Nachricht gesendet hat.

Pipelinekonfigurationselement `<xop_options>`:

Gibt an, ob XOP-Verarbeitung im Direktmodus oder im Kompatibilitätsmodus für Pipelines stattfinden kann, die Java nicht unterstützen.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

<dfhmtom_configuration>

Attribute:

Attribut	Beschreibung
apphandler_supports_xop	<p>Im Providermodus gibt dieses Attribut an, ob der Anwendungshandler XOP-Dokumente im Direktmodus verarbeiten kann.</p> <p>no Der Anwendungshandler kann XOP-Dokumente nicht direkt verarbeiten. Dies ist der Standardwert, wenn im Element <apphandler> DFHPITP nicht angegeben ist.</p> <p>Der Kompatibilitätsmodus wird in der Pipeline verwendet, um alle eingehenden oder ausgehenden Nachrichten zu verarbeiten, die im MTOM-Format empfangen oder gesendet werden.</p> <p>yes Der Anwendungshandler kann XOP-Dokumente verarbeiten. Dies ist der Standardwert, wenn im Element <apphandler> DFHPITP angegeben ist.</p> <p>Der Direktmodus wird in der Pipeline verwendet, um alle eingehenden oder ausgehenden Nachrichten zu verarbeiten, die im MTOM-Format empfangen oder gesendet werden. Dies unterliegt zur Laufzeit Einschränkungen. Wenn Sie beispielsweise WS-Security-bezogene Elemente in der Pipelinekonfigurationsdatei angegeben haben, bestimmt der MTOM-Handler, dass die Pipeline den Kompatibilitätsmodus statt des Direktmodus für die Verarbeitung von XOP-Dokumenten verwenden sollte.</p> <p>Im Requestermodus gibt dieses Attribut an, ob Service-Requester-Anwendungen die CICS-Web-Service-Unterstützung verwenden, um XOP-Dokumente im Direktmodus zu erstellen und zu verarbeiten.</p> <p>no Service-Requester-Anwendungen verwenden die CICS-Web-Service-Unterstützung nicht. Geben Sie diesen Wert an, wenn Ihre Requester-Anwendung mit DFHPIRT verknüpft ist, um die Pipeline zu betreiben, und deshalb XOP-Dokumente nicht im Direktmodus erstellen und verarbeiten kann.</p> <p>yes Service-Requester-Anwendungen verwenden die CICS-Web-Service-Unterstützung. Geben Sie diesen Wert an, wenn Ihre Requester-Anwendung den Befehl EXEC CICS INVOKE WEBSERVICE verwenden.</p>

Beispiel

```

<provider_pipeline>
  <cics_mtom_handler>
    <dfhmtom_configuration version="1">
      <xop_options apphandler_supports_xop="no"/>
    </dfhmtom_configuration>
  </cics_mtom_handler>
</provider_pipeline>

```

```

    </dfhmtom_configuration>
  </cics_mtom_handler>
  ...
</provider_pipeline>

```

In diesem Beispiel einer Provider-Pipeline werden eingehende MTOM-Nachrichten und ausgehende Antwortnachrichten in der Pipeline mithilfe des Kompatibilitätsmodus verarbeitet.

Pipelinekonfigurationselement <mime_options>:

Gibt den Domänennamen an, der verwendet werden soll, wenn MIME-'content-ID'-Werte für Pipelines generiert werden, die Java nicht unterstützen. Die MIME-'content-ID'-Werte geben binäre Anhänge an.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

```
<dfhmtom_configuration>
```

Attribute:

Attribut	Beschreibung
content_id_domain	<p>Die korrekte Syntax ist <i>domänenname</i>.</p> <p>Um Internetstandards zu entsprechen, sollte der Name ein gültiger Internet-Host-Name und eindeutig im CICS-System sein, in dem die Pipeline installiert wird. Beachten Sie, dass CICS dies nicht prüft.</p> <p>Wenn dieses Element weggelassen wird, verwendet CICS den Wert <i>cicsts</i>.</p>

Beispiel

```

<provider_pipeline>
  <dfhmtom_configuration version="1">
    <mime_options content_id_domain="example.org"/>
  </dfhmtom_configuration>
  ...
</provider_pipeline>

```

In diesem Beispiel werden Referenzen auf binäre Anhänge unter Verwendung von *cid:eindeutiger_wert@example.org* erstellt.

Pipelinekonfiguration für WS-Security

Damit Web-Service-Requester- und -Provider-Anwendungen in WS-Security-Protokolle einbezogen werden können, müssen Sie Ihre Pipelines entsprechend konfigurieren, z. B. indem Sie den Nachrichtenhandler DFHWSSE einschließen und Konfigurationsinformationen für den Handler bereitstellen.

Example

Eine Provider-Pipelinekonfigurationsdatei, die WS-Security verwendet, kann das folgende Format haben:

```

<?xml version="1.0"?>
<provider_pipeline
  xmlns="http://www.ibm.com/software/htp/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>

```

Pipelinekonfigurationselement <wsse_handler>:

Gibt Parameter an, die von dem von CICS bereitgestellten Nachrichtenhandler verwendet werden, der Unterstützung für WS-Security bietet.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

<service_handler_list>

Enthält:

- The <dfhwsse_configuration>pipeline configuration element

In einer Provider-Pipeline-Konfigurationsdatei muss der von CICS bereitgestellte Nachrichtenhandler für WS-Security möglicherweise eine verschlüsselte Nachricht entschlüsseln. Das Element <wsse_handler> muss vor allen anderen Handlerprogrammen definiert werden, die den verschlüsselten Nachrichteninhalt verarbeiten müssen.

In einer Requester-Pipeline-Konfigurationsdatei muss der von CICS bereitgestellte Nachrichtenhandler für WS-Security möglicherweise eine Nachricht verschlüsseln. Er muss nach allen anderen Handlerprogrammen, die den unverschlüsselten Nachrichteninhalt verarbeiten müssen, einschließlich des CICS-SOAP-Handlerprogramms, definiert werden.

Pipelinekonfigurationselement <dfhwsse_configuration>:

Gibt Konfigurationsinformationen für den Sicherheitshandler DFHWSSE1 an, der Unterstützung für das Sichern von Web-Services bereitstellt.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

<wsse_handler>

Attribute:

Name	Beschreibung
version	Eine ganze Zahl, die die Version der Konfigurationsinformationen angibt. Der einzige gültige Wert ist 1.

Enthält:

1. Eines der folgenden Elemente:

- Ein optionales Element <authentication>.
 - In einer Service-Requester-Pipeline gibt das Element <authentication> den Typ von Authentifizierung an, der im Sicherheitsheader von ausgehenden SOAP-Nachrichten verwendet werden muss.
 - In einer Service-Provider-Pipeline gibt das Element an, ob CICS die Sicherheitstoken in einer eingehenden SOAP-Nachricht verwendet, um die Benutzer-ID zu ermitteln, unter der die Verarbeitung stattfindet.
- Ein optionales Element <sts_authentication>.

Das Attribut action in diesem Element gibt den Typ von Anforderung an, die an den Sicherheitstokenservice gesendet werden soll. Wenn die Anforderung darin besteht, ein Identitätstoken auszugeben, verwendet CICS die Werte in den verschachtelten Elementen, um ein Identitätstoken des angegebenen Typs anzufordern.

2. Wenn Sie ein Element <sts_authentication> angeben, müssen Sie auch ein Element <sts_endpoint> angeben.

Wenn dieses Element vorhanden ist, verwendet CICS den URI im Element <endpoint>, um eine Anforderung an den Sicherheitstokenservice zu senden.

3. Ein optionales leeres Element <expect_signed_body/>.

Das Element <expect_signed_body/> gibt an, dass der Hauptteil (<body>) der eingehenden Nachricht signiert sein muss. Wenn der Hauptteil einer eingehenden Nachricht nicht korrekt signiert ist, weist CICS die Nachricht mit einem Sicherheitsfehler zurück.

4. Ein optionales leeres Element <expect_encrypted_body/>.

Das Element <expect_encrypted_body/> gibt an, dass der Hauptteil (<body>) der eingehenden Nachricht verschlüsselt sein muss. Wenn der Hauptteil einer eingehenden Nachricht nicht korrekt verschlüsselt ist, weist CICS die Nachricht mit einem Sicherheitsfehler zurück.

5. Ein optionales Element <sign_body>.

Wenn dieses Element vorhanden ist, signiert CICS den Hauptteil (<body>) der ausgehenden Nachricht unter Verwendung des im Element <algorithm> angegebenen Algorithmus, das wiederum im Element <sign_body> enthalten ist.

6. Ein optionales Element <encrypt_body>.

Wenn dieses Element vorhanden ist, verschlüsselt CICS den Hauptteil (<body>) der ausgehenden Nachricht unter Verwendung des im Element <algorithm> angegebenen Algorithmus, das wiederum im Element <encrypt_body> enthalten ist.

7. Nur in Provider-Pipelines ein optionales Element <reject_signature/>.

Wenn dieses Element vorhanden ist, weist CICS alle Nachrichten zurück, die ein Zertifikat in ihrem Header enthalten, das einen Teil oder den gesamten Nachrichtenhauptteil signiert. Es wird ein SOAP-Fehler an den Web-Service-Requester ausgegeben.

8. Nur in Provider-Pipelines ein optionales Element `<reject_encryption/>`.

Wenn dieses Element vorhanden ist, weist CICS alle Nachrichten zurück, die teilweise oder vollständig verschlüsselt sind. Es wird ein SOAP-Fehler an den Web-Service-Requester ausgegeben.

Beispiel

```
<dfhwsse_configuration version="1">
  <sts_authentication action="issue">
    <auth_token_type>
      <namespace>http://example.org.tokens</namespace>
      <element>benutzernamenstoken</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>
```

Pipelinekonfigurationselement `<authentication>`:

Gibt die Verwendung von Sicherheitstoken in den Headern von ein- und ausgehenden SOAP-Nachrichten an.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

```
<dfhwsse_configuration>
```

Attribute:

Attribut	Beschreibung
'trust' und 'mode'	<p>Gemeinsam geben die Attribute 'trust' und 'mode' Folgendes an:</p> <ul style="list-style-type: none"> • Ob eine zugesicherte Identität verwendet wird. • Die Kombination von Sicherheitstoken, die in SOAP-Nachrichten verwendet werden. <p>Eine zugesicherte Identität ermöglicht es einem vertrauenswürdigen Benutzer, zu bestimmen, dass die Arbeit unter einer anderen Identität ausgeführt werden muss, der <i>zugesicherten Identität</i>, ohne dass der vertrauenswürdige Benutzer die Berechtigungsnachweise besitzt, die dieser Identität zugeordnet sind.</p> <p>Wenn eine zugesicherte Identität verwendet wird, enthalten Nachrichten ein <i>Vertrauensstellungstoken</i> und ein <i>Identitätstoken</i>. Das Vertrauensstellungstoken wird verwendet, um zu prüfen, dass der Sender über die korrekten Berechtigungen verfügt, um Identitäten zuzusichern. Das Identitätstoken enthält die zugesicherte Identität, d. h. die Benutzer-ID, unter der die Anforderung ausgeführt wird.</p> <p>Die Verwendung einer zugesicherten Identität erfordert, dass ein Service-Provider dem Requester soweit vertraut, dass er ihm diese Zusage überlässt. In CICS wird die Vertrauensbeziehung mit Sicherheitsmanager-Ersatzdefinitionen eingerichtet: die anfordernde Identität muss über die richtige Berechtigung verfügen, um im Auftrag der zugesicherten Identität zu arbeiten.</p> <p>Die zulässigen Kombinationen dieser Attribute und die jeweilige Bedeutung werden unter Tabelle 3 und Tabelle 4 auf Seite 140 beschrieben.</p>

Tabelle 3. Die Attribute **mode** und **trust** in einer Service-Requester-Pipeline

trust	mode	Bedeutung
n/v	n/v	Es werden keine Berechtigungsnachweise zur Nachricht hinzugefügt.
n/v	basic	<i>Ungültige Kombination von Attributwerten</i>
n/v	signature	Zugesicherte Identität wird nicht verwendet. CICS verwendet ein einzelnes X.509-Sicherheitstoken, das zur Nachricht hinzugefügt und zum Signieren des Nachrichtenhauptteils verwendet wird. Das Zertifikat wird mit dem Element <certificate_label> angegeben und der Algorithmus wird im Element <algorithm> angegeben.
blind	n/v	<i>Ungültige Kombination von Attributwerten</i>
blind	basic	Zugesicherte Identität wird nicht verwendet. CICS fügt ein Identitätstoken zur Nachricht hinzu, stellt aber kein Vertrauensstellungstoken bereit. Bei dem Identitätstoken handelt es sich um einen Benutzernamen ohne Kennwort. Die Benutzer-ID, die in dem Identitätstoken platziert wird, ist der Inhalt des Containers DFHWS-USERID (der standardmäßig die Benutzer-ID der aktiven Task enthält).
blind	signature	<i>Ungültige Kombination von Attributwerten</i>
basic	n/v	<i>Ungültige Kombination von Attributwerten</i>
basic	basic	<i>Ungültige Kombination von Attributwerten</i>

Tabelle 3. Die Attribute **mode** und **trust** in einer Service-Requester-Pipeline (Forts.)

trust	mode	Bedeutung
basic	signature	Ungültige Kombination von Attributwerten
signature	n/v	Ungültige Kombination von Attributwerten
signature	basic	<p>Zugesicherte Identität wird verwendet. CICS fügt die folgenden Token zur Nachricht hinzu:</p> <ul style="list-style-type: none"> • Das Vertrauensstellungstoken ist ein X.509-Sicherheitstoken. • Bei dem Identitätstoken handelt es sich um einen Benutzernamen ohne Kennwort. <p>Das Zertifikat, das zum Signieren des Identitätstokens und des Nachrichtenhauptteils verwendet wird, wird durch <certificate_label> angegeben. Die Benutzer-ID, die in dem Identitätstoken platziert wird, ist der Inhalt des Containers DFHWS-USERID (der standardmäßig die Benutzer-ID der aktiven Task enthält).</p>
signature	signature	Ungültige Kombination von Attributwerten

Tabelle 4. Die Attribute **mode** und **trust** in einer Service-Provider-Pipeline

trust	mode	Bedeutung
n/v	n/v	Eingehende Nachrichten müssen keine Berechtigungsnachweise enthalten und CICS versucht nicht, Berechtigungsnachweise, die in einer Nachricht gefunden werden, zu extrahieren oder zu verifizieren. CICS prüft jedoch, dass alle signierten Elemente korrekt signiert sind.
n/v	basic	Eingehende Nachrichten müssen ein Benutzernamen-Sicherheitstoken mit einem Kennwort enthalten. CICS stellt den Benutzernamen in den Container DFHWS-USERID.
n/v	basic-ICRX	Ungültige Kombination von Attributwerten
n/v	basic-kerberos	Ungültige Kombination von Attributwerten
n/v	signature	Eingehende Nachrichten müssen ein X.509-Sicherheitstoken enthalten, das zum Signieren des Nachrichtenhauptteils verwendet wurde.
blind	n/v	Ungültige Kombination von Attributwerten
blind	basic	Eingehende Nachrichten müssen ein Identitätstoken enthalten, wobei das Identitätstoken eine Benutzer-ID und optional ein Kennwort enthält. CICS stellt die Benutzer-ID in den Container DFHWS-USERID. Wenn kein Kennwort angegeben ist, verwendet CICS die Benutzer-ID, ohne sie zu überprüfen. Wenn ein Kennwort angegeben ist, wird es vom Handler DFHWSSE1 überprüft.
blind	basic-ICRX	Eingehende Nachrichten müssen ein ICRX-Identitätstoken enthalten. CICS löst die Identität auf, stellt die Benutzer-ID in den Container DFHWS-USERID und die ICRX in den Container DFHWS-ICRX. Wenn eine Authentifizierung erforderlich ist, verwendet sie clientzertifiziertes SSL oder ein anderes Sicherheitsprotokoll.

Tabelle 4. Die Attribute **mode** und **trust** in einer Service-Provider-Pipeline (Forts.)

trust	mode	Bedeutung
blind	basic-kerberos	Ungültige Kombination von Attributwerten
blind	signature	Eingehende Nachrichten müssen ein Identitätstoken enthalten, wobei das Identitätstoken das erste X.509-Zertifikat im SOAP-Nachrichtenheader ist. Das Zertifikat muss die Nachricht nicht signiert haben. Der Sicherheitshandler extrahiert die übereinstimmende Benutzer-ID und platziert sie im Container DFHWS-USERID.
basic	n/v	Ungültige Kombination von Attributwerten
basic	basic	Eingehende Nachrichten müssen eine zugesicherte Identität verwenden: <ul style="list-style-type: none"> • Das Vertrauensstellungstoken ist ein Benutzernamenstoken mit einem Kennwort. • Das Identitätstoken ist ein zweites Benutzernamenstoken ohne Kennwort. CICS stellt diesen Benutzernamen in den Container DFHWS-USERID.
basic	basic-ICRX	Eingehende Nachrichten müssen eine zugesicherte Identität verwenden: <ul style="list-style-type: none"> • Das Vertrauensstellungstoken ist ein Benutzernamenstoken mit einem Kennwort. <p>CICS stellt fest, ob die Kombination aus Benutzer-ID und Kennwort gültig ist. Wenn sie gültig ist, löst CICS die zugesicherte ICRX-basierte Identität in eine Benutzer-ID auf. Anschließend führt CICS eine Ersatzsicherheitsprüfung von der authentifizierten Identität für die zugesicherte Identität aus.</p> <ul style="list-style-type: none"> • Das Identitätstoken ist eine ICRX, die den spezifischen Clientbenutzer angibt. CICS stellt den Benutzernamen in den Container DFHWS-USERID und die ICRX in den Container DFHWS-ICRX.
basic	basic-kerberos	Eingehende Nachrichten müssen eine zugesicherte Identität verwenden. <p>Ein Token ist erforderlich, ein Kerberos Version 5-Token mit einem der folgenden Formattypen:</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>Das Token muss Base64-codiert sein. CICS validiert das Token mithilfe des Netzauthentifizierungsservice für z/OS und stellt die dem Token zugeordnete Benutzer-ID in den Container DFHWS-USERID.</p>

Tabelle 4. Die Attribute **mode** und **trust** in einer Service-Provider-Pipeline (Forts.)

trust	mode	Bedeutung
basic	signature	Eingehende Nachrichten müssen eine zugesicherte Identität verwenden: <ul style="list-style-type: none"> • Das Vertrauensstellungstoken ist ein Benutzernamenstoken mit einem Kennwort. • Das Identitätstoken ist ein X.509-Zertifikat. CICS stellt die dem Zertifikat zugeordnete Benutzer-ID in den Container DFHWS-USERID.
signature	n/v	<i>Ungültige Kombination von Attributwerten</i>
signature	basic	Eingehende Nachrichten müssen eine zugesicherte Identität verwenden: <ul style="list-style-type: none"> • Das Vertrauensstellungstoken ist ein X.509-Zertifikat. • Das Identitätstoken ist ein Benutzernamenstoken ohne Kennwort. CICS stellt den Benutzernamen in den Container DFHWS-USERID. <p>Das Identitätstoken und der Hauptteil müssen mit dem X.509-Zertifikat signiert sein.</p>
signature	basic-ICRX	Eingehende Nachrichten müssen eine zugesicherte Identität verwenden. <ul style="list-style-type: none"> • Das Vertrauensstellungstoken ist eine ICRX, signiert mit einem X.509-Zertifikat. <p>CICS löst das X.509-Zertifikat in eine Benutzer-ID auf und stellt sicher, dass die XML-Signatur gültig ist. CICS löst die zugesicherte ICRX-basierte Identität in eine Benutzer-ID auf. Anschließend führt CICS eine Ersatzsicherheitsprüfung von der authentifizierten X.509-Identität für die zugesicherte ICRX-Identität aus.</p> <ul style="list-style-type: none"> • Das Identitätstoken ist ein Benutzernamenstoken ohne Kennwort. CICS stellt den Benutzernamen in den Container DFHWS-USERID und die ICRX in den Container DFHWS-ICRX.
signature	basic-kerberos	<i>Ungültige Kombination von Attributwerten</i>
signature	signature	Eingehende Nachrichten müssen eine zugesicherte Identität verwenden: <ul style="list-style-type: none"> • Das Vertrauensstellungstoken ist ein X.509-Zertifikat. • Das Identitätstoken ist ein zweites X.509-Zertifikat. CICS stellt die diesem Zertifikat zugeordnete Benutzer-ID in den Container DFHWS-USERID. <p>Das Identitätstoken und der Hauptteil müssen mit dem ersten X.509-Zertifikat (dem Vertrauensstellungstoken) signiert sein.</p>

Hinweise:

1. Die Kombinationen der 'trust'- und 'mode'-Attributwerte werden bei der Installation der PIPELINE geprüft. Die Installation schlägt fehl, wenn die Attribute falsch codiert sind.
2. CICS verwendet die Kennwortüberprüfung, um eine Benutzer-ID während des beschriebenen Prozesses in VERIFY PHRASE zu überprüfen.

Enthält:

1. Ein optionales leeres <suppress/>-Element.

Wenn dieses Element in einer Service-Provider-Pipeline angegeben ist, versucht der Handler nicht, anhand von Sicherheitstoken in der Nachricht zu bestimmen, welche Benutzer-ID aktiv ist.

Wenn dieses Element in einer Service-Requester-Pipeline angegeben ist, versucht der Handler nicht, Sicherheitstoken, die für die Authentifizierung erforderlich sind, zu der ausgehenden SOAP-Nachricht hinzuzufügen.

2. In einer Requester-Pipeline ein optionales <algorithm>-Element, das den URI des Algorithmus angibt, der zum Signieren des Hauptteils der SOAP-Nachricht verwendet wird. Sie müssen dieses Element angeben, wenn die Kombination von 'trust' und 'mode'-Attributen angibt, dass die Nachrichten signiert sind. Sie können nur den 'RSA mit SHA1'-Algorithmus in diesem Element angeben. Der URI ist <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.
3. Ein optionales <certificate_label>-Element, das die Bezeichnung angibt, die einem digitalen X.509-Zertifikat zugeordnet ist, das in RACF installiert ist. Wenn Sie dieses Element in einer Service-Requester-Pipeline angeben und das <suppress>-Element ist nicht angegeben, wird das Zertifikat zum Sicherheitsheader in der SOAP-Nachricht hinzugefügt. Wenn Sie kein <certificate_label>-Element angeben, verwendet CICS das Standardzertifikat im RACF-Schlüsselring.

Dieses Element wird in einer Service-Provider-Pipeline ignoriert.

Beispiel

```
<authentication trust="signature" mode="basic">
  <suppress/>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>AUTHCERT03</certificate_label>
</authentication>
```

Pipelinekonfigurationselement <sts_authentication>:

Gibt an, dass ein Sicherheitstokenservice (STS) für die Authentifizierung verwendet werden muss, und bestimmt die Art von Anforderung, die gesendet wird.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

```
<dfhwsse_configuration>
```

Attribute:

Name	Beschreibung
action	<p>Gibt an, welche Art von Anforderung CICS an den STS sendet, wenn eine Nachricht in der Service-Provider-Pipeline empfangen wird. Gültige Werte sind:</p> <p>issue Der STS gibt ein Identitätstoken für die SOAP-Nachricht aus. Dieser Wert ist für SAML in einer Provider-Pipeline nicht gültig.</p> <p>validate Der STS validiert das bereitgestellte Identitätstoken und gibt zurück, ob das Token für den Sicherheitshandler gültig ist. Wenn Sie dieses Attribut nicht angeben, nimmt CICS an, dass die Aktion darin besteht, ein Identitätstoken zu senden.</p> <p>In einer Service-Requester-Pipeline können Sie dieses Attribut nicht angeben, weil CICS immer erfordert, dass der STS ein Token ausgibt.</p>
extract	<p>Dieses Attribut ist nur gültig, wenn Sie SAML verwenden. Sollen die Elemente des SAML-Token extrahiert werden? Gültige Werte sind:</p> <p>no Die Elemente des SAML-Tokens werden nicht in Container extrahiert.</p> <p>yes Die Hauptelemente des SAML-Tokens werden extrahiert und in Container gestellt, die von CICS erstellt werden.</p>
token_signature	<p>Dieses Attribut ist nur gültig, wenn Sie SAML verwenden. Muss eine Tokensignatur bereitgestellt werden? Gültige Werte sind:</p> <p>ignored Alle bereitgestellten Signaturen werden ignoriert.</p> <p>required Es muss eine gültige Signatur bereitgestellt werden. Dies ist der Standardwert.</p>

Name	Beschreibung
tran_channel	<p>Dieses Attribut ist nur gültig, wenn Sie SAML verwenden. In einer Service-Provider-Pipeline gibt dieses Attribut an, ob SAML-Zusicherungen, die in einer Nachricht enthalten sind, die in der Pipeline empfangen wird, für das Ziellanwendungsprogramm in Containern im Transaktionskanal DFHTRANSACTION verfügbar gemacht werden. Gültige Werte sind:</p> <p>yes Die SAML-Zusicherungen werden in Container in den Kanal DFHTRANSACTION kopiert, um dem Programm zur Verfügung gestellt zu werden. Weitere Informationen zu Containernamen und -typen finden Sie unter The SAML linkable interface, DFHSAML.</p> <p>no SAML-Zusicherungen werden dem Programm nicht über den Kanal DFHTRANSACTION zur Verfügung gestellt, sondern in Containern in dem Kanal, der an das Programm von der Pipeline übergeben wird. Dies ist der Standardwert.</p> <p>Wenn Sie dieses Attribut für einen Service-Provider nicht angeben, werden die Zusicherungen nur in Containern in dem Kanal zur Verfügung gestellt, der an das Programm von der SOAP-Pipeline übergeben wird.</p> <p>In einer Service-Requester-Pipeline gibt dieses Attribut an, ob das SAML-Token, das im Container DFHSAML-OUTTOKEN des Transaktionskanals DFHTRANSACTION enthalten ist, für die Anforderung verwendet wird. Gültige Werte sind:</p> <p>yes Der Inhalt des Containers DFHSAML-OUTTOKEN des Kanals DFHTRANSACTION wird als SAML-Token für die Anforderung verwendet.</p> <p>no Der Inhalt des Containers DFHSAML-OUTTOKEN in dem Kanal, der an die Pipeline übergeben wird, wird als SAML-Token der Anforderung verwendet. Dies ist der Standardwert.</p> <p>Wenn Sie dieses Attribut für einen Service-Requester nicht angeben, wird das SAML-Token aus dem Container DFHSAML-OUTTOKEN in dem Kanal genommen, der an die SOAP-Pipeline übergeben wird.</p>

Enthält:

- Ein Element <auth_token_type>. Dieses Element ist erforderlich, wenn Sie das Element <sts_authentication> in einer Service-Requester-Pipeline angeben, und optional in einer Service-Provider-Pipeline. Weitere Informationen finden Sie unter The <auth_token_type> element.
 - In einer Service-Requester-Pipeline gibt das <auth_token_type>-Element den Typ des Tokens an, das der STS ausgibt, wenn CICS die Benutzer-ID sendet, die im Container DFHWS-USERID enthalten ist. Das Token, das CICS von dem STS empfängt, wird in dem Header der ausgehenden Nachricht platziert.
 - In einer Service-Provider-Pipeline wird das <auth_token_type>-Element verwendet, um das Identitätstoken zu bestimmen, das CICS dem Nachrichtenheader entnimmt und zwecks Austausch oder Validierung an den STS sen-

det. CICS verwendet das erste Identitätstoken des angegebenen Typs im Nachrichtenheader. Wenn Sie dieses Element nicht angeben, verwendet CICS das erste Identitätstoken, das es im Nachrichtenheader findet. Folgendes zieht CICS nicht als Identitätstoken in Betracht:

- wsu:Timestamp
- xenc:ReferenceList
- xenc:EncryptedKey
- ds:Signature

2. Ein optionales leeres Element `<suppress/>` (nur in einer Service-Provider-Pipeline). Wenn dieses Element angegeben ist, versucht der Handler nicht, anhand von Sicherheitstoken in der Nachricht zu bestimmen, welche Benutzer-ID aktiv ist. Das Element `<suppress/>` schließt das Identitätstoken ein, das von dem STS zurückgegeben wird.

Beispiel

Das folgende Beispiel zeigt eine Service-Provider-Pipeline, in der der Sicherheitshandler ein Token vom STS anfordert.

```
<sts_authentication action="issue">
  <auth_token_type>
    <namespace>http://example.org.tokens</namespace>
    <element>benutzernamens token</element>
  </auth_token_type>
  <suppress/>
</sts_authentication>
```

Pipelinekonfigurationselement `<auth_token_type>`:

Gibt an, welcher Typ von Identitätstoken erforderlich ist.

Dieses Element ist obligatorisch, wenn Sie das `<sts_authentication>`-Element in einer Service-Requester-Pipeline angeben, und optional in einem Service-Provider.

- In einer Service-Requester-Pipeline gibt das `<auth_token_type>`-Element den Typ des Tokens an, das der STS ausgibt, wenn CICS die Benutzer-ID sendet, die im Container DFHWS-USERID enthalten ist. Das Token, das CICS von dem STS empfängt, wird in dem Header der ausgehenden Nachricht platziert.
- In einer Service-Provider-Pipeline wird das `<auth_token_type>`-Element verwendet, um das Identitätstoken zu bestimmen, das CICS dem Nachrichtenheader entnimmt und zwecks Austausch oder Validierung an den STS sendet. CICS verwendet das erste Identitätstoken des angegebenen Typs im Nachrichtenheader. Wenn Sie dieses Element nicht angeben, verwendet CICS das erste Identitätstoken, das es im Nachrichtenheader findet. Folgendes zieht CICS nicht als Identitätstoken in Betracht:

- wsu:Timestamp
- xenc:ReferenceList
- xenc:EncryptedKey
- ds:Signature

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

<sts_authentication>

Enthält:

1. Ein Element <namespace>. Dieses Element enthält den Namensbereich des Tokentyps, der validiert oder ausgetauscht werden soll.
Wenn Sie SAML verwenden, legen Sie den Inhalt dieses Elements abhängig von der Version von SAML auf urn:oasis:names:tc:SAML:1.0:assertion oder urn:oasis:names:tc:SAML:2.0:assertion fest.
2. Ein Element <element>. Dieses Element enthält den lokalen Namen des Tokentyps, der validiert oder ausgetauscht werden soll.
Verwenden Sie für SAML den lokalen Namen Assertion.

Die Werte dieser Elemente bilden den Warteschlangennamen des Token.

Beispiel

```
<auth_token_type>
  <namespace>http://example.org.tokens</namespace>
  <element>benutzernamens token</element>
</auth_token_type>
```

Pipelinekonfigurationselement <sts_endpoint>:

Gibt die Position des Sicherheitstokenservice (STS) an.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

<dfhsse_configuration>

Enthält:

- Ein Element <endpoint>. Dieses Element enthält einen URI, der auf die Position des Sicherheitstokenservice (STS) im Netz zeigt. Es wird empfohlen, statt HTTP SSL oder TLS zu verwenden, um die Verbindung zum STS zu sichern.
Um die SAML-Unterstützung zu aktivieren, legen Sie den Endpunkt auf cics://PROGRAM/DFHSAML fest.
Sie können mithilfe des JMS-Formats des URI auch einen WebSphere MQ-Endpunkt angeben.
- Ein optionales Element <jvmserver>. Dieses Element gibt den JVM-Server an, der für die Ausführung des SAML-Token-Service konfiguriert ist. Ist dieses Element nicht angegeben, wird standardmäßig der JVM-Server der Beispielfressource, DFHXSTS, angenommen. Dieses Element ist nur gültig, wenn SAML verwendet wird. In allen anderen Fällen tritt ein Fehler auf.

Beispiele

In diesem Beispiel ist der Endpunkt so konfiguriert, dass eine sichere Verbindung zum STS an dem angegebenen URI verwendet wird.

```
<sts_endpoint>
  <endpoint>https://example.com/SecurityTokenService</endpoint>
</sts_endpoint>
```

In diesem Beispiel ist der Endpunkt so konfiguriert, dass CICS-SAML-Unterstützung verwendet wird.

```
<sts_endpoint>
  <endpoint>cics://PROGRAM/DFHSAML</endpoint>
</sts_endpoint>
```

Pipelinekonfigurationselement <sign_body>:

Weist DFHWSSE an, den Hauptteil ausgehender SOAP-Nachrichten zu signieren, und stellt Informationen darüber bereit, wie die Nachrichten signiert werden sollen.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

```
<dfhwsse_configuration>
```

Enthält:

1. Ein <algorithm>-Element, das den URI enthält, der den Algorithmus zum Signieren des Hauptteils der SOAP-Nachricht angibt. Sie können die unter Signature algorithms aufgeführten Algorithmen angeben.
2. Ein Element <certificate_label>, das die Bezeichnung angibt, die einem digitalen Zertifikat, das in RACF installiert ist, zugeordnet wird. Das digitale Zertifikat stellt den Schlüssel zum Signieren der Nachricht bereit.

Beispiel

```
<sign_body>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>SIGCERT01</certificate_label>
</sign_body>
```

Pipelinekonfigurationselement <encrypt_body>:

Weist DFHWSSE an, den Hauptteil ausgehender SOAP-Nachrichten zu verschlüsseln, und stellt Informationen darüber bereit, wie die Nachrichten verschlüsselt werden sollen.

Verwendet in:

- Service-Provider
- Service-Requester

Enthalten in:

```
<dfhwsse_configuration>
```

Enthält:

1. Ein <algorithm>-Element, das den URI enthält, der den Algorithmus zum Verschlüsseln des Hauptteils der SOAP-Nachricht angibt. Sie können die unter Encryption algorithms aufgeführten Algorithmen angeben.
2. Ein Element <certificate_label>, das die Bezeichnung angibt, die einem digitalen Zertifikat in RACF zugeordnet wird. Das digitale Zertifikat stellt den Schlüssel zur Verschlüsselung der Nachricht bereit.

Beispiel

```
<encrypt_body>
  <algorithm>http://www.w3.org/2001/04/xmlenc#aes256-cbc</algorithm>
  <certificate_label>ENCCERT02</certificate_label>
</encrypt_body>
```

Anwendungshandler

Ein Anwendungshandler ist ein CICS-Programm, mit dem der Terminal-Handler einer SOAP-Service-Provider-Pipeline zur Laufzeit verknüpft wird.

Anwendungshandler werden in Pipelines im Providermodus verwendet, in denen der Terminal-Handler einer der bereitgestellten SOAP-Nachrichtenhandler ist. Dies ist der Fall, wenn das `<terminal_handler>`-Element ein `<cics_soap_1.1_handler>`-, `<cics_soap_1.2_handler>`-, `<cics_soap_1.1_handler_java>`- oder `<cics_soap_1.2_handler_java>`-Element enthält.

Der Anwendungshandler ist für die Verarbeitung des Hauptteils einer SOAP-Anforderung und für die Generierung einer Antwort unter Verwendung der zurückgegebenen Daten verantwortlich. Der Anwendungshandler kann andere Programme aufrufen, um diese Verarbeitung abzuschließen. Typischerweise agiert der Anwendungshandler als vielseitig einsetzbare Darstellungsebene für eine oder mehrere Geschäftsanwendungen. Er ist verantwortlich für das Zuordnen von XML zu einem Formular, das eine Anwendung verwenden kann, das Anhängen dieser Anwendung und anschließend das Generieren einer Antwort unter Verwendung der zurückgegebenen Daten.

Ein Anwendungshandler kann auf zwei Arten von CICS angehängt werden. Die typische Vorgehensweise bezieht einen Kanal und Steuercontainer ein, die andere Methode verwendet Java-Bindungen für Axis2.

Über einen Kanal angehängte Anwendungshandler werden im `<apphandler>`-Element des `<provider_pipeline>`-Elements angegeben. Zur Laufzeit wird der Container DFHWS-APPHANDLER mit den Inhalten von `<apphandler>` gefüllt. Allerdings kann der Container DFHWS-APPHANDLER dynamisch von einem beliebigen anderen Nachrichtenhandler aktualisiert werden. Deshalb kann sich das Programm, das zur Laufzeit verknüpft wird, von dem im `<apphandler>`-Element angegebenen Programm unterscheiden. Die folgenden Anwendungshandler können im `<apphandler>`-Element oder im Container DFHWS-APPHANDLER angegeben werden:

- Der bereitgestellte, über einen Kanal angehängte SOAP-Anwendungshandler, DFHPITP. Weitere Informationen zu über einen Kanal angehängten Anwendungshandlern finden Sie unter „Über einen Kanal angehängte Anwendungshandler“ auf Seite 150.
- Ihr eigener, über einen Kanal angehängter Anwendungshandler. Dieser Anwendungshandler kann in anderen Sprachen als Java geschrieben werden. Weitere Informationen zu den Steuercontainern, die in Ihrem, über einen Kanal angehängten Anwendungshandler verwendet werden können, finden Sie unter „Steuercontainer“ auf Seite 166.
- Ihr eigener Java-Anwendungshandler für Java-basierte Pipelines, der die Java-Schnittstelle 'ApplicationHandler' implementiert und mithilfe von Axis2 MessageContext an die Pipeline angehängt wird. Weitere Informationen zur Java-Schnittstelle 'ApplicationHandler' finden Sie unter Interface ApplicationHandler.

Um einen Anwendungshandler mit Java-Bindungen für Axis2 zu verwenden, müssen Sie das `<apphandler_class>`-Element des `<provider_pipeline>`-Elements angeben. Axis2-Anwendungshandler erfordern auch, dass ein JVM-Server vorhanden

ist, auf dem die Web-Service-Pipeline und der Anwendungshandler ausgeführt werden können, und dass der Terminal-Handler Ihrer Web-Service-Pipeline entweder der Nachrichtenhandler `<cics_soap_1.1_handler_java>` oder der Nachrichtenhandler `<cics_soap_1.2_handler_java>` ist. Um den bereitgestellten Axis2-Anwendungshandler zu verwenden, müssen Sie `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` in dem `<apphandler_class>`-Element angeben. Sie können jedoch auch Ihre eigene Axis2-Anwendungshandlerklasse angeben. Zur Laufzeit wird der Container DFHWS-APPHANCLAS mit den Inhalten von `<apphandler_class>` gefüllt.

Für Web-Service-Anwendungen, die mithilfe des CICS-Web-Service-Assistenten implementiert werden, müssen Sie entweder DFHPITP oder Ihren eigenen Anwendungshandler angeben, der DFHPITP im `<apphandler>`-Element verwendet. Oder Sie geben `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` im `<apphandler_class>`-Element an. Weitere Informationen zum CICS-Web-Service-Assistenten finden Sie unter The CICS web services assistant.

Es ist auch möglich, Axis2-Anwendungen als Web-Services im Providermodus in CICS zu implementieren. Verwenden Sie dazu den Axis2-Stil der Web-Service-Implementierung. Weitere Informationen finden Sie unter Deploying a Java provider-mode web service in an Axis2 JVM server.

Über einen Kanal angehängte Anwendungshandler

Über einen Kanal angehängte Anwendungshandler sind Anwendungshandler, die in CICS mithilfe eines Kanals und von Steuercontainern angehängt werden.

Bei dem vom Anwendungshandler verwendeten Kanal handelt es sich um den Kanal DFHAHC-V1. Dieser Kanal übergibt die folgenden Container zwischen dem Terminal-Handler und der Web-Service-Anwendung im Providermodus:

DFHWS-XMLNS

Enthält eine Liste mit Name/Wert-Paaren, die Namensbereichspräfixe zu Namensbereichen zuordnen.

- Bei Eingabe enthält die Liste die Namensbereiche, die sich im Geltungsbereich des SOAP-Envelopes befinden.
- Bei Ausgabe enthält die Liste die Namensbereichsdaten, von denen angenommen wird, dass sie sich im Envelope-Tag befinden.

DFHWS-BODY

Enthält den Hauptteil des SOAP-Envelopes. In der Regel ändert die Anwendung den Inhalt. Wenn die Anwendung den Inhalt nicht ändert, muss das Anwendungshandlerprogramm den Inhalt dieses Containers aktualisieren, auch wenn es denselben Inhalt wieder an den Container zurückgibt, bevor es zum Terminal-Handler zurückkehrt.

DFHNORESPONSE

Gibt in der Anforderungsphase einer Service-Requester-Pipeline an, dass der Service-Provider keine Antwort zurückgeben soll. Der Inhalt des Containers DFHNORESPONSE ist nicht definiert. Nachrichtenhandler, die wissen müssen, ob der Service-Provider eine Antwort zurückgeben soll, müssen nur feststellen, ob der Container vorhanden ist oder nicht:

- Wenn der Container DFHNORESPONSE vorhanden ist, wird keine Antwort erwartet.
- Wenn der Container DFHNORESPONSE fehlt, **wird** eine Antwort erwartet.

Der Kanal übergibt auch alle Kontextcontainer, die an den Terminal-Handler übergeben wurden. Beispielsweise kann ein Programm zur Headerverarbeitung Container zum Kanal hinzufügen. Diese Container werden als Benutzercontainer übergeben. Weitere Informationen zu Anwendungshandlern finden Sie unter „Anwendungshandler“ auf Seite 106.

Nachrichtenhandler

Ein Nachrichtenhandler ist ein CICS-Programm zur Verarbeitung einer Web-Service-Anforderung bei der Eingabe und der Antwort bei der Ausgabe. Nachrichtenhandler interagieren über Kanäle und Container miteinander und mit dem System.

Über die Nachrichtenhandlerschnittstelle können Sie die folgenden Tasks in einem Nachrichtenhandlerprogramm ausführen:

- Untersuchen des Inhalts einer XML- oder JSON-Anforderung oder -Antwort, ohne sie zu ändern
- Ändern des Inhalts einer XML- oder JSON-Anforderung oder -Antwort
- Übergeben einer XML- oder JSON-Anforderung oder -Antwort an den nächsten Nachrichtenhandler in der Pipeline im Fall eines Nicht-Terminal-Nachrichtenhandlers
- Aufrufen eines Anwendungsprogramms und Generieren einer Antwort im Fall eines Terminal-Nachrichtenhandlers
- Erzwingen einer Umsetzung aus der Anforderungsphase der Pipeline in die Antwortphase, wobei die Anforderung eingegliedert und eine Antwort generiert wird
- Beheben von Fehlern

Tip: Es empfiehlt sich, die SOAP-Handler `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` oder `<cics_soap_1.2_handler_java>` für die Arbeit mit SOAP-Nachrichten zu verwenden. Mit diesen Handlern können Sie direkt mit den Hauptelementen in einer SOAP-Nachricht (SOAP-Header und SOAP-Hauptteil) arbeiten.

Alle Programme, die als Nachrichtenhandler verwendet werden, werden über dieselbe Schnittstelle aufgerufen: einen *Kanal*, der eine Anzahl von Containern enthält. Die Container können wie folgt kategorisiert werden:

Steuercontainer

Diese sind entscheidend für den Betrieb der Pipeline. Nachrichtenhandler können die Steuercontainer verwenden, um die Reihenfolge zu ändern, in der nachfolgende Handler verarbeitet werden.

Kontextcontainer

In einigen Fällen benötigen Nachrichtenhandlerprogramme Informationen über den Kontext, in dem sie aufgerufen werden. CICS stellt diese Informationen in einem Satz von *Kontextcontainern* bereit, der an die Programme übergeben wird.

Einige Kontextcontainer enthalten Informationen, die Sie in Ihrem Nachrichtenhandler ändern können. In einer Service-Provider-Pipeline können Sie beispielsweise die Benutzer-ID und die Transaktions-ID des Zielanwendungsprogramms ändern, indem Sie die Inhalte der entsprechenden Kontextcontainer ändern.

Benutzercontainer

Diese enthalten Informationen, die ein Nachrichtenhandler an einen anderen übergeben muss. Benutzercontainer werden ausschließlich von Nachrichtenhandlern verwendet.

Restriction: Die Namen von Benutzercontainern dürfen nicht mit DFH beginnen.

Steuerung der Pipelineprotokolle durch Container

Die Inhalte der Container DFHFUNCTION, DFHREQUEST und DFHRESPONSE steuern gemeinsam die Pipelineprotokolle.

Während der zwei Phasen der Ausführung einer Pipeline (Anforderungsphase und Antwortphase) bestimmt der Wert von DFHFUNCTION, welche Steuercontainer an die einzelnen Nachrichtenhandler übergeben werden:

DFHFUNCTION	Kontext	DFHREQUEST	DFHRESPONSE
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Vorhanden (Länge > 0)	Vorhanden (Länge = 0)
SEND-RESPONSE	Service-Provider; Antwortphase	Nicht vorhanden	Vorhanden (Länge > 0)
SEND-REQUEST	Service-Requester; Anforderungsphase	Vorhanden (Länge > 0)	Vorhanden (Länge = 0)
RECEIVE-RESPONSE	Service-Requester; Antwortphase	Nicht vorhanden	Vorhanden (Länge > 0)
PROCESS-REQUEST	Service-Provider; Terminal-Handler	Vorhanden (Länge > 0)	Vorhanden (Länge = 0)
HANDLER-ERROR	Service-Requester oder -Provider; beide Phasen	Nicht vorhanden	Vorhanden (Länge = 0)
NO-RESPONSE	Service-Requester oder -Provider; Antwortphase	Nicht vorhanden	Nicht vorhanden

Die nachfolgende Verarbeitung wird von den Containern bestimmt, die Ihr Nachrichtenhandler an die Pipeline zurückgibt:

Während der Anforderungsphase

- Ihr Nachrichtenhandler kann den Container DFHREQUEST zurückgeben. Die Verarbeitung wird in der Anforderungsphase mit dem nächsten Handler fortgesetzt. Die Länge der Daten im Container darf nicht null sein.
- Ihr Nachrichtenhandler kann den Container DFHRESPONSE zurückgeben. Die Verarbeitung wechselt in die Antwortphase und derselbe Handler wird aufgerufen, wobei DFHFUNCTION auf SEND-RESPONSE in einem Service-Provider und auf RECEIVE-RESPONSE in einem Service-Requester gesetzt ist. Die Länge der Daten im Container darf nicht null sein.
- Ihr Nachrichtenhandler kann keine Container zurückgeben. Die Verarbeitung wechselt in die Antwortphase und derselbe Handler wird aufgerufen, wobei DFHFUNCTION auf NO-RESPONSE gesetzt ist.

Im Terminal-Handler (nur Service-Provider)

- Ihr Nachrichtenhandler kann den Container DFHRESPONSE zurückgeben. Die Verarbeitung wechselt in die Antwortphase und der vorherige Handler wird mit einem neuen Wert von DFHFUNCTION (SEND-RESPONSE) aufgerufen. Die Länge der Daten im Container darf nicht null sein.
- Ihr Nachrichtenhandler kann keine Container zurückgeben. Die Verarbeitung wechselt in die Antwortphase und der vorherige Handler wird mit einem neuen Wert von DFHFUNCTION (NO-RESPONSE) aufgerufen.

Während der Antwortphase

- Ihr Nachrichtenhandler kann den Container DFHRESPONSE zurückgeben. Die Verarbeitung wird in der Antwortphase fortgesetzt und der nächste Handler wird aufgerufen. Die Länge der Daten im Container darf nicht null sein.
- Ihr Nachrichtenhandler kann keine Container zurückgeben. Die Verarbeitung wird in der Antwortphase fortgesetzt und der nächste Handler in der Abfolge wird mit einem neuen Wert von DFHFUNCTION (NO-RESPONSE) aufgerufen.

Important: Während der Anforderungsphase kann Ihr Nachrichtenhandler DFHREQUEST oder DFHRESPONSE zurückgeben, aber nicht beides. Da beide Container vorhanden sind, wenn Ihr Nachrichtenhandler aufgerufen wird, müssen Sie einen löschen.

In dieser Tabelle ist die Aktion aufgeführt, die von der Pipeline für alle Werte von DFHFUNCTION ausgeführt wird, und es sind alle Kombinationen von DFHREQUEST und DFHRESPONSE angegeben, die von den einzelnen Nachrichtenhandler zurückgegeben werden.

DFHFUNCTION	Kontext	DFHREQUEST	DFHRESPONSE	Aktion
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Vorhanden (Länge > 0)	Vorhanden	(Fehler)
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Vorhanden (Länge > 0)	Nicht vorhanden	Aufrufen des nächsten Handlers mit der Funktion RECEIVE-REQUEST
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Vorhanden (Länge = 0)	Nicht zutreffend	(Fehler)
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Nicht vorhanden	Vorhanden (Länge > 0)	Wechseln in die Antwortphase und Aufrufen desselben Handlers mit der Funktion SEND-RESPONSE
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Nicht vorhanden	Vorhanden (Länge = 0)	(Fehler)
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Nicht vorhanden	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE
SEND-RESPONSE	Service-Provider; Antwortphase	Nicht zutreffend	Vorhanden (Länge > 0)	Aufrufen des vorherigen Handlers mit der Funktion SEND-RESPONSE
SEND-RESPONSE	Service-Provider; Antwortphase	Nicht zutreffend	Vorhanden (Länge = 0)	(Fehler)
SEND-RESPONSE	Service-Provider; Antwortphase	Nicht zutreffend	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE
SEND-REQUEST	Service-Requester; Anforderungsphase	Vorhanden (Länge > 0)	Vorhanden (Länge ≥ 0)	(Fehler)

DFHFUNCTION	Kontext	DFHREQUEST	DFHRESPONSE	Aktion
SEND-REQUEST	Service-Requester; Anforderungsphase	Vorhanden (Länge > 0)	Nicht vorhanden	Aufrufen des nächsten Handlers mit der Funktion SEND-REQUEST
SEND-REQUEST	Service-Requester; Anforderungsphase	Vorhanden (Länge = 0)	Nicht zutreffend	(Fehler)
SEND-REQUEST	Service-Requester; Anforderungsphase	Nicht vorhanden	Vorhanden (Länge > 0)	Wechseln in die Antwortphase und Aufrufen des vorherigen Handlers mit der Funktion RECEIVE-RESPONSE
SEND-REQUEST	Service-Requester; Anforderungsphase	Nicht vorhanden	Vorhanden (Länge = 0)	(Fehler)
SEND-REQUEST	Service-Requester; Anforderungsphase	Nicht vorhanden	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE
RECEIVE-RESPONSE	Service-Requester; Antwortphase	Nicht zutreffend	Vorhanden (Länge > 0)	Aufrufen des vorherigen Handlers mit der Funktion RECEIVE-RESPONSE
RECEIVE-RESPONSE	Service-Requester; Antwortphase	Nicht zutreffend	Vorhanden (Länge = 0)	(Fehler)
RECEIVE-RESPONSE	Service-Requester; Antwortphase	Nicht zutreffend	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE
PROCESS-REQUEST	Service-Provider; Terminal-Handler	Nicht zutreffend	Vorhanden (Länge > 0)	Aufrufen des vorherigen Handlers mit der Funktion RECEIVE-RESPONSE
PROCESS-REQUEST	Service-Provider; Terminal-Handler	Nicht zutreffend	Vorhanden (Länge = 0)	(Fehler)
PROCESS-REQUEST	Service-Provider; Terminal-Handler	Nicht zutreffend	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE
HANDLER-ERROR	Service-Requester oder -Provider; bei- de Phasen	Nicht zutreffend	Vorhanden (Länge > 0)	Aufrufen des vorherigen Handlers mit der Funktion SEND-RESPONSE function
HANDLER-ERROR	Service-Requester oder -Provider; bei- de Phasen	Nicht zutreffend	Vorhanden (Länge = 0)	(Fehler)
HANDLER-ERROR	Service-Requester oder -Provider; bei- de Phasen	Nicht zutreffend	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE

Eigene Nachrichtenhandler bereitstellen

Wenn Sie Nachrichten, die zwischen einem Service-Requester und einem Service-Provider ausgetauscht werden, auf spezielle Weise verarbeiten möchten, und CICS stellt keinen Nachrichtenhandler bereit, der Ihre Anforderungen erfüllt, müssen Sie Ihren eigenen Nachrichtenhandler bereitstellen.

About this task

In den meisten Situationen können Sie die gesamte erforderliche Verarbeitung mit den von CICS bereitgestellten Nachrichtenhandlern ausführen. Sie können beispielsweise die SOAP 1.1- und 1.2-Nachrichtenhandler verwenden, die CICS zur Verarbeitung von SOAP-Nachrichten bereitstellt. Es gibt jedoch Situationen, in denen Sie Ihre eigenen spezialisierten Operationen für Web-Service-Anforderungen und -Antworten ausführen möchten. Dafür müssen Sie Ihre eigenen Nachrichtenhandler bereitstellen.

Procedure

1. Schreiben Sie Ihr Nachrichtenhandlerprogramm. Ein Nachrichtenhandler ist ein CICS-Programm mit einer Kanalschnittstelle. Sie können Ihr Programm in einer beliebigen von CICS unterstützten Sprache schreiben und jeden beliebigen CICS-Befehl im DPL-Subset in Ihrem Programm verwenden.
2. Kompilieren Sie Ihr Programm und stellen Sie eine Programmverbindung her. Nachrichtenhandlerprogramme werden üblicherweise unter der Transaktion CPIH ausgeführt, die mit dem Attribut TASKDATALOC(ANY) definiert wird. Wenn Sie eine Programmverbindung verwenden, müssen Sie daher die Option AMODE(31) angeben.
3. Installieren Sie das Programm regulär in Ihrem CICS-System.
4. Definieren Sie das Programm in der Pipelinekonfigurationsdatei. Verwenden Sie das Element <handler>, um Ihre Nachrichtenhandler zu definieren. Codieren Sie im Element <handler> ein Element <program>, das den Namen des Programms enthält.

Mit Nachrichten in einem Nicht-Terminal-Nachrichtenhandler arbeiten

Ein typischer Nicht-Terminal-Nachrichtenhandler verarbeitet eine Nachricht und übergibt die Steuerung dann an einen anderen Nachrichtenhandler in der Pipeline.

About this task

In einem Nicht-Terminal-Nachrichtenhandler können Sie mit einer Anforderung oder einer Antwort arbeiten, sie ändern oder nicht ändern, und sie anschließend an den nächsten Nachrichtenhandler übergeben.

Anmerkung: Obwohl Web-Services normalerweise SOAP-Nachrichten verwenden, die XML enthalten, funktionieren Ihre Nachrichtenhandler genauso gut mit anderen Nachrichtenformaten.

Procedure

1. Bestimmen Sie anhand der Inhalte des Containers DFHFUNCTION, ob die an diesen Nachrichtenhandler übergebene Nachricht eine Anforderung oder Antwort ist.

DFHFUNCTION	Anforderung oder Antwort	Typ von Nachrichtenhandler	Eingehend oder ausgehend
RECEIVE-REQUEST	Anforderung	Nicht-Terminal	Eingehend
SEND-RESPONSE	Antwort	Nicht-Terminal	Ausgehend
SEND-REQUEST	Anforderung	Nicht-Terminal	Ausgehend
RECEIVE-RESPONSE	Antwort	Nicht-Terminal	Eingehend

Tip:

- Wenn DFHFUNCTION den Wert PROCESS-REQUEST enthält, handelt es sich bei dem Nachrichtenhandler um einen Terminal-Nachrichtenhandler und die folgenden Schritte finden keine Anwendung.
 - Wenn DFHFUNCTION den Wert HANDLER-ERROR enthält, wird der Handler zur Fehlerbehandlung aufgerufen und die folgenden Schritte finden keine Anwendung.
2. Rufen Sie die Anforderung oder die Antwort aus dem passenden Container ab.

- Wenn die Nachricht eine Anforderung ist, wird sie an das Programm im Container DFHREQUEST übergeben. Der Container DFHRESPONSE ist ebenfalls vorhanden und hat eine Länge von null.
 - Wenn die Nachricht eine Antwort ist, wird sie an das Programm im Container DFHRESPONSE übergeben.
3. Führen Sie die erforderliche Nachrichtenverarbeitung aus. Abhängig vom Zweck des Nachrichtenhandlers umfasst dies möglicherweise Folgendes:
- Untersuchen Sie die Nachricht, ohne sie zu ändern, und übergeben Sie sie an den nächsten Nachrichtenhandler in der Pipeline.
 - Ändern Sie die Anforderung und übergeben Sie sie an den nächsten Nachrichtenhandler in der Pipeline.
 - Wenn die Nachricht eine Anforderung ist, können Sie die folgenden Nachrichtenhandler in der Pipeline überspringen und stattdessen eine Antwortnachricht erstellen.

Anmerkung: Die Inhalte des Containers, den ein Nachrichtenhandler zurückgibt, bestimmen, welcher Nachrichtenhandler als nächstes aufgerufen wird.

Ein Fehler tritt auf, wenn ein Nachrichtenhandler keine Änderungen an den Containern vornimmt, die an ihn übergeben werden.

Ein Fehler tritt auf, wenn ein Nachrichtenhandlerprogramm Folgendes zurückgibt:

- Einen leeren Container DFHRESPONSE.
- Einen nicht leeren Container DFHREQUEST und einen nicht leeren Container DFHRESPONSE.
- Einen leeren Container DFHREQUEST in der ausgehenden Anforderung.

Nachrichten an den nächsten Nachrichtenhandler in der Pipeline übergeben:

In einem typischen Nicht-Terminal-Nachrichtenhandler verarbeiten Sie eine Anforderung oder Antwort, die Sie ändern oder nicht ändern, und übergeben diese an den nächsten Nachrichtenhandler.

Procedure

1. Geben Sie die Nachricht - ob geändert oder nicht - im passenden Container an die Pipeline zurück.
 - Wenn die Nachricht eine Anforderung ist und sie diese geändert haben, geben Sie sie im Container DFHREQUEST zurück.
 - Wenn die Nachricht eine Antwort ist und Sie diese geändert haben, stellen Sie sie in den Container DFHRESPONSE.
 - Wenn Sie die Nachricht nicht geändert haben, befindet sie sich bereits im richtigen Container.
2. Wenn die Nachricht eine Anforderung ist, löschen Sie den Container DFHRESPONSE. Wenn ein Nachrichtenhandler für eine Anforderung aufgerufen wird, werden die Container DFHREQUEST und DFHRESPONSE an das Programm übergeben. DFHRESPONSE hat eine Länge von null. Es tritt jedoch ein Fehler auf, wenn sowohl DFHREQUEST als auch DFHRESPONSE zurückgegeben wird.

Results

Die Nachricht wird an den nächsten Nachrichtenhandler in der Pipeline übergeben.

Übergang in die Antwortphase der Pipeline erzwingen:

Wenn Sie eine Anforderung verarbeiten, kann es vorkommen, dass Sie eine sofortige Antwort generieren wollen, anstatt die Anforderung an den nächsten Nachrichtenhandler in der Pipeline zu übergeben.

Procedure

1. Löschen Sie den Container DFHREQUEST.
2. Erstellen Sie Ihre Antwort und stellen Sie sie in den Container DFHRESPONSE.

Results

Die Antwort wird an den nächsten Nachrichtenhandler in der Antwortphase der Pipeline übergeben.

Antwort unterdrücken:

In manchen Situationen möchten Sie unter Umständen eine Anforderung eingliedern, ohne eine Antwort zu senden.

Procedure

1. Löschen Sie den Container DFHREQUEST.
2. Löschen Sie den Container DFHRESPONSE.

Unidirektionale Nachrichten in einer Service-Requester-Pipeline verarbeiten:

Wenn eine Service-Requester-Pipeline eine Anforderung an einen Service-Provider sendet, wird normalerweise eine Antwort erwartet, und es wird davon ausgegangen, dass nach dem Senden der Anforderung die Nachrichtenhandler in der Pipeline erneut aufgerufen werden, wenn die Antwort eintrifft. Manche Web-Services senden keine Antwort, deshalb müssen Sie spezielle Aktionen ausführen, um anzugeben, dass CICS nicht auf eine Antwort warten soll, bevor die Nachrichtenhandler ein zweites Mal aufgerufen werden.

About this task

Dafür müssen Sie sicherstellen, dass der Container DFHNORESPONSE am Ende der Pipelineverarbeitung in der Anforderungsphase vorhanden ist. Dies erfolgt typischerweise auf Ebene des Anwendungscode, weil das Wissen, ob eine Antwort erwartet wird, in der Anwendung selbst vorhanden ist:

- In Anwendungen ,die mit dem CICS-Web-Service-Assistenten implementiert werden, erstellt der CICS-Code den Container.
- Anwendungen, die nicht mit dem Assistenten implementiert werden, erstellen den Container in der Regel, bevor Sie die Anwendung aufrufen.

Wenn Sie den Container DFHNORESPONSE in einem Nachrichtenhandler erstellen oder löschen, müssen Sie sicherstellen, dass dies nicht das Nachrichtenprotokoll zwischen dem Service-Requester und dem Service-Provider stört.

Mit Nachrichten in einem Terminal-Nachrichtenhandler arbeiten

Ein typischer Terminal-Handler verarbeitet eine Anforderung, ruft ein Anwendungsprogramm auf und generiert eine Antwort.

About this task

Anmerkung: Obwohl Web-Services normalerweise SOAP-Nachrichten verwenden, die XML enthalten, funktionieren Ihre Nachrichtenhandler genauso gut mit anderen Nachrichtenformaten.

In einem Terminal-Handler können Sie mit einer Anforderung arbeiten und - optional - eine Antwort generieren und über die Pipeline zurückgeben. Ein typischer Terminal-Handler verwendet die Anforderung als Eingabe für ein Anwendungsprogramm und verwendet die Antwort des Anwendungsprogramms, um die Antwort zu erstellen.

Procedure

1. Bestimmen Sie anhand des Inhalts des Containers DFHFUNCTION, dass die an diesen Handler übergebene Nachricht eine Anforderung ist und dass der Handler als Terminal-Handler aufgerufen wird.

DFHFUNCTION	Anforderung oder Antwort	Typ von Handler	Eingehend oder ausgehend
PROCESS-REQUEST	Anforderung	Terminal	Eingehend

Tip:

- Wenn DFHFUNCTION einen anderen Wert enthält, ist der Handler kein Terminal-Handler und diese Schritte finden keine Anwendung.
2. Rufen Sie die Anforderung aus dem Container DFHREQUEST ab. Der Container DFHRESPONSE ist ebenfalls vorhanden und hat eine Länge von null.
 3. Führen Sie die erforderliche Nachrichtenverarbeitung aus. In der Regel ruft ein Terminal-Handler ein Anwendungsprogramm auf.
 4. Erstellen Sie Ihre Antwort und stellen Sie sie in den Container DFHRESPONSE. Wenn es keine Antwort gibt, müssen Sie den Container DFHRESPONSE löschen.

Results

Die Antwort wird an den nächsten Handler in der Antwortphase der Pipeline übergeben. Der Handler wird für die Funktion SEND-RESPONSE aufgerufen. Wenn es keine Antwort gibt, wird der nächste Handler für die Funktion NO-RESPONSE aufgerufen.

Fehler beheben

Nachrichtenhandler sollten so konzipiert sein, dass sie Fehler beheben können, die in der Pipeline auftreten können.

About this task

Wenn in einem Nachrichtenhandler ein Fehler auftritt, wird das Programm erneut zur Fehlerbehandlung aufgerufen. Die Fehlerbehandlung findet immer in der Antwortphase der Pipeline statt. Wenn der Fehler in der Anforderungsphase aufgetreten ist, werden nachfolgende Handler in der Anforderungsphase übergangen.

In den meisten Fällen müssen Sie deshalb Ihr eigenes Handlerprogramm schreiben, um mögliche Fehler zu beheben.

Procedure

1. Prüfen Sie, dass der Container DFHFUNCTION den Wert HANDLER-ERROR enthält, was angibt, dass der Nachrichtenhandler für die Fehlerbehandlung aufgerufen wurde.

Tip:

- Wenn DFHFUNCTION einen anderen Wert enthält, wurde der Nachrichtenhandler nicht für die Fehlerbehandlung aufgerufen und die folgenden Schritte finden keine Anwendung.
2. Analysieren Sie die Fehlerinformationen und bestimmen Sie, ob der Nachrichtenhandler den Fehler beheben kann, indem er eine passende Antwort erstellt. Der Container DFHERROR enthält Informationen zu dem Fehler. Detaillierte Informationen zu diesem Container finden Sie unter „Container DFHERROR“ auf Seite 167.

Der Container DFHRESPONSE ist ebenfalls vorhanden und hat eine Länge von null.

3. Führen Sie alle Wiederherstellungsschritte aus.
 - Wenn der Nachrichtenhandler eine Wiederherstellung durchführen kann, erstellen Sie eine Antwort und geben Sie sie im Container DFHRESPONSE zurück.
 - Wenn der Nachrichtenhandler eine Wiederherstellung durchführen kann, aber keine Antwort erforderlich ist, löschen Sie den Container DFHRESPONSE und geben Sie stattdessen den Container DFHNORESPONSE zurück.
 - Wenn der Nachrichtenhandler keine Wiederherstellung durchführen kann, geben Sie den Container DFHRESPONSE unverändert (d. h. mit einer Länge von null) zurück.

Results

Wenn Ihr Nachrichtenhandler den Fehler beheben kann, wird die Pipelineverarbeitung normal fortgesetzt. Wenn nicht, generiert CICS einen SOAP-Fehler, der Informationen zu dem Fehler enthält. Im Fall eines Transaktionsabbruchs wird der Abbruchcode in den Fehler eingeschlossen.

Schnittstelle des Nachrichtenhandlers

Die CICS-Pipeline stellt eine Verknüpfung mit den Nachrichtenhandlern mithilfe eines Kanals her, der über eine Anzahl von Containern verfügt. Einige Container sind optional, andere sind für alle Nachrichtenhandler erforderlich und wieder andere werden von manchen Nachrichtenhandlern verwendet, von anderen nicht.

Bevor ein Handler aufgerufen wird, werden einige oder alle Container mit Informationen gefüllt, die der Handler für seine Arbeit verwenden kann. Die von dem Handler zurückgegebenen Container bestimmen die nachfolgende Verarbeitung und werden an später in der Pipeline vorhandene Handler übergeben.

SOAP-Nachrichtenhandler

Die SOAP-Nachrichtenhandler sind von CICS bereitgestellte Nachrichtenhandler, die Sie zur Verarbeitung von SOAP 1.1- und SOAP 1.2-Nachrichten in Ihre Pipeline einbinden können. Sie können die SOAP-Nachrichtenhandler in einem Service-Requester oder in einer Service-Provider-Pipeline verwenden.

Bei der Eingabe parsen die SOAP-Nachrichtenhandler eingehende SOAP-Nachrichten und extrahieren das SOAP-Element `<Body>` zur Verwendung durch Ihr Anwendungsprogramm. Bei der Ausgabe erstellen die Handler die vollständige SOAP-Nachricht, wobei sie das Element `<Body>` verwenden, das von Ihrer Anwendung bereitgestellt wird.

Wenn Sie SOAP-Header in Ihren Nachrichten verwenden, können die SOAP-Handler benutzergeschriebene Programme zur Headerverarbeitung aufrufen, die es Ihnen ermöglichen, die Header in eingehenden Nachrichten zu verarbeiten und zu ausgehenden Nachrichten hinzuzufügen.

SOAP-Nachrichtenhandler und alle Programme zur Headerverarbeitung werden in der Pipelinekonfigurationsdatei angegeben. Für Pipelines, die Java nicht unterstützen, müssen die Nachrichtenhandler `<cics_soap_1.1_handler>` oder `<cics_soap_1.2_handler>` angegeben werden. Für Pipelines, die Java unterstützen, müssen die Nachrichtenhandler `<cics_soap_1.1_handler_java>` oder `<cics_soap_1.2_handler_java>` angegeben werden.

In der Regel benötigen Sie nur einen SOAP-Handler in einer Pipeline. Es gibt jedoch Situationen, in denen mehr als ein Handler erforderlich ist. Sie können beispielsweise sicherstellen, dass SOAP-Header in einer bestimmten Reihenfolge verarbeitet werden, indem Sie mehrere SOAP-Handler definieren.

Sie dürfen die Nachrichtenhandler `<cics_soap_1.1_handler>` und `<cics_soap_1.2_handler>` oder `<cics_soap_1.1_handler_java>` und `<cics_soap_1.2_handler_java>` nicht in derselben Pipeline definieren. Wenn Ihre Pipeline sowohl SOAP 1.1- als auch SOAP 1.2-Nachrichten verarbeiten soll, sollten Sie den Nachrichtenhandler `<cics_soap_1.2_handler>` oder `<cics_soap_1.2_handler_java>` verwenden.

Programme zur Headerverarbeitung

Programme zur Headerverarbeitung sind benutzerdefinierte CICS-Programme, die von den von CICS bereitgestellten SOAP 1.1- und SOAP 1.2-Nachrichtenhandlern verknüpft werden, um SOAP-Headerblöcke zu verarbeiten.

Sie können Ihr Programm zur Headerverarbeitung in einer beliebigen von CICS unterstützten Sprache schreiben und jeden beliebigen CICS-Befehl im DPL-Subset verwenden. Ihr Programm zur Headerverarbeitung kann mit anderen CICS-Programmen verknüpft werden.

Die Programme zur Headerverarbeitung haben eine Kanalschnittstelle. Die Container enthalten Informationen, die das Headerprogramm überprüfen oder ändern kann, einschließlich des SOAP-Headerblocks, für den das Programm aufgerufen wird, und des SOAP-Nachrichtenhauptteils.

Der Kanal und die Container, die das Programm zur Headerverarbeitung verwenden kann, werden unter „Schnittstelle des Programms zur Headerverarbeitung“ auf Seite 162 beschrieben.

Andere Container enthalten Informationen über die Umgebung, in der das Headerprogramm aufgerufen wird, z. B.:

- Die Transaktions-ID, unter der das Headerprogramm aufgerufen wurde.
- Ob das Programm für eine Service-Provider- oder eine Service-Requester-Pipeline aufgerufen wurde.
- Ob die verarbeitete Nachricht eine Anforderung oder Antwort ist.

Programme zur Headerverarbeitung werden üblicherweise unter der Transaktion CPIH ausgeführt, die mit dem Attribut TASKDATALOC(ANY) definiert wird. Wenn Sie eine Programmverbindung verwenden, müssen Sie daher die Option AMODE(31) angeben.

Informationen zum Aufrufen von Programmen zur Headerverarbeitung für eine SOAP-Anforderung

Die Elemente <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> und <cics_soap_1.2_handler_java> in einer Pipelinekonfiguration enthalten kein, ein oder mehrere <headerprogram>-Elemente, die jeweils die folgenden untergeordneten Elemente enthalten:

- <program_name>
- <namespace>
- <localname>
- <mandatory>

Wenn eine Pipeline eine eingehende SOAP-Nachricht verarbeitet (eine Anforderung im Fall eines Service-Providers, eine Antwort im Fall eines Service-Requesters), wird das im Element <program_name> angegebene Headerprogramm aufgerufen oder nicht, abhängig von den folgenden Elementen:

- Dem Inhalt der Elemente <namespace>, <localname> und <mandatory>
- Dem Wert bestimmter Attribute des Stammelements des SOAP-Headers selbst (das Attribut **actor** für SOAP 1.1, das Attribut **role** für SOAP 1.2)

Die folgenden Regeln bestimmen, ob das Headerprogramm in einem gegebenen Fall aufgerufen wird:

Das Element <mandatory> in der Pipelinekonfigurationsdatei

Wenn das Element true (oder 1) enthält, wird das Programm zur Headerverarbeitung mindestens einmal aufgerufen, auch wenn keiner der Header in der SOAP-Nachricht für die Verarbeitung durch die übrigen Regeln ausgewählt wurde:

- Wenn kein Headerblock ausgewählt wird, wird das Programm zur Headerverarbeitung einmal aufgerufen.
- Wenn Headerblöcke durch die verbleibenden Regeln ausgewählt werden, wird das Programm zur Headerverarbeitung einmal für jeden ausgewählten Header aufgerufen.

Attribute im SOAP-Headerblock

Für SOAP 1.1 kann ein Headerblock nur dann verarbeitet werden, wenn das Attribut **actor** fehlt oder den Wert <http://schemas.xmlsoap.org/soap/actor/next> hat.

Für SOAP 1.2 kann ein Headerblock nur dann verarbeitet werden, wenn das Attribut **role** fehlt oder einen der folgenden Werte hat:

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

Ein Headerblock, der verarbeitet werden kann, wird nur dann verarbeitet, wenn er von der nächsten Regel ausgewählt wird.

Die Elemente <namespace> und <localname> in der Pipelinekonfigurationsdatei

Ein Headerblock, der laut der vorherigen Regel verarbeitet werden kann, wird nur dann zu Verarbeitung ausgewählt, wenn die folgenden Bedingungen erfüllt sind:

- Der Name des Stammelements des Headerblocks stimmt mit dem Element `<localname>` in der Pipelinekonfigurationsdatei überein.
- Der Namensbereich des Stammelements stimmt mit dem Element `<namespace>` in der Pipelinekonfigurationsdatei überein.

Sehen Sie sich beispielsweise diesen Headerblock an:

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

Vorbehaltlich der anderen Regeln wird der Headerblock zur Verarbeitung ausgewählt, wenn die folgenden Zeilen in der Pipelinekonfigurationsdatei codiert sind:

```
<namespace>http://mynamespace</namespace>
<localname>myheaderblock</localname>
```

Die `<localname>`-Elemente können ein `*` enthalten, um anzugeben, dass alle Headerblöcke in dem Namensbereich verarbeitet werden sollen. Aus diesem Grund wird derselbe Headerblock durch den folgenden Code ausgewählt:

```
<namespace>http://mynamespace</namespace>
<localname>*</localname>
```

Wenn die SOAP-Nachricht mehr als einen Header enthält, wird das Programm zur Headerverarbeitung einmal für jeden übereinstimmenden Header aufgerufen, aber die Reihenfolge, in der die Header verarbeitet werden, ist nicht definiert.

Die von CICS bereitgestellten SOAP-Nachrichtenhandler wählen die Programme zur Headerverarbeitung aus, die auf der Basis der Headerblöcke aufgerufen werden, die in der SOAP-Nachricht vorhanden sind, wenn der Nachrichtenhandler sie empfängt. Aus diesem Grund wird ein Programm zur Headerverarbeitung nie als Ergebnis eines Headerblocks aufgerufen, der einer Nachricht in demselben SOAP-Nachrichtenhandler hinzugefügt wird. Wenn Sie den neuen Header (oder alle geänderten Header) in Ihrer Pipeline verarbeiten wollen, müssen Sie einen anderen SOAP-Nachrichtenhandler in Ihrer Pipeline definieren.

Für eine ausgehende Nachricht (eine Anforderung in einem Service-Requester, eine Antwort in einem Service-Provider) erstellen von CICS bereitgestellte SOAP-Nachrichtenhandler eine SOAP-Nachricht, die keine Header enthält. Um einen oder mehrere Header zur Nachricht hinzuzufügen, müssen Sie ein Header-Handlerprogramm schreiben. Stellen Sie sicher, dass dieser Header-Handler aufgerufen wird, indem Sie ihn in Ihrer Pipelinekonfigurationsdatei definieren und `<mandatory>true</mandatory>` angeben.

Wenn ein Header-Handler in der Anforderungsphase einer Pipeline aufgerufen wird, wird er erneut in der Antwortphase aufgerufen, selbst wenn die Nachricht in der Antwortphase keinen passenden Header enthält.

Schnittstelle des Programms zur Headerverarbeitung

Die von CICS bereitgestellten SOAP 1.1- und SOAP 1.2-Nachrichtenhandler stellen eine Verknüpfung mit den Programmen zur Headerverarbeitung über den Kanal DFHHHC-V1 her. Zu den Containern, die an den Kanal übergeben werden, zählen diverse, die spezifisch für die Schnittstelle des Programms zur Headerverarbeitung sind, sowie Gruppen von *Kontextcontainern* und *Benutzercontainern*, die für alle Programme zur Headerverarbeitung und alle Nachrichtenhandlerprogramme in der Pipeline zugänglich sind.

Der Container DFHHEADER ist spezifisch für die Schnittstelle des Programms zur Headerverarbeitung. Andere Container sind an anderen Stellen in Ihrer Pipeline

verfügbar, haben aber einen bestimmten Zweck in einem Programm zur Headerverarbeitung. Die Container in dieser Kategorie sind DFHWS-XMLNS, DFHWS-BODY und DFHXMLSS-PARSE.

Anmerkung: Obwohl ein Web-Service, der Axis2 verwendet, um SOAP-Nachrichten zu verarbeiten, die Schnittstelle des Programms zur Headerverarbeitung nutzen kann, ist es effizienter, Ihre eigenen Axis2-Handler in Java zu schreiben, um die SOAP-Header zu verarbeiten. Weitere Informationen zum Erstellen von Axis2-Handlern finden Sie unter Writing Your Own Axis2 Module.

Container DFHHEADER

Wenn das Programm zur Headerverarbeitung aufgerufen wird, enthält DFHHEADER den einzelnen Headerblock, der die Ausführung des Programms zur Headerverarbeitung startet. Wenn das Headerprogramm mit `<mandatory>true</mandatory>` oder `<mandatory>1</mandatory>` in der Pipelinekonfigurationsdatei angegeben ist, wird es auch dann aufgerufen, wenn es keinen passenden Headerblock in der SOAP-Nachricht gibt. In diesem Fall hat der Container DFHHEADER eine Länge von null. Dies passiert, wenn ein Programm zur Headerverarbeitung aufgerufen wird, um einen Headerblock zu einer SOAP-Nachricht hinzuzufügen, die keine Headerblöcke hat.

Die SOAP-Nachricht, die CICS erstellt, hat ursprünglich keine Header. Wenn Sie Ihrer Nachricht Header hinzufügen möchten, müssen Sie sicherstellen, dass mindestens ein Programm zur Headerverarbeitung aufgerufen wird, indem Sie `<mandatory>true</mandatory>` oder `<mandatory>1</mandatory>` angeben.

Wenn das Headerprogramm zurückkehrt, muss der Container DFHHEADER keinen, einen oder mehrere Headerblöcke enthalten, die CICS anstelle des Originals in die SOAP-Nachricht einfügt:

- Sie können den ursprünglichen Headerblock unverändert zurückgeben.
- Sie können den Inhalt des Headerblocks ändern.
- Sie können einen oder mehrere neue Headerblöcke an den ursprünglichen Block anhängen.
- Sie können den ursprünglichen Headerblock durch einen oder mehrere andere Blöcke ersetzen.
- Sie können den Headerblock vollständig löschen.

Container DFHWS-XMLNS

Wenn das Programm zur Headerverarbeitung aufgerufen wird, enthält DFHWS-XMLNS Informationen zu XML-Namensbereichen, die im SOAP-Envelope deklariert sind. Das Headerprogramm kann diese Informationen verwenden, um die folgenden Tasks auszuführen:

- Auflösen qualifizierter Namen, die im Headerblock gefunden werden
- Erstellen qualifizierter Namen in neuen oder geänderten Headerblöcken

Die Namensbereichsinformationen bestehen aus einer Liste von Namensbereichsdeklarationen, die die XML-Standardnotation für das Deklarieren von Namensbereichen verwenden. Die Namensbereichsdeklarationen in DFHWS-XMLNS sind durch Leerzeichen getrennt. Beispiel:

```
xmlns:na='http://abc.example.org/schema' xmlns:nx='http://xyz.example.org/schema'
```

Sie können weitere Namensbereichsdeklarationen zum SOAP-Envelope hinzufügen, indem Sie sie an die Inhalte von DFHWS-XMLNS anhängen. Namensbereiche, deren Geltungsbereich ein SOAP-Headerblock oder ein SOAP-Hauptteil sind, werden jedoch am besten in dem Headerblock bzw. im Hauptteil selbst deklariert. Es wird empfohlen, keine Namensbereichsdeklarationen aus dem Container DFHWS-XMLNS in einem Programm zur Headerverarbeitung zu löschen, da XML-Elemente, die im Programm nicht sichtbar sind, möglicherweise von ihnen abhängen.

Container DFHWS-BODY

Dieser Container enthält den Hauptabschnitt des SOAP-Envelopes. Das Programm zur Headerverarbeitung kann diesen Inhalt ändern.

Wenn das Programm zur Headerverarbeitung aufgerufen wird, enthält DFHWS-BODY das SOAP-Element `<Body>`.

Wenn das Headerprogramm zurückgegeben wird, muss der Container DFHWS-BODY wieder einen gültigen SOAP-Hauptteil (`<Body>`) enthalten, den CICS anstelle des Originals in die SOAP-Nachricht einfügt:

- Sie können den ursprünglichen Hauptteil unverändert zurückgeben.
- Sie können den Inhalt des Hauptteils ändern.

Sie dürfen den SOAP-Hauptteil nicht vollständig löschen, da jede SOAP-Nachricht ein Element `<Body>` enthalten muss.

Container DFHXMLSS-PARSE

Wenn Sie das Element `<cics_soap_1.1_handler>` oder `<cics_soap_1.2_handler>` in Ihrer Pipelinekonfiguration verwenden und das Headerprogramm aufgerufen wird, enthält DFHXMLSS-PARSE die XMLSS-Datensätze (XML System Services) für diesen Header. Dieser Container wird nicht erstellt, wenn die Elemente `<cics_soap_1.1_handler_java>` oder `<cics_soap_1.2_handler_java>` verwendet werden.

Steuer-, Kontext- und Benutzercontainer

Neben den beschriebenen Containern übergibt die Schnittstelle die *Steuercontainer*, *Kontextcontainer* und *Benutzercontainer* im Kanal DFHHHC-V1.

Weitere Informationen zu diesen Containern finden Sie unter „In der Pipeline verwendete Container“ auf Seite 165.

Dynamisches Routing von eingehenden Anforderungen in einen Terminal-Handler

Wenn der Terminal-Handler einer Service-Provider-Pipeline einer der von CICS bereitgestellten SOAP-Nachrichtenhandler ist, kommt für das Zielanwendungshandlerprogramm, das im Container **DFHWS-APPHANDLER** angegeben wird, in einen Fällen das dynamische Routing in Frage. Die gesamte Pipelineverarbeitung vor dem Anwendungshandlerprogramm wird immer lokal in der CICS-Region ausgeführt, die die SOAP-Nachricht empfangen hat.

Die Transaktion, die das Zielanwendungshandlerprogramm ausführt, kommt für das Routing infrage, wenn eine der folgenden Bedingungen wahr ist:

- Die Transaktion, unter der die Pipeline die Nachricht verarbeitet, ist als DYNAMIC oder REMOTE definiert. Diese Transaktion ist in der URIMAP definiert, die zum Zuordnen des URI aus der eingehenden SOAP-Nachricht verwendet wird.

- Ein Programm in der Pipeline hat den Anfangswert des Inhalt des Containers **DFHWS-USERID** geändert.
- Ein Programm in der Pipeline hat den Anfangswert des Inhalt des Containers **DFHWS-TRANID** geändert.
- In der eingehenden SOAP-Nachricht gibt es einen WS-AT-SOAP-Header.

In allen vorhergehenden Szenarios tritt während der Pipelineverarbeitung ein Taskwechsel auf. Die zweite Task wird unter der Transaktion ausgeführt, die im Container **DFHWS-TRANID** angegeben ist. Dieser Taskwechsel bietet die Gelegenheit, dass dynamisches Routing stattfindet, aber nur, wenn MRO verwendet wird, um die CICS-Regionen miteinander zu verbinden. Darüber hinaus muss die CICS-Region, zu der Sie routen, Kanäle und Container unterstützen.

Das Routing findet nur statt, wenn die TRANSACTION-Definition für die in **DFHWS-TRANID** benannte Transaktion einen der folgenden Attributsätze angibt:

DYNAMIC(YES)

Die Transaktion wird mithilfe des verteilten Routing-Modells geroutet, in dem das Routing-Programm im Systeminitialisierungsparameter **DSRTPGM** angegeben ist.

DYNAMIC(NO) REMOTESYSTEM(*sysid*)

Die Transaktion wird an das durch *sysid* angegebene System geroutet.

Weitere Informationen zum Routing von Web-Service-Anforderungen finden Sie in der Technote Routing von CICS-Web-Services im Providermodus.

Für Anwendungen, die mit dem CICS-Web-Service-Assistenten implementiert werden, gibt es eine zweite Möglichkeit, die Anforderung dynamisch zu routen, und zwar an dem Punkt, an dem CICS mit dem Benutzerprogramm verknüpft ist. Die Anforderung wird mithilfe des dynamischen Routing-Modells geroutet, in dem das Routing-Programm im Systeminitialisierungsparameter **DTRPGM** angegeben ist. Die Eignung für das Routing wird in diesem Fall durch die Merkmale des Programms bestimmt. Wenn Sie beim Verknüpfen mit dem Programm einen Kanal und Container verwenden, können Sie die Anforderung nur dynamisch an CICS-Regionen mit Version 3.1 oder höher routen. Wenn Sie einen Kommunikationsbereich verwenden, gilt diese Einschränkung nicht.

Wenn eine Anforderung dynamisch an eine Zielregion geroutet wurde, kann sie nicht dynamisch vom Ziel an eine dritte Region geroutet werden, auch dann nicht, wenn die Transaktion als ROUTABLE(YES) und DYNAMIC(YES) definiert ist. Die Transaktion kann jedoch statistisch aus der Zielregion an eine dritte Region geroutet werden.

In der Pipeline verwendete Container

Eine Pipeline besteht in der Regel aus einer Reihe von Nachrichtenhandlerprogrammen und, wenn die von CICS bereitgestellten SOAP-Nachrichtenhandler verwendet werden, aus einer Reihe von Programmen zur Headerverarbeitung. CICS übergibt mithilfe von Containern Informationen zwischen diesen Programmen. Die Programme verwenden Container auch zur Kommunikation mit anderen Programmen in der Pipeline.

Die CICS-Pipeline stellt eine Verknüpfung mit den Nachrichtenhandlern und den Programmen zur Headerverarbeitung mithilfe eines Kanals her, der über eine Anzahl von Containern verfügt. Einige Container sind optional, andere sind für alle

Nachrichtenhandler erforderlich und wieder andere werden von manchen Nachrichtenhndlern verwendet, von anderen nicht.

Bevor ein Handler aufgerufen wird, werden einige oder alle Container mit Informationen gefüllt, die der Handler für seine Arbeit verwenden kann. Die von dem Handler zurückgegebenen Container bestimmen die nachfolgende Verarbeitung und werden an später in der Pipeline vorhandene Handler übergeben.

Die Container können auf folgende Weise kategorisiert werden:

Steuercontainer

Diese Container sind entscheidend für den Betrieb der Pipeline. Handler können die Steuercontainer verwenden, um die Reihenfolge zu ändern, in der sie verarbeitet werden. Die Namen der Steuercontainer werden von CICS definiert und beginnen mit den Zeichen DFH.

Kontextcontainer

Diese Container enthalten Informationen über die Umgebung, in der die Handler aufgerufen werden. CICS füllt diese Container mit Informationen, bevor es den ersten Nachrichtenhandler aufruft, aber in manchen Fällen können die Handler die Inhalte ändern oder die Container löschen. Änderungen an den Kontextcontainern wirken sich nicht direkt auf die Reihenfolge aus, in der die Handler aufgerufen werden. Die Namen der Kontextcontainer werden von CICS definiert und beginnen mit den Zeichen DFH.

Container des Programms zur Headerverarbeitung

Diese Container enthalten Informationen, die von Programmen zur Headerverarbeitung verwendet werden, die von den von CICS bereitgestellten SOAP-Nachrichtenhndlern aufgerufen werden. Informationen zu diesen Containern finden Sie unter Schnittstelle des Programms zur Headerverarbeitung.

Sicherheitscontainer

Diese Container enthalten Informationen, die von der vertrauenswürdigen Clientschnittstelle und dem Sicherheitsnachrichtenhandler verwendet werden, um Sicherheitstoken unter Verwendung eines Sicherheitstokenservice (STS) zu verarbeiten. Die Namen der Sicherheitscontainer werden von CICS definiert und beginnen mit den Zeichen DFH.

Generierte Container

Diese Container enthalten die Daten aus der SOAP-Nachricht, z. B. variable Arrays und lange Zeichenfolgen, die an das Anwendungsprogramm und von dem Anwendungsprogramm zur Verarbeitung übergeben werden. CICS erstellt diese Container automatisch während der Pipelineverarbeitung und ihre Namen beginnen mit den Zeichen DFH.

Benutzercontainer

Diese Container enthalten Informationen, die ein Nachrichtenhandler an einen anderen übergeben muss. Benutzercontainer werden ausschließlich von Nachrichtenhndlern verwendet. Sie können Ihre eigenen Namen für diese Container verwenden, aber diese Namen dürfen nicht mit der Zeichenfolge DFH beginnen.

Steuercontainer

Die Steuercontainer sind entscheidend für den Betrieb der Pipeline. Handler können die Steuercontainer verwenden, um die Reihenfolge zu ändern, in der sie verarbeitet werden.

Container DFHERROR:

DFHERROR ist ein Container vom Typ DATATYPE(BIT), der verwendet wird, um Informationen zu Pipelinefehlern an andere Nachrichtenhandler zu übergeben.

Tabelle 5. Struktur des Containers DFHERROR

Feldname	Länge (Byte)	Inhalt
PIISNEB-MAJOR-VERSION	1	„1“
PIISNEB-MINOR-VERSION	1	„1“
PIISNEB-ERROR-TYPE	1	Ein numerischer Wert, der den Typ des Fehlers angibt. Die Werte sind unter Tabelle 6 beschrieben.
PIISNEB-ERROR-MODE	1	P Der Fehler trat in einer Provider-Pipeline auf. R Der Fehler trat in einer Requester-Pipeline auf. T Der Fehler trat in einem vertrauenswürdigen Client auf.
PIISNEB-ABCODE	4	Der Abbruchcode, wenn der Fehler einem Transaktionsabbruch zugeordnet ist.
PIISNEB-ERROR-CONTAINER1	16	Der Name des Containers, wenn der Fehler einem Container zugeordnet ist.
PIISNEB-ERROR-CONTAINER2	16	Der Name des zweiten Containers, wenn der Fehler mehr als einem Container zugeordnet ist.
PIISNEB-ERROR-NODE	8	Der Name des Handlerprogramms, in dem der Fehler aufgetreten ist.

Tabelle 6. Werte für das Feld PIISNEB-ERROR-TYPE

Wert von PIISNEB-ERROR-TYPE	Bedeutung
1	Das Handlerprogramm ist fehlgeschlagen. Der Abbruchcode befindet sich im Feld PIISNEB-ABCODE.
2	Ein für den Handler erforderlicher Container war leer. Der Name des Containers befindet sich im Feld PIISNEB-ERROR-CONTAINER1.
3	Ein für den Handler erforderlicher Container hat gefehlt. Der Name des Containers befindet sich im Feld PIISNEB-ERROR-CONTAINER1.

Tabelle 6. Werte für das Feld PIISNEB-ERROR-TYPE (Forts.)

Wert von PIISNEB-ERROR-TYPE	Bedeutung
4	Zwei Container wurden an den Handler übergeben, obwohl nur einer erwartet wurde. Die Namen der Container befinden sich in den Feldern PIISNEB-ERROR-CONTAINER1 und PIISNEB-ERROR-CONTAINER2.
5	Der Versuch, eine Verknüpfung mit dem Zielprogramm herzustellen, ist fehlgeschlagen. Wenn das Zielprogramm fehlgeschlagen ist, befindet sich der Abbruchcode im Container PIISNEB-ABCODE.
6	Der Pipelinemanager konnte aufgrund eines Fehlers im zugrunde liegenden Transport nicht mit einem fernen Server kommunizieren.
7	Fehler im Container DFHWS-STSACTION. Er fehlt, ist fehlerhaft oder enthält einen falschen Wert.
8	DFHPIRT konnte die Pipeline nicht starten.
9	DFHPIRT konnte keine Nachricht in einen Container stellen.
10	DFHPIRT konnte keine Nachricht aus einem Container abrufen.
11	Ein nicht behandelter Fehler ist aufgetreten.

Die COBOL-Deklaration der Containerstruktur sieht so aus:

```
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).
```

In der folgenden Tabelle sind die Copybooks für die einzelnen Sprachen aufgeführt, die den Container zuordnen.

Tabelle 7. Copybooks, die den Container zuordnen

Sprache	Copybook
COBOL	DFHPIUCO
PL/I	DFHPIUCL
C und C++	dfhpiuch.h
Assembler	DFHPIUCD

Container DFHFUNCTON:

DFHFUNCTON ist ein Container vom Typ DATATYPE(CHAR), der eine 16-stellige Zeichenfolge enthält, die angibt, wo in einer Pipeline ein Programm aufgerufen wird.

Die Zeichenfolge hat einen der folgenden Werte. Die Zeichenpositionen ganz rechts werden mit Leerzeichen aufgefüllt.

RECEIVE-REQUEST

Bei dem Handler handelt es sich um einen Nicht-Terminal-Handler in einer Service-Provider-Pipeline. Er wird aufgerufen, um eine eingehende Anforderungsnachricht zu verarbeiten. Wenn die Nachricht bei dem Handler eingeht, befindet sie sich im Steuercontainer DFHREQUEST.

SEND-RESPONSE

Bei dem Handler handelt es sich um einen Nicht-Terminal-Handler in einer Service-Provider-Pipeline. Er wird aufgerufen, um eine ausgehende Antwortnachricht zu verarbeiten. Wenn die Nachricht bei dem Handler eingeht, befindet sie sich im Steuercontainer DFHRESPONSE.

SEND-REQUEST

Der Handler wird von einer Pipeline aufgerufen, die eine Anforderung sendet, d. h. in einem Service-Requester, der eine ausgehende Nachricht verarbeitet.

RECEIVE-RESPONSE

Der Handler wird von einer Pipeline aufgerufen, die eine Antwort empfängt, d. h. in einem Service-Requester, der eine eingehende Nachricht verarbeitet.

PROCESS-REQUEST

Der Handler wird als Terminal-Handler einer Service-Provider-Pipeline aufgerufen.

NO-RESPONSE

Der Handler wird nach der Verarbeitung einer Anforderung aufgerufen, wenn keine Antwort verarbeitet werden soll.

HANDLER-ERROR

Der Handler wird aufgerufen, weil ein Fehler festgestellt wurde.

In einer Service-Provider-Pipeline, die eine Anforderung verarbeitet und eine Antwort zurückgibt, sind die Werte von DFHFUNCTION RECEIVE-REQUEST, PROCESS-REQUEST und SEND-RESPONSE. Abb. 22 zeigt die Reihenfolge an, in der die Handler aufgerufen werden, sowie die Werte von DFHFUNCTION, die an die einzelnen Handler übergeben werden.

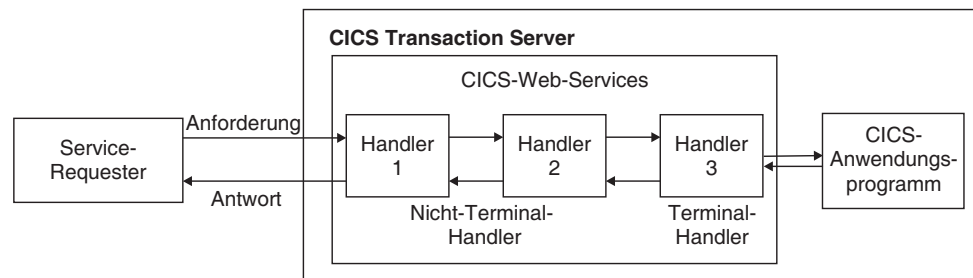


Abbildung 22. Reihenfolge von Handlern in einer Service-Provider-Pipeline

Reihenfolge	Handler	DFHFUNCTION
1	Handler 1	RECEIVE-REQUEST
2	Handler 2	RECEIVE-REQUEST

Reihenfolge	Handler	DFHFUNCTION
3	Handler 3	PROCESS-REQUEST
4	Handler 2	SEND-RESPONSE
5	Handler 1	SEND-RESPONSE

In einer Service-Requester-Pipeline, die eine Anforderung sendet und eine Antwort enthält, sind die Werte von DFHFUNCTION SEND-REQUEST und RECEIVE-RESPONSE. Abb. 23 zeigt die Reihenfolge an, in der die Handler aufgerufen werden, sowie die Werte von DFHFUNCTION, die an die einzelnen Handler übergeben werden.

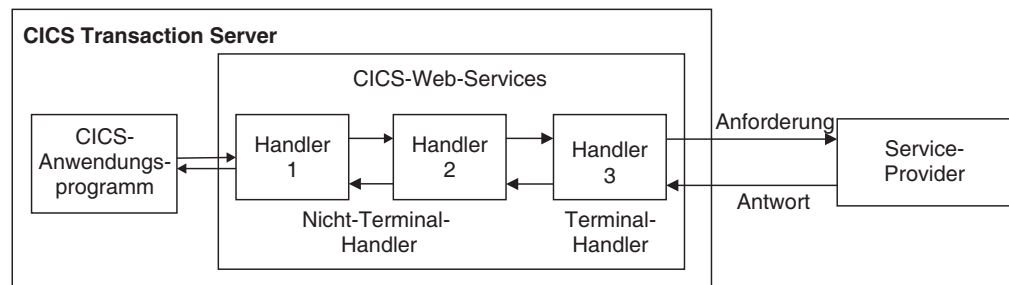


Abbildung 23. Reihenfolge von Handlern in einer Service-Requester-Pipeline

Reihenfolge	Handler	DFHFUNCTION
1	Handler 1	SEND-REQUEST
2	Handler 2	SEND-REQUEST
3	Handler 3	SEND-REQUEST
4	Handler 3	RECEIVE-RESPONSE
5	Handler 2	RECEIVE-RESPONSE
6	Handler 1	RECEIVE-RESPONSE

Die Werte von DFHFUNCTION, die ein gegebener Nachrichtenhandler annehmen kann, hängen davon ab, ob die Pipeline ein Provider oder Requester ist, ob sie sich in der Anforderungs- oder der Antwortphase befindet und ob es sich bei dem Handler um einen Terminal-Handler oder einen Nicht-Terminal-Handler handelt. In der folgenden Tabelle sind die einzelnen Werte zusammengefasst:

Wert von DFHFUNCTION	Provider- oder Requester-Pipeline	Pipelinephase	Terminal- oder Nicht-Terminal-Handler
RECEIVE-REQUEST	Provider	Anforderungsphase	Nicht-Terminal
SEND-RESPONSE	Provider	Antwortphase	Nicht-Terminal
SEND-REQUEST	Requester	Anforderungsphase	Nicht-Terminal
RECEIVE-RESPONSE	Requester	Antwortphase	Nicht-Terminal
PROCESS-REQUEST	Provider	Anforderungsphase	Terminal
NO-RESPONSE	Beide	Antwortphase	Nicht-Terminal
HANDLER-ERROR	Beide	Beide	Beide

Container DFHHTTPMETHOD:

Dies ist ein Container vom Typ DATATYPE(CHAR), der für Anwendungsprogramme in allen CICS-Pipelines im HTTP-Providermodus verfügbar ist.

Dieser Container ist acht Zeichen lang und enthält den Namen der HTTP-Methode, die für die eingehende Anforderung verwendet wurde. Er wird nicht gefüllt, wenn die Anforderung nicht über HTTP empfangen wurde.

Container DFHHTTPSTATUS:

DFHHTTPSTATUS ist ein Container vom Typ DATATYPE(CHAR), mit dem der HTTP-Statuscode und der Statustext einer Nachricht angegeben wird, die in der Antwortphase einer Service-Provider-Pipeline generiert wurde.

Der Inhalt des Containers DFHHTTPSTATUS muss derselbe sein wie die ursprüngliche Statuszeile einer HTTP-Antwortnachricht, die die folgende Struktur aufweist:

HTTP/1.1 nnn tttttttt

HTTP/1.1

Version und Release von HTTP.

nnn Dreistelliger dezimaler HTTP-Statuscode, der zurückgegeben werden soll.

tttttttt

Der lesbare Statustext, der dem Statuscode nnn zugeordnet ist.

Die folgende Zeichenfolge ist ein Beispiel für den Inhalt:

HTTP/1.1 412 Precondition Failed

Der Container DFHHTTPSTATUS wird ignoriert, wenn die Pipeline den WebSphere MQ-Transport verwendet.

Wenn der Container mehr als 45 Bytes Daten enthält, sendet CICS 45 Bytes und ignoriert die verbleibenden Daten.

Container DFHMEDIATYPE:

DFHMEDIATYPE ist ein Container vom Typ DATATYPE(CHAR), mit dem der Medientyp für eine Nachricht angegeben wird, die in der Antwortphase einer Service-Provider-Pipeline erzeugt wurde.

Der Inhalt des Containers DFHMEDIATYPE muss aus einem Typ und einem Subtyp bestehen, die durch einen Schrägstrich getrennt sind. Die folgenden beiden Zeichenfolgen sind Beispiele für einen korrekten Inhalt für den Container DFHMEDIATYPE:

text/plain

image/svg+xml

Der Container DFHMEDIATYPE wird ignoriert, wenn die Pipeline den WebSphere MQ-Transport verwendet.

Container DFHNORESPONSE:

DFHNORESPONSE ist ein Container vom Typ DATATYPE(CHAR), der in der Anforderungsphase einer Service-Requester-Pipeline angibt, dass vom Service-Provider nicht erwartet wird, dass er eine Antwort zurückgibt.

Der Inhalt des Containers DFHNORESPONSE ist nicht definiert. Nachrichtenhandler, die wissen müssen, ob der Service-Provider eine Antwort zurückgeben soll, müssen nur feststellen, ob der Container vorhanden ist oder nicht:

- Wenn der Container DFHNORESPONSE vorhanden ist, wird keine Antwort erwartet.
- Wenn der Container DFHNORESPONSE fehlt, *wird* eine Antwort erwartet.

Diese Informationen werden anfänglich von der Service-Requester-Anwendung bereitgestellt, basierend auf dem mit dem Service-Provider verwendeten Protokoll. Deshalb wird davon abgeraten, diesen Container in einem Nachrichtenhandler zu löschen (oder ihn zu erstellen, falls er nicht vorhanden ist), weil dies das Protokoll zwischen den Endpunkten stören kann.

Abgesehen von der Anforderungsphase einer Service-Requester-Pipeline ist die Verwendung dieses Containers nicht definiert.

Container DFHREQUEST:

DFHREQUEST ist ein Container vom Typ DATATYPE(CHAR), der die Anforderungsnachricht enthält, die in der Anforderungsphase einer Pipeline verarbeitet wird.

Wenn einer der von CICS bereitgestellten SOAP-Nachrichtenhandler in der Pipeline konfiguriert ist, wird der Container DFHREQUEST aktualisiert, um die SOAP-Nachrichtenheader in den SOAP-Envelope einzuschließen. Wenn die Nachricht von einem durch CICS bereitgestellten SOAP-Nachrichtenhandler erstellt und im Anschluss nicht geändert wird, enthält DFHREQUEST einen vollständigen SOAP-Envelope, dessen Inhalt in der UTF-8-Codepage vorliegt.

Der Container DFHREQUEST ist in der Anforderung vorhanden, wenn ein Nachrichtenhandler aufgerufen wird, und der Container DFHFUNCTION RECEIVE-REQUEST oder SEND-REQUEST enthält.

In diesem Fall soll das normale Protokoll DFHREQUEST an die Pipeline mit denselben oder geänderten Inhalten zurückgeben. Die Verarbeitung der Pipelineanforderungsphase wird normal fortgesetzt, mit dem nächsten Nachrichtenhandlerprogramm in der Pipeline, wenn ein solches vorhanden ist.

Alternativ kann Ihr Nachrichtenhandler den Container DFHREQUEST löschen und eine Antwort in den Container DFHRESPONSE stellen. Auf diese Weise wird die normale Reihenfolge der Verarbeitung umgekehrt und die Verarbeitung startet mit der Antwortphase der Pipeline.

Container DFHRESPONSE:

DFHRESPONSE ist ein Container vom Typ DATATYPE(CHAR), der die Antwortnachricht enthält, die in der Antwortphase einer Pipeline verarbeitet wird. Wenn die Nachricht von einem durch CICS bereitgestellten SOAP-Nachrichtenhandler erstellt und im Anschluss nicht geändert wurde, enthält DFHRESPONSE einen vollständigen SOAP-Envelope, dessen Inhalt in der UTF-8-Codepage vorliegt.

Der Container DFHRESPONSE ist vorhanden, wenn ein Nachrichtenhandler aufgerufen wird, und der Container DFHFUNCTION SEND-RESPONSE oder RECEIVE-RESPONSE enthält.

In diesem Fall soll das normale Protokoll DFHRESPONSE an die Pipeline mit denselben oder geänderten Inhalten zurückgeben. Die Pipelineverarbeitung wird normal fortgesetzt, mit dem nächsten Nachrichtenhandlerprogramm in der Pipeline, wenn ein solches vorhanden ist.

Der Container DFHRESPONSE ist ebenfalls vorhanden, mit einer Länge von null, wenn DFHFUNCTION RECEIVE-REQUEST, SEND-REQUEST, PROCESS-REQUEST oder HANDLER-ERROR enthält.

Container DFHWS-CCSID:

DFHWS-CCSID ist ein Container vom Typ DATATYPE(BIT), der ein Vollwort (4 Bytes) enthält, das die CCSID der Daten im Antwortcontainer angibt.

Der Container ist nur für eine Pipeline im Providermodus gültig, die CICS-Code verwendet, um die Sprachstruktur in XML umzusetzen.

Die CCSID muss mit der CCSID kompatibel sein, die zum Generieren der WS-BIND-Datei verwendet wird. Ist sie dies nicht, kann die erzeugte SOAP-Antwort falsche oder ungültige Zeichen enthalten.

Die CCSID darf nicht in oder von 930, 1390, 5026 und 1026 geändert werden. CICS lässt außerdem nicht zu, dass die CCSID so geändert wird, dass sie als Client-CCSID verwendet werden kann.

Wenn bei der Verarbeitung des Werts im Container DFHWS-CCSID Probleme auftreten, wird die Verarbeitung unter Verwendung der CCSID aus der WSBIND-Datei fortgesetzt.

Der Container DFHWS-CCSID wird nur bei Rückgabe aus einem kanalgesteuerten Anwendungsprogramm geprüft.

Container DFHWS-NODEJSAPP:

DFHWS-NODEJSAPP ist ein Container vom Typ DATATYPE(CHAR), der den Namen der NODEJSAPP-Ressource für diese Pipeline enthält.

DFHWS-NODEJSAPP ist ein Container vom Typ DATATYPE(CHAR), der den Namen der NODEJSAPP-Ressource für diese Pipeline enthält. Der Container ist nur für eine Providermodus-Pipeline gültig und nur dann, wenn die Pipeline über eine NODEJSAPP-Ressource aufgerufen wird.

Steuerung der Pipelineprotokolle durch Container

Die Inhalte der Container DFHFUNCTION, DFHREQUEST und DFHRESPONSE steuern gemeinsam die Pipelineprotokolle.

Während der zwei Phasen der Ausführung einer Pipeline (Anforderungsphase und Antwortphase) bestimmt der Wert von DFHFUNCTION, welche Steuercontainer an die einzelnen Nachrichtenhandler übergeben werden:

DFHFUNCTION	Kontext	DFHREQUEST	DFHRESPONSE
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Vorhanden (Länge > 0)	Vorhanden (Länge = 0)
SEND-RESPONSE	Service-Provider; Antwortphase	Nicht vorhanden	Vorhanden (Länge > 0)
SEND-REQUEST	Service-Requester; Anforderungsphase	Vorhanden (Länge > 0)	Vorhanden (Länge = 0)
RECEIVE-RESPONSE	Service-Requester; Antwortphase	Nicht vorhanden	Vorhanden (Länge > 0)
PROCESS-REQUEST	Service-Provider; Terminal-Handler	Vorhanden (Länge > 0)	Vorhanden (Länge = 0)
HANDLER-ERROR	Service-Requester oder -Provider; beide Phasen	Nicht vorhanden	Vorhanden (Länge = 0)
NO-RESPONSE	Service-Requester oder -Provider; Antwortphase	Nicht vorhanden	Nicht vorhanden

Die nachfolgende Verarbeitung wird von den Containern bestimmt, die Ihr Nachrichtenhandler an die Pipeline zurückgibt:

Während der Anforderungsphase

- Ihr Nachrichtenhandler kann den Container DFHREQUEST zurückgeben. Die Verarbeitung wird in der Anforderungsphase mit dem nächsten Handler fortgesetzt. Die Länge der Daten im Container darf nicht null sein.
- Ihr Nachrichtenhandler kann den Container DFHRESPONSE zurückgeben. Die Verarbeitung wechselt in die Antwortphase und derselbe Handler wird aufgerufen, wobei DFHFUNCTION auf SEND-RESPONSE in einem Service-Provider und auf RECEIVE-RESPONSE in einem Service-Requester gesetzt ist. Die Länge der Daten im Container darf nicht null sein.
- Ihr Nachrichtenhandler kann keine Container zurückgeben. Die Verarbeitung wechselt in die Antwortphase und derselbe Handler wird aufgerufen, wobei DFHFUNCTION auf NO-RESPONSE gesetzt ist.

Im Terminal-Handler (nur Service-Provider)

- Ihr Nachrichtenhandler kann den Container DFHRESPONSE zurückgeben. Die Verarbeitung wechselt in die Antwortphase und der vorherige Handler wird mit einem neuen Wert von DFHFUNCTION (SEND-RESPONSE) aufgerufen. Die Länge der Daten im Container darf nicht null sein.
- Ihr Nachrichtenhandler kann keine Container zurückgeben. Die Verarbeitung wechselt in die Antwortphase und der vorherige Handler wird mit einem neuen Wert von DFHFUNCTION (NO-RESPONSE) aufgerufen.

Während der Antwortphase

- Ihr Nachrichtenhandler kann den Container DFHRESPONSE zurückgeben. Die Verarbeitung wird in der Antwortphase fortgesetzt und der nächste Handler wird aufgerufen. Die Länge der Daten im Container darf nicht null sein.

- Ihr Nachrichtenhandler kann keine Container zurückgeben. Die Verarbeitung wird in der Antwortphase fortgesetzt und der nächste Handler in der Abfolge wird mit einem neuen Wert von DFHFUNCTION (NO-RESPONSE) aufgerufen.

Important: Während der Anforderungsphase kann Ihr Nachrichtenhandler DFHREQUEST oder DFHRESPONSE zurückgeben, aber nicht beides. Da beide Container vorhanden sind, wenn Ihr Nachrichtenhandler aufgerufen wird, müssen Sie einen löschen.

In dieser Tabelle ist die Aktion aufgeführt, die von der Pipeline für alle Werte von DFHFUNCTION ausgeführt wird, und es sind alle Kombinationen von DFHREQUEST und DFHRESPONSE angegeben, die von den einzelnen Nachrichtenhandler zurückgegeben werden.

DFHFUNCTION	Kontext	DFHREQUEST	DFHRESPONSE	Aktion
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Vorhanden (Länge > 0)	Vorhanden	(Fehler)
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Vorhanden (Länge > 0)	Nicht vorhanden	Aufrufen des nächsten Handlers mit der Funktion RECEIVE-REQUEST
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Vorhanden (Länge = 0)	Nicht zutreffend	(Fehler)
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Nicht vorhanden	Vorhanden (Länge > 0)	Wechseln in die Antwortphase und Aufrufen desselben Handlers mit der Funktion SEND-RESPONSE
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Nicht vorhanden	Vorhanden (Länge = 0)	(Fehler)
RECEIVE-REQUEST	Service-Provider; Anforderungsphase	Nicht vorhanden	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE
SEND-RESPONSE	Service-Provider; Antwortphase	Nicht zutreffend	Vorhanden (Länge > 0)	Aufrufen des vorherigen Handlers mit der Funktion SEND-RESPONSE
SEND-RESPONSE	Service-Provider; Antwortphase	Nicht zutreffend	Vorhanden (Länge = 0)	(Fehler)
SEND-RESPONSE	Service-Provider; Antwortphase	Nicht zutreffend	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE
SEND-REQUEST	Service-Requester; Anforderungsphase	Vorhanden (Länge > 0)	Vorhanden (Länge ≥ 0)	(Fehler)
SEND-REQUEST	Service-Requester; Anforderungsphase	Vorhanden (Länge > 0)	Nicht vorhanden	Aufrufen des nächsten Handlers mit der Funktion SEND-REQUEST
SEND-REQUEST	Service-Requester; Anforderungsphase	Vorhanden (Länge = 0)	Nicht zutreffend	(Fehler)
SEND-REQUEST	Service-Requester; Anforderungsphase	Nicht vorhanden	Vorhanden (Länge > 0)	Wechseln in die Antwortphase und Aufrufen des vorherigen Handlers mit der Funktion RECEIVE-RESPONSE
SEND-REQUEST	Service-Requester; Anforderungsphase	Nicht vorhanden	Vorhanden (Länge = 0)	(Fehler)
SEND-REQUEST	Service-Requester; Anforderungsphase	Nicht vorhanden	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE
RECEIVE-RESPONSE	Service-Requester; Antwortphase	Nicht zutreffend	Vorhanden (Länge > 0)	Aufrufen des vorherigen Handlers mit der Funktion RECEIVE-RESPONSE

DFHFUNCTION	Kontext	DFHREQUEST	DFHRESPONSE	Aktion
RECEIVE-RESPONSE	Service-Requester; Antwortphase	Nicht zutreffend	Vorhanden (Länge = 0)	(Fehler)
RECEIVE-RESPONSE	Service-Requester; Antwortphase	Nicht zutreffend	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE
PROCESS-REQUEST	Service-Provider; Terminal-Handler	Nicht zutreffend	Vorhanden (Länge > 0)	Aufrufen des vorherigen Handlers mit der Funktion RECEIVE-RESPONSE
PROCESS-REQUEST	Service-Provider; Terminal-Handler	Nicht zutreffend	Vorhanden (Länge = 0)	(Fehler)
PROCESS-REQUEST	Service-Provider; Terminal-Handler	Nicht zutreffend	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE
HANDLER-ERROR	Service-Requester oder -Provider; beide Phasen	Nicht zutreffend	Vorhanden (Länge > 0)	Aufrufen des vorherigen Handlers mit der Funktion SEND-RESPONSE function
HANDLER-ERROR	Service-Requester oder -Provider; beide Phasen	Nicht zutreffend	Vorhanden (Länge = 0)	(Fehler)
HANDLER-ERROR	Service-Requester oder -Provider; beide Phasen	Nicht zutreffend	Nicht vorhanden	Aufrufen desselben Handlers mit der Funktion NO-RESPONSE

Kontextcontainer

In einigen Fällen benötigen vom Benutzer geschriebene Nachrichtenhandlerprogramme und Programme zur Headerverarbeitung Informationen über den Kontext, in dem sie aufgerufen werden. CICS stellt diese Informationen in einem Satz von *Kontextcontainern* bereit, der an die Programme übergeben wird.

CICS initialisiert die Inhalte des Kontextcontainers, aber in manchen Fällen können Sie die Inhalte in Ihren Nachrichtenhandlerprogrammen und Programmen zur Headerverarbeitung ändern. Wenn z. B. in einer Service-Provider-Pipeline der Terminal-Handler einer der von CICS bereitgestellten SOAP-Handler ist, können Sie die Benutzer-ID und die Transaktions-ID des Zielanwendungsprogramms ändern, indem Sie die Inhalte der entsprechenden Kontextcontainer ändern.

Manche der in den Containern bereitgestellten Informationen gelten nur für einen Service-Provider oder nur für einen Service-Requester, weswegen einige der Kontextcontainer nicht in beiden verfügbar sind.

Container DFH-EXIT-HEADER1:

DFH-EXIT-HEADER1 ist ein Container vom Typ DATATYPE(CHAR). Er enthält einen oder mehrere SOAP-Header, die zu einer Antwort aus einer Web-Service-Provider-Anwendung in CICS hinzugefügt werden.

Programme, die den globalen Benutzerexit XWSPRRWO ausführen, können einen Header zu einer SOAP-Antwort hinzufügen. Der Header muss ein gültiges SOAP-Element sein und die Namensbereiche müssen in der Header-XML eigenständig sein. Ein Programm, das Daten in diesen Container stellt, muss prüfen, ob der Container vorhanden ist, und den neuen Header am Ende der Daten hinzufügen. Wenn Sie dieses bewährte Verfahren befolgen, können mehrere Programme bei Bedarf an demselben Exitpunkt gesteuert werden.

Container DFH-HANDLERPLIST:

DFH-HANDLERPLIST ist ein Container vom Typ DATATYPE(CHAR), der mit den Inhalten des entsprechenden <handler_parameter_list>-Elements der Pipelinekonfigurationsdatei initialisiert wird.

Wenn Sie keine Handlerparameterliste in der Pipelinekonfigurationsdatei angegeben haben, ist der Container leer, d. h. er hat eine Länge von null.

Sie können die Inhalte dieses Containers nicht ändern.

Container DFH-SERVICEPLIST:

DFH-SERVICEPLIST ist ein Container vom Typ DATATYPE(CHAR), der den Inhalt des Elements <service_parameter_list> der Pipelinekonfigurationsdatei enthält.

Wenn Sie keine Serviceparameterliste in der Pipelinekonfigurationsdatei angegeben haben, ist der Container leer, d. h. er hat eine Länge von null.

Sie können die Inhalte dieses Containers nicht ändern.

Container DFHWS-APPHANDLER:

DFHWS-APPHANDLER ist ein Container vom Typ DATATYPE(CHAR), der in einer Service-Provider-Pipeline mit dem Inhalt des Elements <apphandler> der Pipelinekonfigurationsdatei initialisiert wird.

Im Terminal-Handler einer Pipeline, die das Element <apphandler> enthält, rufen die bereitgestellten SOAP-Handler den Namen des Zielanwendungsprogramms aus diesem Container ab.

Sie können den Inhalt dieses Containers in Ihren Nachrichtenhandlern oder Programmen zur Headerverarbeitung ändern.

CICS stellt diesen Container nicht in einer Service-Requester-Pipeline bereit.

Related concepts:

„Anwendungshandler“ auf Seite 106

Ein Anwendungshandler ist ein CICS-Programm, mit dem der Terminal-Handler einer SOAP-Service-Provider-Pipeline zur Laufzeit verknüpft wird.

Container DFHWS-APPHANCLAS:

DFHWS-APPHANCLAS ist ein Container vom Typ DATATYPE(CHAR), der in einer Service-Provider-Pipeline mit dem Inhalt des Elements <apphandler_class> der Pipelinekonfigurationsdatei initialisiert wird.

Im Terminal-Handler einer Java-basierten Pipeline rufen die bereitgestellten SOAP-Handler, <cics_soap_1.1_handler_java> und <cics_soap_1.2_handler_java>, den Namen des Zielanwendungsprogramms aus diesem Container ab.

CICS stellt diesen Container nicht in einer Service-Requester-Pipeline bereit.

Related concepts:

„Anwendungshandler“ auf Seite 106

Ein Anwendungshandler ist ein CICS-Programm, mit dem der Terminal-Handler einer SOAP-Service-Provider-Pipeline zur Laufzeit verknüpft wird.

Related reference:

„Pipelinekonfigurationselement <apphandler_class>“ auf Seite 108

Gibt an, dass der Terminal-Handler der Pipeline eine Verknüpfung zu einem Axis2-Anwendungshandler einrichtet.

Container DFHWS-DATA:

DFHWS-DATA ist ein Container vom Typ DATATYPE(BIT), der in Service-Requester-Anwendungen und optional in Service-Provider-Anwendungen verwendet wird, die mit dem CICS-Web-Service-Assistenten implementiert werden. Er enthält die übergeordnete Datenstruktur, die einer SOAP-Anforderung zugeordnet ist und von dieser zugeordnet wird.

In Service-Requester-Anwendungen muss der Container DFHWS-DATA vorhanden sein, wenn das Service-Requester-Programm einen Befehl **EXEC CICS INVOKE SERVICE** ausgibt. Wenn der Befehl ausgegeben wird, konvertiert CICS die Datenstruktur, die sich in dem Container befindet, in eine SOAP-Anforderung. Wenn die SOAP-Antwort empfangen wird, konvertiert CICS sie in eine andere Datenstruktur, die an die Anwendung in demselben Container zurückgegeben wird.

In Service-Provider-Anwendungen wird der Container DFHWS-DATA standardmäßig verwendet, wenn Sie den Parameter **CONTID** nicht in den Stapeljobs DFHLS2WS oder DFHWS2LS angeben. CICS konvertiert die SOAP-Anforderungsnachricht in die Datenstruktur, die an die Anwendung im Container DFHWS-DATA übergeben wird. Die Antwort wird dann in demselben Container gespeichert und CICS konvertiert die Datenstruktur in eine SOAP-Antwortnachricht.

Container DFHWS-FAULT:

DFHWS-FAULT ist ein Container vom Typ DATATYPE(BIT), der Informationen zum Typ des SOAP-Fehlers enthält, den CICS generiert.

Der Container enthält ein binäres Vollwort, das den Fehlertyp angibt, der in der weiteren Verarbeitung für eine Web-Service-Antwort verwendet werden kann:

1. Der zuletzt aufgetretene SOAP-Fehler war ein CICS-Fehler (z. B. ein Abbruch durch CICS oder den Benutzer).
2. Der zuletzt aufgetretene SOAP-Fehler war ein Anwendungsfehler. Der Container wird gelöscht, wenn Sie den Befehl **EXEC CICS SOAPFAULT DELETE** absetzen. Wenn ein zweiter oder neuer SOAP-Fehler erstellt wird, aktualisiert CICS den neuen Container entsprechend.

Sie können die Inhalte dieses Containers nicht ändern.

Container DFHWS-LOCATION:

DFHWS-LOCATION ist ein Container vom Typ DATATYPE(CHAR), der den bereitgestellten Positionsholder enthält, wenn die HTTP-Antwort 302, 303 oder 307 war.

Container DFHWS-MEP:

DFHWS-MEP ist ein Container vom Typ DATATYPE(BIT), der einen repräsentativen Wert für das Nachrichtenaustauschmuster (MEP - Message Exchange Pattern) einer ein- oder ausgehenden SOAP-Nachricht enthält. Dieser Wert ist 1 Byte lang.

CICS unterstützt vier Nachrichtenaustauschmuster (Message Exchange Pattern, MEP) für Service-Requester und Service-Provider. Das Nachrichtenaustauschmuster ist im WSDL 2.0-Dokument für den Web-Service definiert und bestimmt, ob CICS als Provider antwortet und ob CICS eine Antwort von einem externen Provider erwartet. Im Requestermodus wird die Zeit, in der CICS auf eine Antwort wartet, mithilfe der PIPELINE-Ressource konfiguriert.

Wenn Sie den CICS-Web-Service-Assistenten verwendet haben, um Ihre Anwendung zu implementieren, wird dieser Container von CICS gefüllt:

- In einer Service-Provider-Pipeline wird dieser Container von dem DFHPITP-Anwendungshandler gefüllt, wenn er die eingehende Nachricht vom Terminal-Handler empfängt.
- In einer Service-Requester-Pipeline wird dieser Container gefüllt, wenn die Anwendung den Befehl **INVOKE SERVICE** ausführt.

Wenn die Anwendung den Kanal DFHPIRT zum Starten der Pipeline verwendet, füllt die Anwendung diesen Container. Wenn der Container nicht vorhanden oder leer ist, geht CICS davon aus, dass die Anforderung entweder das In-Out- oder das In-Only-MEP verwendet, abhängig davon, ob der Container DFHNORESPONSE im Kanal vorhanden ist.

Dieser Container wird von dem bereitgestellten Anwendungshandlerprogramm, DFHPITP, gefüllt. Wenn Sie einen anderen Anwendungshandler verwenden, kann dieser Container nicht verwendet werden.

Tabelle 8. Mögliche Werte im Container DFHWS-MEP

Wert	MEP	URI
1	in-only	http://www.w3.org/ns/wsdli/in-only
2	in-out	http://www.w3.org/ns/wsdli/in-out
4	robust in-only	http://www.w3.org/ns/wsdli/robust-in-only
8	in-optional-out	http://www.w3.org/ns/wsdli/in-opt-out

Container DFHWS-OPERATION:

DFHWS-OPERATION ist ein Container vom Typ DATATYPE(CHAR), der in der Regel in einer Service-Provider-Anwendung verwendet wird, die mit dem CICS-Web-Service-Assistenten implementiert wird. Er enthält den Namen der Operation, die in einer SOAP-Anforderung angegeben wird.

In einem Service-Provider gibt der Container den Namen der Operation an, für die die Anwendung aufgerufen wird. Er wird gefüllt, wenn ein bereitgestellter SOAP-Nachrichtenhandler die Steuerung an das Zielanwendungsprogramm übergibt, und ist nur dann sichtbar, wenn das Zielpogramm über eine Kanalschnittstelle aufgerufen wird.

In einer Service-Requester-Pipeline enthält der Container den in der Option OPERATION angegebenen Namen des Befehls **EXEC CICS INVOKE SERVICE**. Der Container ist nicht verfügbar für die Anwendung, die den Befehl ausgibt.

Dieser Container wird von dem bereitgestellten Anwendungshandlerprogramm, DFHPITP, gefüllt. Wenn Sie einen anderen Anwendungshandler verwenden, kann dieser Container nicht verwendet werden.

Container DFHWS-PIPELINE:

DFHWS-PIPELINE ist ein Container vom Typ DATATYPE(CHAR), der den Namen der PIPELINE enthält, in der das Programm ausgeführt wird.

Sie können die Inhalte dieses Containers nicht ändern.

Container DFHWS-RESPWAIT:

DFHWS-RESPWAIT ist ein Container vom Typ DATATYPE(BIT), der eine nicht signierte Vollwort-Binärzahl enthält, die das Zeitlimit in Sekunden darstellt, das für ausgehende Web-Service-Anforderungs- und -Antwortnachrichten gilt.

Der Wert dieses Containers wird von dem Attribut RESPWAIT der PIPELINE-Definition bereitgestellt und von CICS festgelegt, wenn der Befehl INVOKE SERVICE ausgegeben wird. Ein Wert, der in diesem Container von der Benutzeranwendung festgelegt wird, bevor der Befehl INVOKE SERVICE ausgeführt wurde, wird ignoriert.

Ein Nachrichtenhandlerprogramm, das während der Pipelineverarbeitung aufgerufen wird, kann den Wert des Containers DFHWS-RESPWAIT überschreiben. Ist dies der Fall, wird der aktualisierte Wert nur verwendet, wenn die PIPELINE-Definition das Attribut RESPWAIT hat, das nicht auf DEFT festgelegt wurde oder leer ist. Wenn das Attribut RESPWAIT in der PIPELINE-Definition auf DEFT festgelegt wurde oder leer ist, wird immer das Standardzeitlimit des Transportprotokolls verwendet, unabhängig von dem Wert im Container DFHWS-RESPWAIT.

Dieser Container wird nur in Pipelines im Requestermodus verwendet.

Container DFHWS-SOAPLEVEL:

DFHWS-SOAPLEVEL ist ein Container vom Typ DATATYPE(BIT), der Informationen zur SOAP-Version enthält, die in der verarbeiteten Nachricht verwendet wird.

Der Container enthält ein binäres Vollwort, das die SOAP-Version angibt, die für eine Web-Service-Anforderung oder -Antwort verwendet wird:

- 1 Die Anforderung oder Antwort ist eine SOAP 1.1-Nachricht.
- 2 Die Anforderung oder Antwort ist eine SOAP 1.2-Nachricht.
- 10 Die Anforderung oder Antwort ist keine SOAP-Nachricht.

Sie können die Inhalte dieses Containers nicht ändern.

Container DFHWS-TRANID:

DFHWS-TRANID ist ein Container vom Typ DATATYPE(CHAR), der mit der Transaktions-ID der Task initialisiert wird, in der die Pipeline ausgeführt wird.

Wenn Sie den Inhalt dieses Containers in einer Service-Provider-Pipeline ändern, in der der Terminal-Handler einer der von CICS bereitgestellten SOAP-Handler ist (und Sie dies tun, bevor die Steuerung an das Zielanwendungsprogramm übergeben wird), wird die Zielanwendung in einer neuen Task mit der neuen Transaktions-ID ausgeführt.

Neue Tasks können nicht gestartet werden, wenn sowohl der Terminal-Handler als auch der Anwendungshandler einer Pipeline auf demselben JVM-Server ausgeführt

werden. Wenn Sie JAX-WS-Axis2-Anwendungen in CICS implementieren, kann aus diesem Grund DFHWS-TRANID nicht zum Ändern der Benutzer-ID verwendet werden.

Container DFHWS-URI:

DFHWS-URI ist ein Container vom Typ DATATYPE(CHAR), der den URI des Service enthält.

In einer Service-Provider-Pipeline extrahiert CICS den relativen URI aus der eingehenden Nachricht und platziert ihn im Container DFHWS-URI.

Beispiel: Wenn der URI der Web-Services `http://example.com/location/address` oder `jms://queue?destination=INPUT.QUEUE&targetService=/location/address` lautet, ist der relative URI `/location/address`.

Wenn Sie Web-Service-Adressierung in Ihrer Requester-Pipeline verwenden, wird dieser Container in der folgenden Reihenfolge erstellt und aktualisiert:

1. Wenn der Befehl **INVOKE SERVICE** ausgeführt wird, erstellt er den Container DFHWS-URI und initialisiert ihn mit dem Wert der WSDL-Serviceendpunktadresse. Wenn der API-Befehl **WSACONTEXT BUILD** verwendet wurde, um einen Adressierungskontext zu erstellen, dürfen Sie die Parameter **URI** oder **URI-MAP** im Befehl **INVOKE SERVICE** nicht angeben.
2. Wenn der Web-Service-Adressierungshandler (DFHWSADH) ausgeführt wird, falls eine `<wsa:To>`-EPR im Adressierungskontext mit einem nicht anonymen URI vorhanden ist, wird der URI im Container DFHWS-URI durch den Wert der `<wsa:To>`-EPR überschrieben. Der anonyme URI wird ignoriert.

Die SOAP-Nachricht wird an den durch den URI definierten Service in DFHWS-URI gesendet.

In einer Service-Requester-Pipeline stellt CICS den URI, der im Befehl **INVOKE SERVICE** angegeben ist, oder, falls dieser fehlt, den URI aus der Web-Service-Bindung in den Container DFHWS-URI. Sie können diesen URI überschreiben, indem Sie einen Nachrichtenhandler in der Pipeline verwenden.

Ein Service kann einen HTTP-, HTTPS-, JMS- oder WebSphere MQ-URI für externe Services verwenden. Ein Service kann auch einen CICS-URI für einen Service verwenden, der von einer anderen CICS-Anwendung bereitgestellt wird:

URI	Abfragezeichenfolge	Beschreibung
<code>cics://PROGRAM/programm</code>	<code>?optionen</code>	Der CICS-Transporthandler verwendet den Befehl EXEC CICS LINK PROGRAM , um das angegebene Programm zu verknüpfen, wobei der aktuelle Kanal und die aktuellen Container übergeben werden. Die Anwendungsdaten werden nicht umgesetzt.

URI	Abfragezeichenfolge	Beschreibung
cics://SERVICE/service	?targetServiceUri=zielservice-uri &optionen	Der CICS-Transporthandler verwendet den Pfad des Service, ausgedrückt als <i>zielservice-uri</i> , um eine URIMAP-Ressource zuzuordnen und die Anforderung eine Provider-Pipeline durchlaufen zu lassen. Sie müssen einen Wert für den Parameter targetServiceUri angeben, wenn Sie diesen URI-Typ verwenden.
cics://PIPELINE/pipeline	?targetServiceUri=zielservice-uri	Der CICS-Transporthandler startet eine andere Service-Requester-Pipeline.

Sie können Parameter zu jedem Typ von CICS-URI mithilfe des Formats *parameter=wert* hinzufügen, wobei die einzelnen Parameter durch Et-Zeichen getrennt ist. Die folgenden Regeln gelten für den CICS-URI:

- Dem ersten Parameter in der Abfragezeichenfolge muss ein Fragezeichen vorangestellt werden. Vor diesem Punkt im URI können Sie kein Fragezeichen verwenden.
- Um ein Et-Zeichen in einen Parameterwert einzuschließen, müssen Sie ihm ein Escapezeichen voranstellen. Weitere Informationen finden Sie im Beispielabschnitt am Ende dieses Themas.
- CICS ändert alle Kleinbuchstaben für *programm* und *pipeline* in Großbuchstaben.

Die Parameter in der Zeichenfolge bestimmen, wie CICS die Anforderung am Ende der Requester-Pipeline verarbeitet.

maxCommareaLength=wert

Geben Sie die maximale Länge des Kommunikationsbereichs, die für das Zielanwendungsprogramm erforderlich ist, in Byte an. Der Wert darf nicht größer als 32.763 sein. Wenn dieser Parameter in der Abfragezeichenfolge vorhanden ist, stellt CICS eine Verknüpfung zu dem angegebenen Programm mithilfe eines Kommunikationsbereichs her. Wenn dieser Parameter nicht in der Abfragezeichenfolge vorhanden ist, stellt CICS eine Verknüpfung zu dem angegebenen Programm mithilfe eines Kanals her.

Für diesen Parameter muss die Groß-/Kleinschreibung nicht beachtet werden. Er ist für den URI 'cics://PROGRAM' gültig.

newTask=yes|no

Geben Sie an, ob der Transporthandler die Anforderung als neue Task ausführt.

Für diesen Parameter muss die Groß-/Kleinschreibung nicht beachtet werden. *cics://PROGRAM/testapp?newTask=yes* und *cics://PROGRAM/testapp?NEWTASK=Yes* sind identisch.

targetServiceUri=uri

Geben Sie den Pfad des aufzurufenden Service an. Bei einem Zieltyp SERVICE verwendet der Transporthandler den Wert mit *host=localhost*, um die URIMAP-Ressource zu lokalisieren und eine Service-Provider-Pipeline zu starten. Bei einem Zieltyp PIPELINE verwendet der Transporthandler den Wert, um eine andere Requester-Pipeline zu starten.

Für diesen Parameter muss die Groß-/Kleinschreibung beachtet werden.

transid=char(4)

Geben Sie eine Transaktion an, unter der die Anforderung ausgeführt wird.
Der Transporthandler startet unter Verwendung der angegebenen Transaktions-ID einen Anforderungsdatenstrom.

Für diesen Parameter muss die Groß-/Kleinschreibung beachtet werden.

userid=char(8)

Geben Sie eine Benutzer-ID an, unter der die Anforderung ausgeführt wird.
Der Transporthandler startet unter Verwendung der angegebenen Benutzer-ID einen Anforderungsdatenstrom.

Für diesen Parameter muss die Groß-/Kleinschreibung nicht beachtet werden.

Zieltyp	Parameter im URI	
PROGRAM	userid	Optional
PROGRAM	transid	Optional
PROGRAM	maxCommareaLength	Optional
PROGRAM	newTask	Optional. Muss 'yes' oder gar nicht angegeben sein, wenn Sie userid oder transid angeben.
PROGRAM	targetServiceUri	Nicht unterstützt
SERVICE	userid	Optional
SERVICE	transid	Optional
SERVICE	maxCommareaLength	Nicht unterstützt
SERVICE	newTask	Optional. Muss 'yes' oder gar nicht angegeben sein, wenn Sie userid oder transid angeben.
SERVICE	targetServiceUri	Erforderlich
PIPELINE	userid	Nicht unterstützt
PIPELINE	transid	Nicht unterstützt
PIPELINE	maxCommareaLength	Nicht unterstützt
PIPELINE	newTask	Nicht unterstützt
PIPELINE	targetServiceUri	Erforderlich

Beispiele für CICS-URIs

In diesem ersten Beispiel hat der Container DFHWS-URI den folgenden URI, wenn er das Ende der Pipeline erreicht:

```
cics://PROGRAM/testapp?newTask=yes&userid=user1
```

Der Transporthandler stellt eine Verknüpfung zum CICS-Programm namens 'testapp' her und übergibt den Kanal und die Container. Es findet keine Datenumsetzung statt, deshalb muss das Zielprogramm in der Lage sein, die Inhalte der Container im aktuellen Kanal zu verarbeiten. CICS stellt eine Verknüpfung zu dem Programm unter einer neuen Arbeitseinheit und einer anderen Benutzer-ID 'user1' her.

In diesem zweiten Beispiel hat der Container DFHWS-URI den folgenden URI, wenn er das Ende der Pipeline erreicht:

```
cics://SERVICE/getStockQuote?targetServiceUri=/stock/getQuote&newTask=yes&userid=user2
```

Der Transporthandler ersetzt den URI im Container DFHWS-URI durch den Wert `/stock/getQuote`, findet die URIMAP anhand des Pfads im Parameter **targetServiceUri**, um den URI aufzulösen, und startet die Provider-Pipeline unter einer neuen Task und einer anderen Benutzer-ID.

In diesem dritten Beispiel hat der Container DFHWS-URI den folgenden URI, wenn er das Ende der Pipeline erreicht:

```
cics://PIPELINE/reqpipeA?targetServiceUri=cics://PROGRAM/testapp?newTask=yes%26userid=user1
```

Der Transporthandler ersetzt den URI im Container DFHWS-URI durch den Wert `cics://PROGRAM/testapp?newTask=yes&userid=user1` und startet die Anforderungspipeline namens 'reqpipeA', wobei der aktuelle Kanal und die aktuellen Container übergeben werden. Die `%26`-Zeichen sind Escapezeichen für das Et-Zeichen, damit der Transporthandler den gesamten URI in den Container DFHWS-URI stellt.

Container DFHWS-URI-RESID:

Dies ist ein Container vom Typ DATATYPE(CHAR), der nur für Anwendungen verfügbar ist, die von einer JSON-Pipeline angehängt wurden.

Dieser Container enthält eine vereinfachte Kopie des URI-Pfads (eine Ressourcen-ID oder ResID), aus dem das Pfad-URI-Fragment, das für den URIMAP-Abgleich verwendet wurde, entfernt wurde. Beispiel:

Wenn die URIMAP, die mit der eingehenden Anforderung übereinstimmt, folgenden Pfad hat:

```
/JSONServices/CustomerDetails/*
```

Und wenn der eingehende URI vom Client folgendermaßen lautete:

```
http://www.example.org:10000/JSONServices/CustomerDetails/customerNumber/13388?action=query
```

Dann wäre der Inhalt von DFHWS-URI-RESID:

```
customerNumber/13388
```

REST-konforme JSON-Anwendungen können diesen Container verwenden, um die Ressourcen-ID (oder den Primärschlüssel) für REST-konforme Ressourcen zu ermitteln, die mithilfe einer URIMAP mit Platzhaltern abgeglichen werden. Dies sollte wesentlich einfacher sein als ein Parsing der Inhalte von DFHWS-URI.

Anmerkung: Wenn das PATH-Attribut der übereinstimmenden URIMAP keine Platzhalter enthält (d. h. es enthält den vollständigen Pfad für den URI), sind die Inhalte dieses Containers leer.

Anmerkung: Das PATH-Attribut der übereinstimmenden URIMAP kann ein optionales Abfragezeichenfolgefragment enthalten. Ist dies der Fall, wird das Abfragezeichenfolgefragment beim Erstellen dieses Containers ignoriert.

Container DFHWS-URI-QUERY:

Dies ist ein Container vom Typ DATATYPE(CHAR), der für Anwendungsprogramme in allen CICS-Pipelines im HTTP-Providermodus verfügbar ist.

Dieser Container enthält das Abfragezeichenfolgefragment des URI. Beispiel:

Wenn der eingehende URI vom Client folgendermaßen lautete:

```
http://www.example.org:10000/JSONServices/CustomerDetails/customerNumber/13388?action=query&page=1
```


Dann wären die Inhalte von DFHWS-URI-QUERY:

action=query&page=1

Anwendungen können die Inhalte dieses Containers parsen, um einzelne **name=value**-Parameter aus dem URI zu identifizieren.

Anmerkung: Wenn der eingehende URI keine Abfragezeichenfolge enthält, wird dieser Container nicht im Kanal vorhanden sein.

Container DFHWS-URIMAPPATH:

Dies ist ein Container vom Typ DATATYPE(CHAR), der eine Kopie der PATH-Daten von der URIMAP enthält, die für den Abgleich der eingehenden URI verwendet wurde.

Alle Anwendungen, die an eine Pipeline angehängt sind, können anhand dieses Containers nachvollziehen, wie sie angehängt wurden.

Container DFHWS-USERID:

DFHWS-USERID ist ein Container vom Typ DATATYPE(CHAR), der mit der Benutzer-ID der Task initialisiert wird, in der die Pipeline ausgeführt wird.

Wenn Sie den Inhalt dieses Containers in einer Service-Provider-Pipeline ändern, in der der Terminal-Handler einer der von CICS bereitgestellten SOAP-Handler ist (und Sie dies tun, bevor die Steuerung an das Zielanwendungsprogramm übergeben wird), wird die Zielanwendung in einer neuen Task ausgeführt, die der neuen Benutzer-ID zugeordnet wird. Wenn Sie den Inhalt des Containers DFHWS-TRANID nicht ändern, hat die neue Task dieselbe Transaktions-ID wie die Task, in der die Pipeline ausgeführt wird.

Neue Tasks können nicht gestartet werden, wenn sowohl der Terminal-Handler als auch der Anwendungshandler einer Pipeline auf demselben JVM-Server ausgeführt werden. Wenn Sie JAX-WS-Axis2-Anwendungen in CICS implementieren, kann aus diesem Grund DFHWS-USERID nicht zum Ändern der Benutzer-ID verwendet werden.

Container DFHWS-WEBSERVICE:

DFHWS-WEBSERVICE ist ein Container vom Typ DATATYPE(CHAR), der nur in einer Service-Provider-Pipeline verwendet wird. Er enthält den Namen des Web-Service, der die Ausführungsumgebung angibt, wenn die Zielanwendung unter Verwendung des Web-Service-Assistenten implementiert wurde.

CICS stellt diesen Container nicht in einer Service-Requester-Pipeline bereit.

Container DFHWS-CID-DOMAIN:

DFHWS-CID-DOMAIN ist ein Container vom Typ DATATYPE(CHAR). Er enthält den Domänennamen, der zum Generieren von 'content-ID'-Werten für die Referenzierung binärer Anhänge verwendet wird.

Der Domänenname hat standardmäßig den Wert `cicsts`. Sie können den Wert überschreiben, indem Sie das Element `<mime_options>` in der Pipelinekonfigurationsdatei angeben.

Sie können die Inhalte dieses Containers nicht ändern.

Container DFHWS-MTOM-IN:

DFHWS-MTOM-IN ist ein Container vom Typ DATATYPE(BIT), der Informationen zu den angegebenen Optionen für das Element `<cics_mtom_handler>` der Pipelinekonfigurationsdatei sowie Informationen zu dem Nachrichtenformat enthält, das in der Pipeline empfangen wurde.

Er enthält die Informationen zur Verarbeitung einer eingehenden MTOM-Nachricht in der Pipeline. Die eingehende Nachricht kann eine Anforderungsnachricht von einem Web-Service-Requester oder eine Antwortnachricht von einem Web-Service-Provider sein.

Wenn Sie kein Element `<cics_mtom_handler>` in der Pipelinekonfigurationsdatei angeben oder wenn eine SOAP-Nachricht anstelle einer MTOM-Nachricht empfangen wird, wird dieser Container nicht erstellt.

Wenn Web Services Security in der Pipeline konfiguriert ist oder wenn die Validierung für einen Web-Service aktiviert ist, können die Inhalte des Felds XOP_MODE in DFHWS-MTOM-IN von CICS bei der Erstellung des Containers überschrieben werden. Wenn Sie beispielsweise die Pipeline so konfigurieren, dass der Inhalt von MTOM-Nachrichten im Direktmodus verarbeitet wird, und wenn Sie dann die Validierung für den Web-Service aktivieren, überschreibt CICS den definierten Wert in der Pipelinekonfigurationsdatei und legt fest, dass die XOP-Verarbeitung im Kompatibilitätsmodus ausgeführt werden soll. CICS führt die Überschreibung aufgrund der Einschränkungen bei der Unterstützung für die Verarbeitung von XOP-Dokumenten und binären Anhängen in der Pipeline aus.

Sie können die Inhalte dieses Containers nicht ändern.

Tabelle 9. Struktur des Containers DFHWS-MTOM-IN

Feldname	Länge (Byte)	Inhalt
MTOM_STATUS	4	Enthält den Wert 1, was angibt, dass die von CICS empfangene Nachricht im MTOM-Format vorliegt.
MTOMNOXOP_STATUS	4	Enthält einen der folgenden Werte: 0 Die MTOM-Nachricht enthält binäre Anhänge. 1 Die MTOM-Nachricht enthält keine binären Anhänge.
XOP_MODE	4	Enthält einen der folgenden Werte: 0 Keine XOP-Verarbeitung. 1 XOP-Verarbeitung im Kompatibilitätsmodus. 2 XOP-Verarbeitung im Direktmodus.

Container DFHWS-MTOM-OUT:

DFHWS-MTOM-OUT ist ein Container vom Typ DATATYPE(BIT), der Informationen zu den angegebenen Optionen für das Element `<cics_mtom_handler>` der Pipelinekonfigurationsdatei enthält.

Er enthält die Informationen zur Verarbeitung einer ausgehenden MTOM-Nachricht in der Pipeline, ob es sich um eine Antwortnachricht für einen Web-Service-Requester oder eine Anforderungsnachricht für einen Web-Service-Provider handelt.

Wenn Sie das Element `<cics_mtom_handler>` in der Pipelinekonfigurationsdatei nicht angeben oder wenn das Element `<mtom_options>` in der Pipelinekonfigurationsdatei das Attribut `send_mtom="no"` hat, wird dieser Container nicht erstellt.

Im Providermodus wird dieser Container zur gleichen Zeit erstellt wie der Container DFHWS-MTOM-IN. Wenn das Element `<mtom_options>` in der Pipelinekonfigurationsdatei das Attribut `send_mtom="same"` hat, wird das Feld MTOM_STATUS festgelegt, um anzugeben, ob der Web-Service-Requester eine MTOM- oder SOAP-Antwortnachricht erwartet.

Wenn Web Service Security in der Pipeline konfiguriert ist oder wenn die Validierung für einen Web-Service aktiviert ist, kann das Feld XOP_MODE von DFHWS-MTOM-OUT in CICS bei der Erstellung des Containers geändert werden. Wenn Sie beispielsweise die Pipeline konfigurieren, um das XOP-Dokument und alle binären Anhänge mithilfe des Direktmodus zu verarbeiten, und wenn Sie dann die Validierung für einen Web-Service aktivieren, überschreibt CICS den definierten Wert in der Pipelinekonfigurationsdatei und legt fest, dass die XOP-Verarbeitung im Kompatibilitätsmodus ausgeführt wird, wenn der Container erstellt wird. CICS führt die Überschreibung aufgrund von Einschränkungen bei der Unterstützung für die Verarbeitung von XOP-Dokumenten und binären Anhängen in der Pipeline aus.

Sie können die Inhalte dieses Containers nicht ändern.

Tabelle 10. Struktur des Containers DFHWS-MTOM-OUT

Feldname	Länge (Byte)	Inhalt
MTOM_STATUS	4	Gibt an, ob MTOM aktiviert ist: <div> <div>0</div> <div>MTOM ist nicht aktiviert. Die ausgehende Nachricht wird im SOAP-Format gesendet.</div> </div> <div> <div>1</div> <div>MTOM ist aktiviert. Die ausgehende Nachricht wird im MTOM-Format gesendet.</div> </div>
MTOMNOXOP_STATUS	4	Gibt an, ob MTOM verwendet werden soll, wenn keine binären Anhänge vorhanden sind: <div> <div>0</div> <div>Keine MTOM-Nachricht senden, wenn keine binären Anhänge vorhanden sind.</div> </div> <div> <div>1</div> <div>MTOM-Nachricht senden, wenn keine binären Anhänge vorhanden sind.</div> </div>
XOP_MODE	4	Gibt an, welche XOP-Verarbeitung stattfinden soll: <div> <div>0</div> <div>Keine XOP-Verarbeitung.</div> </div> <div> <div>1</div> <div>XOP-Verarbeitung im Kompatibilitätsmodus.</div> </div> <div> <div>2</div> <div>XOP-Verarbeitung im Direktmodus.</div> </div>

Container DFHWS-WSDL-CTX:

DFHWS-WSDL-CTX ist ein Container vom Typ DATATYPE(CHAR), der entweder in einer Service-Provider- oder in einer Service-Requester-Anwendung verwendet

wird, die mit dem CICS-Web-Service-Assistenten implementiert wird. Er enthält WSDL-Kontextinformationen, die für Überwachungszwecke verwendet werden können.

DFHWS-WSDL-CTX enthält die folgenden Kontextinformationen für das WSDL-Dokument.

- Der Name und Namensbereich der Operation, für die die Anwendung aufgerufen wird.
- Falls bekannt, der Name und Namensbereich für den WSDL 1.1-Port oder den WSDL 2.0-Endpunkt, der verwendet wird.

Diese Werte sind durch Leerzeichen voneinander getrennt. DFHWS-WSDL-CTX wird von CICS nur auf Laufzeitebene 2.1 und höher gefüllt.

Wenn Sie den CICS-Web-Service-Assistenten verwendet haben, um Ihre Anwendung zu implementieren, wird dieser Container von CICS gefüllt:

- In einer Service-Provider-Pipeline wird dieser Container von dem DFHPITP-Anwendungshandler gefüllt, wenn er die eingehende Nachricht vom Terminal-Handler empfängt.
- In einer Service-Requester-Pipeline wird dieser Container gefüllt, wenn die Anwendung den Befehl **INVOKE SERVICE** ausführt.

Wenn die Anwendung das Programm DFHPIRT zum Starten der Pipeline verwendet, füllt die Anwendung den Container DFHWS-WSDL-CTX.

Container DFHWS-XOP-IN:

DFHWS-XOP-IN ist ein Container vom Typ DATATYPE(BIT). Er enthält eine Liste mit Referenzen auf die binären Anhänge, die aus einer eingehenden MIME-Nachricht entpackt und mithilfe der XOP-Verarbeitung in Containern platziert wurden.

Jeder Anhangsdatensatz im Container DFHWS-XOP-IN besteht aus den folgenden Elementen:

- Dem 16-Byte-Namen des Containers, der die MIME-Header für den binären Anhang enthält.
- Dem 16-Byte-Namen des Containers, der den binären Anhang enthält.
- Dem 'content-ID'-Element mit einer Länge von 2 Bytes im signierten Halbwort-Binärformat.
- Dem 'content-ID'-Element, einschließlich der Begrenzungszeichen < und >, gespeichert als ASCII-Zeichenfolge.

Sie können die Inhalte dieses Containers nicht ändern.

Container DFHWS-XOP-OUT:

DFHWS-XOP-OUT ist ein Container vom Typ DATATYPE(BIT). Er enthält eine Liste von Referenzen auf die Container mit binären Anhängen. Die binären Anhänge werden von dem MTOM-Handlerprogramm in eine ausgehende MIME-Nachricht gepackt.

Jeder Anhangsdatensatz im Container DFHWS-XOP-OUT besteht aus den folgenden Elementen:

- Dem 16-Byte-Namen des Containers, der die MIME-Header für den binären Anhang enthält.

- Dem 16-Byte-Namen des Containers, der den binären Anhang enthält.
- Dem 'content-ID'-Element mit einer Länge von 2 Bytes im signierten Halbwort-Binärformat.
- Dem 'content-ID'-Element, einschließlich der Begrenzungszeichen < und >, gespeichert als ASCII-Zeichenfolge.

Sie können die Inhalte dieses Containers nicht ändern.

Container des Programms zur Headerverarbeitung

Die von CICS bereitgestellten SOAP 1.1- und SOAP 1.2-Nachrichtenhandler stellen eine Verknüpfung mit den Programmen zur Headerverarbeitung über den Kanal DFHHHC-V1 her. Zu den Containern, die an den Kanal übergeben werden, zählen diverse, die spezifisch für die Schnittstelle des Programms zur Headerverarbeitung sind, sowie Gruppen von *Kontextcontainern* und *Benutzercontainern*, die für alle Programme zur Headerverarbeitung und alle Nachrichtenhandlerprogramme in der Pipeline zugänglich sind.

Der Container DFHHEADER ist spezifisch für die Schnittstelle des Programms zur Headerverarbeitung. Andere Container sind an anderen Stellen in Ihrer Pipeline verfügbar, haben aber einen bestimmten Zweck in einem Programm zur Headerverarbeitung. Die Container in dieser Kategorie sind DFHWS-XMLNS, DFHWS-BODY und DFHXMLSS-PARSE.

Anmerkung: Obwohl ein Web-Service, der Axis2 verwendet, um SOAP-Nachrichten zu verarbeiten, die Schnittstelle des Programms zur Headerverarbeitung nutzen kann, ist es effizienter, Ihre eigenen Axis2-Handler in Java zu schreiben, um die SOAP-Header zu verarbeiten. Weitere Informationen zum Erstellen von Axis2-Handlern finden Sie unter Writing Your Own Axis2 Module.

Container DFHHEADER

Wenn das Programm zur Headerverarbeitung aufgerufen wird, enthält DFHHEADER den einzelnen Headerblock, der die Ausführung des Programms zur Headerverarbeitung startet. Wenn das Headerprogramm mit `<mandatory>true</mandatory>` oder `<mandatory>1</mandatory>` in der Pipelinekonfigurationsdatei angegeben ist, wird es auch dann aufgerufen, wenn es keinen passenden Headerblock in der SOAP-Nachricht gibt. In diesem Fall hat der Container DFHHEADER eine Länge von null. Dies passiert, wenn ein Programm zur Headerverarbeitung aufgerufen wird, um einen Headerblock zu einer SOAP-Nachricht hinzuzufügen, die keine Headerblöcke hat.

Die SOAP-Nachricht, die CICS erstellt, hat ursprünglich keine Header. Wenn Sie Ihrer Nachricht Header hinzufügen möchten, müssen Sie sicherstellen, dass mindestens ein Programm zur Headerverarbeitung aufgerufen wird, indem Sie `<mandatory>true</mandatory>` oder `<mandatory>1</mandatory>` angeben.

Wenn das Headerprogramm zurückkehrt, muss der Container DFHHEADER keinen, einen oder mehrere Headerblöcke enthalten, die CICS anstelle des Originals in die SOAP-Nachricht einfügt:

- Sie können den ursprünglichen Headerblock unverändert zurückgeben.
- Sie können den Inhalt des Headerblocks ändern.
- Sie können einen oder mehrere neue Headerblöcke an den ursprünglichen Block anhängen.
- Sie können den ursprünglichen Headerblock durch einen oder mehrere andere Blöcke ersetzen.

- Sie können den Headerblock vollständig löschen.

Container DFHWS-XMLNS

Wenn das Programm zur Headerverarbeitung aufgerufen wird, enthält DFHWS-XMLNS Informationen zu XML-Namensbereichen, die im SOAP-Envelope deklariert sind. Das Headerprogramm kann diese Informationen verwenden, um die folgenden Tasks auszuführen:

- Auflösen qualifizierter Namen, die im Headerblock gefunden werden
- Erstellen qualifizierter Namen in neuen oder geänderten Headerblöcken

Die Namensbereichsinformationen bestehen aus einer Liste von Namensbereichsdeklarationen, die die XML-Standardnotation für das Deklarieren von Namensbereichen verwenden. Die Namensbereichsdeklarationen in DFHWS-XMLNS sind durch Leerzeichen getrennt. Beispiel:

```
xmlns:na='http://abc.example.org/schema' xmlns:nx='http://xyz.example.org/schema'
```

Sie können weitere Namensbereichsdeklarationen zum SOAP-Envelope hinzufügen, indem Sie sie an die Inhalte von DFHWS-XMLNS anhängen. Namensbereiche, deren Geltungsbereich ein SOAP-Headerblock oder ein SOAP-Hauptteil sind, werden jedoch am besten in dem Headerblock bzw. im Hauptteil selbst deklariert. Es wird empfohlen, keine Namensbereichsdeklarationen aus dem Container DFHWS-XMLNS in einem Programm zur Headerverarbeitung zu löschen, da XML-Elemente, die im Programm nicht sichtbar sind, möglicherweise von ihnen abhängen.

Container DFHWS-BODY

Dieser Container enthält den Hauptabschnitt des SOAP-Envelopes. Das Programm zur Headerverarbeitung kann diesen Inhalt ändern.

Wenn das Programm zur Headerverarbeitung aufgerufen wird, enthält DFHWS-BODY das SOAP-Element <Body>.

Wenn das Headerprogramm zurückgegeben wird, muss der Container DFHWS-BODY wieder einen gültigen SOAP-Hauptteil (<Body>) enthalten, den CICS anstelle des Originals in die SOAP-Nachricht einfügt:

- Sie können den ursprünglichen Hauptteil unverändert zurückgeben.
- Sie können den Inhalt des Hauptteils ändern.

Sie dürfen den SOAP-Hauptteil nicht vollständig löschen, da jede SOAP-Nachricht ein Element <Body> enthalten muss.

Container DFHXMLSS-PARSE

Wenn Sie das Element <cics_soap_1.1_handler> oder <cics_soap_1.2_handler> in Ihrer Pipelinekonfiguration verwenden und das Headerprogramm aufgerufen wird, enthält DFHXMLSS-PARSE die XMLSS-Datensätze (XML System Services) für diesen Header. Dieser Container wird nicht erstellt, wenn die Elemente <cics_soap_1.1_handler_java> oder <cics_soap_1.2_handler_java> verwendet werden.

Steuer-, Kontext- und Benutzercontainer

Neben den beschriebenen Containern übergibt die Schnittstelle die *Steuercontainer*, *Kontextcontainer* und *Benutzercontainer* im Kanal DFHHHC-V1.

Weitere Informationen zu diesen Containern finden Sie unter „In der Pipeline verwendete Container“ auf Seite 165.

Sicherheitscontainer

Sicherheitscontainer werden im Kanal DFHWSTC-V1 verwendet, um Identitätstoken an einen Sicherheitstokenservice (STS) wie Tivoli Federated Identity Manager zu senden und von diesem zu empfangen. Diese Schnittstelle wird als *vertrauenswürdige Clientschnittstelle* bezeichnet und kann in Web-Service-Requester- und Providerpipelines verwendet werden.

Container DFHWS-IDTOKEN:

DFHWS-IDTOKEN ist ein Container vom Typ DATATYPE(CHAR). Er enthält das Token, das der Sicherheitstokenservice (STS) entweder validiert oder zum Ausgeben eines Identitätstokens für diese Nachricht verwendet.

Das Token muss das XML-Format haben.

Verwenden Sie diesen Container nur mit dem Kanal DFHWSTC-V1 für die vertrauenswürdige Clientschnittstelle.

Container DFHWS-RESTOKEN:

DFHWS-RESTOKEN ist ein Container vom Typ DATATYPE(CHAR). Er enthält die Antwort vom Sicherheitstokenservice.

Die Antwort hängt von der Aktion ab, die von STS im Container DFHWS-STSACTION angefordert wurde.

- Wenn es sich um eine Ausgeben-Aktion handelt, enthält dieser Container das Token, das STS gegen das Token ausgetauscht hat, das im Container DFHWS-IDTOKEN gesendet wurde.
- Wenn es sich um eine Validieren-Aktion handelt, enthält dieser Container einen URI, um anzugeben, ob das Sicherheitstoken, das im Container DFHWS-IDTOKEN gesendet wurde, gültig oder ungültig ist. Folgende URIs können zurückgegeben werden:

URI	Beschreibung
http://schemas.xmlsoap.org/ws/2005/02/trust/status/valid	Das Sicherheitstoken ist gültig.
http://schemas.xmlsoap.org/ws/2005/02/trust/status/invalid	Das Sicherheitstoken ist nicht gültig.

Dieser Container wird im Kanal DFHWSTC-V1 zurückgegeben, wenn die vertrauenswürdige Clientschnittstelle verwendet wird.

Container DFHWS-SERVICEURI:

DFHWS-SERVICEURI ist ein Container vom Typ DATATYPE(CHAR). Er enthält den URI, den der Sicherheitstokenservice (STS) als 'AppliesTo'-Bereich verwendet.

Der 'AppliesTo'-Bereich wird verwendet, um zu bestimmen, welchem Web-Service das Sicherheitstoken zugeordnet ist.

Verwenden Sie diesen Container nur mit dem Kanal DFHWSTC-V1 für die vertrauenswürdige Clientschnittstelle.

Container DFHWS-STSACTION:

DFHWS-STSACTION ist ein Container vom Typ DATATYPE(CHAR). Er enthält den URI der Aktion, die der Sicherheitstokenservice (STS) für die Validierung oder die Ausgabe eines Sicherheitstoken ausführt.

Die URI-Werte, die Sie in diesem Container angeben können, lauten wie folgt:

URI	Beschreibung
<code>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</code>	Der STS gibt im Austausch gegen das im Container DFHWS-IDTOKEN gesendete Token ein neues Token aus.
<code>http://schemas.xmlsoap.org/ws/2005/02/trust/Validate</code>	Der STS validiert das im Container DFHWS-IDTOKEN gesendete Token.

Verwenden Sie diesen Container nur mit dem Kanal DFHWSTC-V1 für die vertrauenswürdige Clientschnittstelle.

Container DFHWS-STSFault:

DFHWS-STSFault ist ein Container vom Typ DATATYPE(CHAR). Er enthält den Fehler, der vom Sicherheitstokenservice (STS) zurückgegeben wurde.

Tritt ein Fehler auf, gibt der STS einen SOAP-Fehler aus. Der Inhalt des SOAP-Fehlers wird in diesem Container zurückgegeben.

Dieser Container wird im Kanal DFHWSTC-V1 zurückgegeben, wenn die vertrauenswürdige Clientschnittstelle verwendet wird.

Container DFHWS-STReason:

DFHWS-STReason ist ein Container vom Typ DATATYPE(CHAR). Er enthält den Inhalt des Elements `<wst:Reason>`, wenn ein solches Element in der Antwortnachricht des Sicherheitstokenservices (STS) enthalten ist.

Das Element `<wst:Reason>` enthält eine optionale Zeichenfolge, die Informationen zum Status der Validierungsantwort enthält, die von CICS an den STS gesendet wurde. Wenn das Sicherheitstoken nicht gültig ist, können die von STS in diesem Element bereitgestellten Informationen dabei helfen, zu bestimmen, warum das Token nicht gültig ist.

Weitere Informationen finden Sie in der *Web Services Trust Language*-Spezifikation unter OASIS WS-Trust v1.4 Standard.

Container DFHWS-STSURI:

DFHWS-STSURI ist ein Container vom Typ DATATYPE(CHAR). Er enthält den absoluten URI des Sicherheitstokenservice (STS), der verwendet wird, um ein Identitätstoken für die SOAP-Nachricht zu validieren oder auszugeben.

Das Format des URI ist `http://www.example.com:8080/TrustServer/SecurityTokenService`. Abhängig von Ihren Sicherheitsanforderungen können Sie HTTP oder HTTPS verwenden.

Verwenden Sie diesen Container nur mit dem Kanal DFHWSTC-V1 für die vertrauenswürdige Clientschnittstelle.

Container DFHWS-TOKENTYPE:

DFHWS-TOKENTYPE ist ein Container vom Typ DATATYPE(CHAR). Er enthält den URI des angeforderten Tokentyps, den der Sicherheitstokenservice (STS) als Identitätstoken für die SOAP-Nachricht ausgibt.

Sie können jeden gültigen Tokentyp angeben, aber er muss vom STS unterstützt werden.

Verwenden Sie diesen Container nur mit dem Kanal DFHWSTC-V1 für die vertrauenswürdige Clientschnittstelle.

Container für SAML-Unterstützung

Die schreibgeschützten Container, die von der CICS-SAML-Unterstützung verwendet werden.

In den folgenden Themen bedeutet *nnn*, dass mehr als einen Container vorhanden sein kann. Container sind mit 001 bis *nnn* beziffert (die Anzahl von Containern, die von diesem Typ zurückgegeben werden). Es werden nicht mehr als 999 Container eines bestimmten Typs unterstützt und die Daten in der zugehörigen SAML-Zusicherung werden ignoriert. Container, die keinem DSECT zugeordnet sind, haben eine variable Länge.

Container DFHSAML-AnnnVmmm:

DFHSAML-AnnnVmmm ist ein Container vom Typ DATATYPE(CHAR). Er enthält den Attributwert *mmm* für das Attribut *nnn*, wobei *nnn* und *mmm* dreistellige Zahlen sind.

SAMLC-ATTRNUM gibt die Anzahl der Attribute im Container DFHSAML-COUNTS an.

DFHSAML-ATTRAnnn gibt die Anzahl von Werten für dieses Attribut an.

Container DFHSAML-ASSQNAME:

DFHSAML-ASSQNAME ist ein Container vom Typ DATATYPE(CHAR). Er enthält den Namensbereich der SAML-Zusicherung.

Mögliche Werte sind:

SAML 1.1

urn:oasis:names:tc:SAML:1.0:assertion

SAML 2.0

urn:oasis:names:tc:SAML:2.0:assertion

Diese Zusicherung muss ein URI sein. Wenn die Zusicherung komplexer ist, wird sie in die drei Teile extrahiert.

Container DFHSAML-ATTRAnnn:

DFHSAML-ATTRAnnn ist ein Container vom Typ DATATYPE(BIT). Er enthält ein BIN(31)-Feld mit der Anzahl von Werten für das Attribut *nnn*. Die maximale Anzahl von Werten ist 999.

SAMLC-ATTRNUM gibt die Anzahl der Attribute im Container DFHSAML-COUNTS an.

Container DFHSAML-ATTRFnnn:

DFHSAML-ATTRFnnn ist ein Container vom Typ DATATYPE(CHAR). Er enthält das Attributnamensformat für das Attribut *nnn*, wobei *nnn* eine dreistellige Zahl ist.

SAMLC-ATTRNUM gibt die Anzahl der Attribute im Container DFHSAML-COUNTS an.

Container DFHSAML-ATTRNnnn:

DFHSAML-ATTRNnnn ist ein Container vom Typ DATATYPE(CHAR). Er enthält den Attributnamen für das Attribut *nnn*, wobei *nnn* eine dreistellige Zahl ist.

SAMLC-ATTRNUM gibt die Anzahl der Attribute im Container DFHSAML-COUNTS an.

Container DFHSAML-ATTRSnnn:

DFHSAML-ATTRSnnn ist ein Container vom Typ DATATYPE(CHAR). Er enthält den Attributnamensbereich für das Attribut *nnn*, wobei *nnn* eine dreistellige Zahl ist.

SAMLC-ATTRNUM gibt die Anzahl der Attribute im Container DFHSAML-COUNTS an.

Container DFHSAML-ATTRYnnn:

DFHSAML-ATTRYnnn ist ein Container vom Typ DATATYPE(CHAR). Er enthält den ausführlichen Namen für das Attribut *nnn*, wobei *nnn* eine dreistellige Zahl ist.

SAMLC-ATTRNUM gibt die Anzahl der Attribute im Container DFHSAML-COUNTS an.

Container DFHSAML-AUDNRnnn:

DFHSAML-AUDNRnnn ist ein Container vom Typ DATATYPE(CHAR). Er enthält den AudienceRestriction-Namen.

AUDNRNUM ist die Anzahl von zurückgegebenen Containern.

Container DFHSAML-AUTHMETH:

DFHSAML-AUTHMETH ist ein Container vom Typ DATATYPE(CHAR). Er enthält die Methode, die zum Authentifizieren des Tokeninhabers verwendet wird.

Diese Methoden sind Kennwort, Kerberos und LTPA.

Container DFHSAML-CERTIDN:

DFHSAML-CERTIDN ist ein Container vom Typ DATATYPE(CHAR). Er enthält den definierten Ausstellernamen des X.509-Zertifikats des SAML-Unterzeichners.

Container DFHSAML-CERTSDN:

DFHSAML-CERTSDN ist ein Container vom Typ DATATYPE(CHAR). Er enthält den definierten Subjektnamen des X.509-Zertifikats des SAML-Unterzeichners.

Container DFHSAML-CERTSNUM:

DFHSAML-CERTSNUM ist ein Container vom Typ DATATYPE(CHAR). Es handelt sich um ein achtstelliges Feld, das die Seriennummer des X.509-Zertifikats des SAML-Unterzeichners enthält.

Container DFHSAML-CONFMETH:

DFHSAML-CONFMETH ist ein Container vom Typ DATATYPE(CHAR). Er enthält die SubjectConfirmation-Methode, die in diesem SAML-Token verwendet wird.

Gültige Methoden sind "holder-of-key", "bearer" oder "sender-vouches". Die zurückgegebene Zeichenfolge basiert auf dem OASIS SAML-Tokenprofil 1.1 und 2.0.

Anmerkung: SAML-Token mit mehr als einer Bestätigungsmethode werden nicht unterstützt. Wenn es mehr als eine Bestätigungsmethode gibt, sind die Ergebnisse unvorhersehbar.

Container DFHSAML-COUNTS:

DFHSAML-COUNTS ist ein Container vom Typ DATATYPE(BIT). Er enthält die Anzahl der Containerausgaben mit variabler Länge.

Container DFHSAML-FLAGS:

DFHSAML-FLAGS ist ein Container vom Typ DATATYPE(CHAR). Er enthält eine Sammlung von Markierungsbytes.

Container DFHSAML-ISSUER:

DFHSAML-ISSUER ist ein Container vom Typ DATATYPE(CHAR). Er enthält den Namen des Ausstellers.

Container DFHSAML-NAMID:

DFHSAML-NAMID ist ein Container vom Typ DATATYPE(CHAR). Er enthält den Wert der Namensformateigenschaft.

Container DFHSAML-NAMIDF:

DFHSAML-NAMIDF ist ein Container vom Typ DATATYPE(CHAR). Er enthält eine URI-Referenz, die die Klassifizierung von zeichenfolgenbasierten ID-Informationen darstellt.

Container DFHSAML-NAMIDQ:

DFHSAML-NAMIDQ ist ein Container vom Typ DATATYPE(CHAR). Er enthält die Sicherheits- oder Verwaltungsdomäne, die den Namen qualifiziert.

Container DFHSAML-NAMIDSP:

DFHSAML-NAMIDSP ist ein Container vom Typ DATATYPE(CHAR). Er enthält die Namens-ID, die von einem Service-Provider oder einem Zusammenschluss von Providern der Entität eingerichtet wird.

Container DFHSAML-NAMIDSPQ:

DFHSAML-NAMIDSPQ ist ein Container vom Typ DATATYPE(CHAR). Er enthält den Namen eines Service-Providers oder eines Zusammenschlusses von Providern.

Container DFHSAML-OUTTOKEN:

DFHSAML-OUTTOKEN ist ein Container vom Typ DATATYPE(CHAR). Er enthält ein SAML-Token.

Wenn es sich um einen Eingabecontainer handelt, enthält er das zuvor validierte Token, das an einen anderen Service-Provider weitergeleitet oder erweitert und dann weitergeleitet wird.

Wenn es sich um einen Ausgabecontainer handelt, enthält er eine SAML-Tokenausgabe der DFHSAML-Verarbeitung. Wenn die Verarbeitung eine Validierung oder Extraktion ist, ist dieses Token das überprüfte, extrahierte oder geänderte und gekündigte Token.

Container DFHSAML-PROXYnnn:

DFHSAML-PROXYnnn ist ein Container vom Typ DATATYPE(CHAR). Er enthält den ProxyRestriction-Zielgruppennamen.

Container DFHSAML-RESPONSE:

DFHSAML-RESPONSE ist ein Container vom Typ DATATYPE(BIT). Er enthält einen Antwortcode, der intern verwendet wird.

Container DFHSAML-SAMLID:

DFHSAML-SAMLID ist ein Container vom Typ DATATYPE(CHAR). Er enthält eine Zeichenfolge, die die ID für SAML 2.0 oder die AssertionID für SAML 1.1 darstellt.

Container DFHSAML-SUBJADDR:

DFHSAML-SUBJADDR ist ein Container vom Typ DATATYPE(CHAR). Er enthält die IP-Adresse in SubjectLocality.

Restriction: Dieser Container wird für SAML 2.0 nicht zurückgegeben.

Container DFHSAML-SUBJDNS:

DFHSAML-SUBJDNS ist ein Container vom Typ DATATYPE(CHAR). Er enthält die DNS-Adresse in SubjectLocality.

Container DFHSAML-TIMES:

DFHSAML-TIMES ist ein Container vom Typ DATATYPE(CHAR). Er enthält eine Sammlung von Zeitwerten.

Von CICS generierte Container

CICS generiert Container zum Speichern von Daten wie Variablenarrays und langen Zeichenfolgen. Diese Container werden während der Pipelineverarbeitung erstellt und als Eingabe für das Anwendungsprogramm oder als Ausgabe des Anwendungsprogramms verwendet. Diesen Containern wird DFH als Präfix vorangestellt.

Die Namenskonvention für diese Container besteht darin, das CICS-Modul zu verwenden, das sie erstellt hat, kombiniert mit einem numerischen Suffix, um den Containernamen in der Anforderung eindeutig zu machen. Diese Containernamen treten bei der Pipelineverarbeitung auf:

DFHPIAXIS-*nnnnnnnn*

Container, die zum Speichern von Zeichenfolgen und Variablenarrays verwendet werden, die an die Anwendung in Axis2-Pipelines übergeben werden. Diese Container können auch Binärdaten enthalten.

DFHPICC-*nnnnnnnnnn*

Container, die zum Speichern von Zeichenfolgen und Variablenarrays verwendet werden, die an die Anwendung übergeben werden. Diese Container können auch Binärdaten enthalten.

DFHPHII-*nnnnnnnnnn*

Ausgehende Anhangscontainer, die erstellt werden, wenn die Pipeline mit dem MTOM-Nachrichtenhandler aktiviert und im Direktmodus ausgeführt wird. Diese Container werden erstellt, wenn Binärdaten in einem Feld statt in einem Container von dem Anwendungsprogramm bereitgestellt werden.

DFHPIMM-*nnnnnnnnnn*

Eingehende Anhangscontainer, die während der Verarbeitung von MIME-Nachrichten erstellt werden. Diese Container werden von CICS generiert, wenn der MTOM-Nachrichtenhandler in der Pipeline aktiviert ist. Wenn die Direktmodusverarbeitung aktiviert ist, können diese Container direkt an die Anwendung übergeben werden.

DFHPIXO-*nnnnnnnnnn*

Ausgehende Anhangscontainer, die erstellt werden, wenn die Pipeline mit dem MTOM-Nachrichtenhandler aktiviert und im Kompatibilitätsmodus ausgeführt wird.

Die nummerierten Containernamen beginnen für jede Web-Service-Anforderung bei 1, z. B. DFHPICC-00000001. Wenn ein Anwendungsprogramm jedoch den Befehl **INVOKE SERVICE** verwendet, um mehr als eine Web-Service-Anforderung in demselben Kanal zu initialisieren, sind die Container, die für eine Antwort an die Anwendung zurückgegeben wurden, möglicherweise noch vorhanden, wenn eine weitere Anforderung ausgegeben wird. In dieser Situation prüft CICS, ob der Container bereits vorhanden ist, und erhöht die Nummer des generierten Containers, um einen Namenskonflikt zu vermeiden.

Benutzercontainer

Diese Container enthalten Informationen, die ein Nachrichtenhandler an einen anderen übergeben muss. Benutzercontainer werden ausschließlich von Nachrichtenhandlern verwendet. Sie können Ihre eigenen Namen für diese Container verwenden, aber diese Namen dürfen nicht mit der Zeichenfolge DFH beginnen.

Laufzeitverarbeitung für Web-Services

Um eine Anforderung an einen Web-Service-Provider zu senden oder eine Anforderung von einem Web-Service-Requester zu empfangen, muss Ihre Anwendung (oder Ihr Wrapperprogramm) korrekt mit der Web-Service-Unterstützung in CICS interagieren. Sie können auch steuern, welche Verarbeitung in der Pipeline stattfindet, um zu bestimmen, wie eingehende und ausgehende Anforderungen verarbeitet werden.

Vorgehensweise in CICS zum Aufrufen eines Web-Service-Provider-Programms, das mit dem Web-Service-Assistenten implementiert wurde

Wenn eine Service-Provider-Anwendung, die mit dem CICS-Web-Service-Assistenten implementiert wurde, aufgerufen wird, stellt CICS eine Verknüpfung damit über einen Kommunikationsbereich oder einen Kanal her.

Sie geben an, welche Art von Schnittstelle verwendet wird, wenn Sie die JCL-Prozedur DFHWS2LS oder DFHLS2WS mit dem Parameter **PGMINT** ausführen. Wenn Sie einen Kanal angeben, können Sie den Container im Parameter **CONTID** benennen.

- Wenn das Programm über eine Kommunikationsbereichsschnittstelle aufgerufen wird, enthält der Kommunikationsbereich die übergeordnete Datenstruktur, die CICS aus der SOAP-Anforderung erstellt hat.
- Wenn das Programm über eine Kanalschnittstelle aufgerufen wird, wird die übergeordnete Datenstruktur an Ihr Programm in dem Container weitergegeben, der im Parameter **CONTID** von DFHWS2LS oder DFHLS2WS angegeben wurde. Wenn Sie den Parameter **CONTID** nicht angegeben haben, werden die Daten an den Container DFHWS-DATA weitergegeben. Die Kanalschnittstelle unterstützt Arrays mit variierender Elementanzahl, die als Serie von verbundenen Datenstrukturen in einer Serie von Containern dargestellt werden. Diese Container werden ebenfalls vorhanden sein.

Wenn Sie API-Befehle für die Arbeit mit den Containern codieren, müssen Sie die Option **CHANNEL** nicht angeben, weil alle Container dem aktuellen Kanal zugeordnet sind (d. h. dem Kanal, der an das Programm übergeben wurde). Wenn Sie den Namen des Kanals wissen müssen, verwenden Sie den Befehl **EXEC CICS ASSIGN CHANNEL**.

Wenn Ihr Programm die Anforderung verarbeitet hat, muss es denselben Mechanismus zum Zurückgeben der Antwort verwenden: Wenn die Anforderung in einem Kommunikationsbereich empfangen wurde, muss die Antwort im Kommunikationsbereich zurückgegeben werden. Wenn die Anforderung in einem Container empfangen wurde, muss die Antwort in demselben Container zurückgegeben werden.

Wenn beim Ausgeben der Antwortnachricht durch das Anwendungsprogramm ein Fehler auftritt, macht CICS alle Änderungen rückgängig, es sei denn, die Anwendung hat einen Synchronisationspunkt ausgeführt.

Wenn der von ihrem Programm bereitgestellte Web-Service nicht für das Zurückgeben einer Antwort konzipiert ist, ignoriert CICS alle Inhalte im Kommunikationsbereich oder im Container, wenn das Programm zurückkehrt.

Web-Service aus einer Anwendung aufrufen, die mit dem Web-Service-Assistenten implementiert wurde

Eine Service-Requester-Anwendung, die mit dem Web-Service-Assistenten implementiert wird, verwendet den Befehl **EXEC CICS INVOKE SERVICE**, um einen Web-

Service aufzurufen. Anforderung und Antwort werden einer Datenstruktur im Container DFHWS-DATA zugeordnet. Diese Methode zum Aufrufen eines Service wird für JSON nicht unterstützt.

Procedure

1. Erstellen Sie einen Kanal und füllen Sie ihn mit Containern. Es muss mindestens der Container DFHWS-DATA vorhanden sein. DFHWS-DATA enthält die übergeordnete Datenstruktur, die CICS in eine SOAP-Anforderung konvertieren wird. Wenn die SOAP-Anforderung Arrays mit variierender Elementanzahl enthält, werden sie als Serie von verbundenen Containern in einer Serie von Datenstrukturen dargestellt. Diese Container müssen auch im Kanal vorhanden sein.
2. Rufen Sie den Ziel-Web-Service auf. Verwenden Sie den folgenden Befehl:

```
EXEC CICS INVOKE SERVICE(web-service)  
                CHANNEL(benutzerkanal)  
                OPERATION(operation)
```

Dabei gilt Folgendes:

- *web-service* ist der Name der WEBSERVICE-Ressource, die definiert, welcher Web-Service aufgerufen werden soll. Die WEBSERVICE-Ressource gibt die Position der Web-Service-Beschreibung und der Web-Service-Bindungsdatei an, die von CICS zur Kommunikation mit dem Web-Service verwendet werden.
- *benutzerkanal* ist der Kanal, der den Container DFHWS-DATA und andere Container enthält, die der Datenstruktur der Anwendung zugeordnet sind.
- *operation* ist der Name der Operation, die im Ziel-Web-Service aufgerufen werden soll.

Weitere Informationen finden Sie unter „Lokale Optimierung für Web-Services“.

3. Wenn der Befehl erfolgreich war, rufen Sie die Antwortcontainer aus dem Kanal ab. Es ist mindestens der Container DFHWS-DATA vorhanden. Er enthält die übergeordnete Datenstruktur, die CICS aus der SOAP-Nachricht erstellt hat. Wenn die Antwort Arrays mit variierender Elementanzahl enthält, werden sie als Serie von verbundenen Datenstrukturen in einer Serie von Containern dargestellt. Diese Container sind auch im Kanal vorhanden.
4. Wenn der Service-Requester eine SOAP-Fehlernachricht von dem aufgerufenen Web-Service empfängt, müssen Sie entscheiden, ob das Anwendungsprogramm Änderungen zurücksetzen soll. Tritt ein SOAP-Fehler auf, wird ein INVREQ-Fehler mit einem RESP2-Wert von 6 an das Anwendungsprogramm zurückgegeben. Ist die Optimierung jedoch wirksam, wird dieselbe Transaktion sowohl für den Requester als auch für den Provider verwendet. Wenn in einem lokal optimierten Web-Service-Provider ein Fehler auftritt, werden alle von der Transaktion ausgeführten Arbeiten sowohl im Provider, als auch im Requester rückgängig gemacht. Ein INVREQ-Fehler wird an den Requester mit einem RESP2-Wert von 16 zurückgegeben.

Lokale Optimierung für Web-Services

Sie können den Provider-Anwendungsnamen in der Web-Service-Bindungsdatei mit der WEBSERVICE-Ressource verwenden, um eine lokale Optimierung der Web-Service-Anforderung zu aktivieren.

Mit dem Befehl INVOKE SERVICE können Sie die URIMAP(urimap) oder den URI(uri) angeben, wobei es sich um den URI des aufzurufenden Web-Service handelt. Wenn eine URIMAP angegeben wird, verwendet CICS die angegebene Client-

modus-URIMAP, um den URI aufzulösen. Wenn diese Optionen nicht angegeben werden, verwendet CICS den in der Web-Service-Beschreibung angegebenen URI, aus dem der WEBSERVICE generiert wurde.

Wenn der angegebene WEBSERVICE in einer PIPELINE im Requestermodus implementiert wird, ruft CICS den fernen Web-Service auf. Dies ist das gängigste Szenario.

Wenn der angegebene WEBSERVICE in einer PIPELINE im Providermodus implementiert wird, ruft CICS den Service lokal auf. Wenn Sie diese Optimierung verwenden, wird der Befehl EXEC CICS INVOKE SERVICE in einem Befehl EXEC CICS LINK optimiert. Dies führt zu bedeutenden Leistungsvorteilen, bringt aber auch die folgenden Einschränkungen mit sich:

- Die PIPELINE wird nicht gesteuert, deshalb werden keine Handlerprogramme verwendet.
- Es gibt keine PIPELINE-Steuercontainer im Kanal. Manche Container sind vorhanden, darunter DFHWS-DATA, DFHWS-OPERATION und DFHWS-URI. Container, die üblicherweise XML enthalten, sind nicht vorhanden. Dazu zählen DFH-REQUEST, DFHWS-BODY und DFHWS-XMLNS.
- Provider- und Requester-Anwendungen müssen dieselben Copybooks verwenden und in derselben Programmiersprache implementiert sein.
- Provider- und Requester-Anwendungen verwenden eine einzige Arbeitseinheit.
- Wenn der Web-Service keine Antwort zurückgeben soll, gibt der Befehl EXEC CICS INVOKE SERVICE erst dann die Steuerung an die Anwendung zurück, wenn das Zielprogramm beendet wurde.

Wenn Sie lokal optimierte Web-Services verwenden wollen, die Daten aber über eine PIPELINE verarbeitet werden müssen, verwenden Sie das hier beschriebene cics-URI-Format: „Optionen für die Steuerung der Requester-Pipelineverarbeitung“ auf Seite 204. Diese Vorgehensweise ist weniger effizient als die umfassend optimierte Methode, vermeidet aber die Verarbeitungskosten für die Auslagerung in das Netz.

Laufzeiteinschränkungen für vom Web-Service-Assistenten generierten Code

Zur Laufzeit kann CICS fast jede gültige SOAP-Nachricht, die der Web-Service-Beschreibung (WSDL) entspricht, in die entsprechenden Datenstrukturen umsetzen. Es gibt jedoch einige Einschränkungen, die Sie kennen sollten, wenn Sie eine Service-Requester- oder Service-Provider-Anwendung unter Verwendung der Stapeljobs im Web-Service-Assistenten entwickeln.

Codepages

CICS unterstützt das Senden und Empfangen von SOAP-Nachrichten in einer beliebigen Codepage, wenn ein entsprechender HTTP- oder WebSphere MQ-Header die Codepage angibt. CICS konvertiert die SOAP-Nachricht in UTF-8, um sie in der Pipeline zu verarbeiten, bevor sie in die vom Anwendungsprogramm benötigte Codepage umgesetzt wird. Um Leistungseinbußen zu vermeiden, wird empfohlen, die UTF-8-Codepage zu verwenden, wenn SOAP-Nachrichten an CICS gesendet werden. CICS sendet SOAP-Nachrichten immer in UTF-8.

CICS kann SOAP-Nachrichten nur umsetzen, wenn die Anwendungsdaten in EBCDIC oder UTF-16 codiert sind. Anwendungen, die voraussetzen, dass Daten in Codepages wie UTF-8, ASCII und ISO8859-1 codiert sind, werden nicht unterstützt. Wenn Sie DBCS-Zeichen in Ihren Datenstrukturen und SOAP-Nachrichten verwenden

den möchten, müssen Sie eine Codepage angeben, die DBCS unterstützt. Die ausgewählte EBCDIC-Codepage muss sowohl von Java- als auch von z/OS-Konvertierungsservices unterstützt werden. z/OS-Konvertierungsservices müssen außerdem für die Konvertierung aus der Codepage der SOAP-Nachricht in UTF-8 konfiguriert sein. Weitere Informationen zur UTF-16-Unterstützung finden Sie unter Support for UTF-16 in application data.

Um eine entsprechende Codepage festzulegen, können Sie entweder den Systeminitialisierungsparameter **LOCALCCSID** oder den optionalen Parameter **CCSID** in den Jobs des Web-Service-Assistenten verwenden. Wenn Sie den Parameter **CCSID** verwenden, überschreibt der angegebene Wert die Codepage **LOCALCCSID** für diesen spezifischen Web-Service. Wenn Sie den Parameter **CCSID** nicht angeben, wird die Codepage **LOCALCCSID** verwendet, um die Daten umzusetzen, und die Web-Service-Bindungsdatei ist in US EBCDIC (Cp037) codiert.

Container

Wenn Sie im Service-Providermodus angeben, dass der Parameter **PGMINT** den Wert **CHANNEL** hat, muss der Container, der Ihre Anwendungsdaten enthält, im Binärmodus geschrieben und gelesen werden. Standardmäßig ist dies der Container **DFHWS-DATA**. Im Befehl **PUT CONTAINER** muss die Option **DATATYPE** entweder auf **BIT** gesetzt sein oder Sie müssen die Option **FROMCCSID** weglassen, sodass **BIT** die Standardeinstellung bleibt. Der folgende Code gibt beispielsweise explizit an, dass die Daten im Container **CUSTOMER-RECORD** im aktuellen Kanal im Binärmodus geschrieben werden sollen.

```
EXEC CICS PUT CONTAINER (CUSTOMER-RECORD)
              FROM (CREC)
              DATATYPE(BIT)
```

Obwohl die Container selbst alle im **BIT**-Modus sind, müssen alle Textfelder innerhalb der Sprachstruktur, die diese Daten zuordnen, eine EBCDIC-Codepage verwenden - dieselbe Codepage, die Sie im Parameter **LOCALCCSID** oder **CCSID** angegeben haben. Wenn Sie **DFHWS2LS** verwenden, um die Web-Service-Bindungsdatei zu generieren, könnten viele Container im Kanal vorhanden sein, die Teile der vollständigen Datenstruktur enthalten. In diesem Fall müssen die Textfelder in all diesen Containern aus derselben Codepage gelesen und in derselben Codepage geschrieben werden.

Wenn das Anwendungsprogramm Container füllt, die in SOAP-Nachrichten konvertiert werden, muss die Anwendung sicherstellen, dass die Container die richtige Menge von Inhalten enthalten. Wenn ein Container weniger Daten als erwartet enthält, gibt CICS einen Konvertierungsfehler aus.

Wenn ein Anwendungsprogramm den Befehl **INVOKE SERVICE** verwendet, könnte jeder Container, der von dem Programm an CICS übergeben wird, wiederverwendet und die darin enthaltenen Daten könnten ersetzt werden. Wenn Sie die Daten in diesen Containern beibehalten möchten, erstellen Sie einen neuen Kanal und kopieren Sie die Container dorthin, bevor Sie das Programm ausführen. Wenn Sie einen Web-Service im Providermodus haben, bei dem es sich gleichzeitig um einen Web-Service im Requestermodus handelt, ist es empfehlenswert, einen anderen Kanal für den Befehl **INVOKE SERVICE** zu verwenden als den Standardkanal, an den der Befehl ursprünglich angehängt war. Wenn Ihr Anwendungsprogramm den Befehl **INVOKE SERVICE** oft verwendet, ist es empfehlenswert, entweder verschiedene Kanäle für jeden Aufruf von CICS zu verwenden, oder sicherzustellen, dass alle wichtigen Daten aus der ersten Anforderung gespeichert werden, bevor Sie die zweite Anforderung ausgeben.

An Web-Service-Beschreibung anpassen

Eine Web-Service-Beschreibung kann manche Inhalte einer SOAP-Nachricht als optional beschreiben. Wenn dies der Fall ist, ordnet DFHWS2LS Felder innerhalb der generierten Sprachstruktur zu, um anzugeben, ob der Inhalt vorhanden ist oder nicht. Zur Laufzeit füllt CICS diese Felder entsprechend. Wenn ein Feld, z. B. ein vorhandenes Flag oder ein Vorkommenfeld, angibt, dass die Informationen nicht vorhanden sind, sollte das Anwendungsprogramm nicht versuchen, die Felder, die optionalen Inhalten zugeordnet sind, zu verarbeiten.

Wenn in einer SOAP-Nachricht bei der Umsetzung durch CICS Inhalte fehlen, werden die äquivalenten Felder in den Datenstrukturen nicht initialisiert, wenn sie an das Anwendungsprogramm übergeben werden.

Eine Web-Service-Beschreibung kann auch die Regeln zur Verarbeitung von Leeräumen angeben, die beim Lesen einer SOAP-Nachricht verwendet werden sollen, und CICS implementiert diese Regeln zur Laufzeit.

- Wenn der Wert der Facette `xsd:whiteSpace replace` lautet, werden Leerraumzeichen wie „Tabstopp“ und „Wagenrücklauf“ durch Leerzeichen ersetzt.
- Wenn der Wert der Facette `xsd:whiteSpace collapse` lautet, werden alle abschließenden Leerraumzeichen beim Generieren von SOAP-Nachrichten entfernt. Zur Laufzeit werden eingehende SOAP-Nachrichten entsprechend der XML Schema-Spezifikation geparkt und alle führenden, abschließenden und eingebetteten Leerzeichen werden entfernt.

SOAP-Nachrichten

CICS unterstützt keine Ableitung von SOAP-Nachrichteninhalten. Eine SOAP-Nachricht könnte beispielsweise das Attribut `xsi:type` verwenden, um anzugeben, dass ein Element einen bestimmten Typ aufweist, und es mit einem Attribut `xsi:schemaLocation` kombinieren, um die Position des Schemas anzugeben, das das Element beschreibt. CICS bietet keine Unterstützung für das dynamische Abrufen des Schemas und Umsetzen des Elementwerts auf der Grundlage des Inhalts des Schemas. CICS unterstützt das Attribut `xsi:nil`, wenn die im Web-Service-Assistenten angegebene Zuordnungsebene 1.1 oder höher ist, aber dies ist das einzige unterstützte XML Schema-Instanzattribut.

DFHWS2LS muss unter Umständen Annahmen über die maximale Länge oder Größe mancher Werte in der SOAP-Nachricht tätigen. Wenn das XML-Schema beispielsweise keine maximale Länge für ein `xsd:string`-Element angibt, nimmt DFHWS2LS an, dass die maximale Länge 255 Zeichen beträgt, und generiert entsprechend eine Sprachstruktur. Sie können diesen Wert ändern, indem Sie den Parameter **DEFAULT-CHAR-MAXLENGTH** in DFHWS2LS verwenden. Wenn CICS zur Laufzeit eine SOAP-Nachricht erkennt, deren Wert größer als der in der Sprachstruktur zugeordnete Leerraum ist, gibt CICS einen Konvertierungsfehler aus.

Wenn CICS der Service-Provider ist, wird die SOAP-Fehlernachricht an den Requester zurückgegeben. Wenn CICS der Service-Requester ist, wird ein entsprechender RESP2-Code vom Befehl **INVOKE SERVICE** zurückgegeben.

Einige Zeichen haben in XML eine besondere Bedeutung, z. B. die Zeichen `<` und `>`. Wenn eines dieser Sonderzeichen in einem Zeichenarray enthalten ist, das von CICS zur Laufzeit verarbeitet wird, wird es durch die äquivalente Entität ersetzt. Folgende XML-Entitäten werden unterstützt:

Zeichen	XML-Entität
&	&
<	<
>	>
"	"
'	'

CICS unterstützt auch die kanonischen Formen der numerischen Zeichenreferenzen, die für Leerraumcodes verwendet werden:

Zeichen	XML-Entität
Tabstopp		
Wagenrücklauf	

Zeilenvorschub	

Beachten Sie, dass diese Unterstützung nicht für die aufgerufenen Pipelinehandlerprogramme gilt.

Das Nullzeichen (x'00') ist in allen XML-Dokumenten ungültig. Wenn ein Zeichentypfeld, das von dem Anwendungsprogramm bereitgestellt wird, das Nullzeichen enthält, schneidet CICS die Daten an diesem Punkt ab. Auf diese Weise können Sie Zeichenarrays als auf null endende Zeichenfolgen behandeln. Von DFHWS2LS aus base64Binary- oder hexBinary-XML-Schemadatentypen generierte Zeichentypenfelder stellen Binärdaten dar und könnten Nullzeichen ohne Abschneiden enthalten.

Achtung: CICS generiert XML- und JSON-Daten aus strukturierten Anwendungsdaten. Wenn diese Anwendungsdaten Bitmuster enthalten, die wie vorformatierte XML, JSON, HTML, JPEG-Bilder oder andere Inhalte aussehen, erkennt CICS diese semantische Bedeutung nicht und verarbeitet solche Daten als normalen Text oder als Binärdaten. CICS versucht nicht, Muster in den Daten zu erkennen oder codierte Daten auf andere Weise zu verarbeiten. Wenn die Daten beispielsweise vorformatierte XML (wie CDATA-codierten Text) enthalten, werden die Daten auf dieselbe Weise verarbeitet wie alle anderen Daten. Sehen Sie sich die folgenden Anwendungsdaten an: "An example: <here>". Die von einer Beispielanwendung bereitgestellten Daten enthalten scheinbar einen XML-Tag, der aber als unformatierter Text verarbeitet wird, was zu folgender XML-Darstellung führt: "An example: < here > ". Wenn eine Anwendung XML selbst generieren muss, sollten Sie xsd:any-Konstrukte in Ihren XML-Schemas verwenden oder XML-ONLY=TRUE in den Assistenten verwenden.

SOAP-Fehlernachrichten

Wenn CICS der Service-Provider ist und das Anwendungsprogramm eine SOAP-Fehlernachricht ausgeben soll, verwenden Sie den Befehl **SOAPFAULT CREATE**. Um diesen API-Befehl zu verwenden, müssen Sie angeben, dass der Parameter **PGMINT** des Web-Service-Assistenten den Wert **CHANNEL** hat. Wenn Sie diesen Wert nicht angeben und das Anwendungsprogramm ruft den Befehl **SOAPFAULT CREATE** auf, versucht CICS nicht, SOAP-Antwortnachrichten zu generieren.

Pipelineverarbeitung anpassen

Zusätzlich zum Bereitstellen Ihrer eigenen Nachrichtenhandler können Sie auch einen Satz globaler Benutzerexitpunkte (Global User Exit Point, GLUE) verwenden, um die Verarbeitung eingehender und ausgehender Web-Services in der Pipeline anzupassen.

Before you begin

Sie müssen die bewährten Verfahren zum Schreiben globaler Benutzerexitprogramme kennen, bevor Sie die Pipeline anpassen. Wenn Sie eine Service-Provider-Pipeline anpassen, müssen Sie den bereitgestellten DFHPITP- oder Axis2-Anwendungshandler in Ihrer Pipeline verwenden.

About this task

Sie können mit den Pipelinedomänenexits auf Container in einer Web-Service-Provider-Pipeline, einer Web-Service-Requester-Pipeline oder in einer Web-Service-Requester-Pipeline, die einen Sicherheitshandler enthält, zugreifen. Die globalen Benutzerexits der Pipeline werden unter Pipeline domain exits im Detail beschrieben.

Procedure

1. Wählen Sie aus, welche globalen Benutzerexitpunkte verwendet werden sollen:
 - Verwenden Sie XWSPRRWI-, XWSPRROI-, XWSPRROO- oder XWSPRRWO-GLUEs, um auf Container in einer Web-Service-Provider-Pipeline zuzugreifen.
 - Verwenden Sie XWSRQRWO-, XWSRQROO-, XWSROROI- oder XWSRQRWI-GLUEs, um auf Container in einer Web-Service-Requester-Pipeline zuzugreifen.
 - Verwenden Sie XWSSRRWO-, XWSSRROO-, XWSSRROI- oder XWSSRRWI-GLUEs, um auf Container in einer gesicherten Web-Service-Requester-Pipeline zuzugreifen.
2. Verwenden Sie das DFH\$PIEX-Beispielexitprogramm, um Ihr eigenes globales Benutzerexitprogramm zu schreiben. DFH\$PIEX befindet sich in der Bibliothek SDFHSAMP. Es wird empfohlen, dieses Programm threadsicher zu machen.
3. Aktivieren Sie das globale Benutzerexitprogramm.
4. Testen Sie Ihr globales Benutzerexitprogramm, um sicherzustellen, dass es ordnungsgemäß funktioniert.

Optionen für die Steuerung der Requester-Pipelineverarbeitung

In Service-Requester-Pipelines können Nachrichtenhandler bestimmen, wohin die Web-Service-Anforderung gesendet werden soll, indem sie den URI ändern. CICS bietet Unterstützung für verschiedene URI-Formate, sodass die Verarbeitung von Service-Anforderungen durch die Pipeline deutlich flexibler wird.

Wenn die Service-Requester-Pipeline das Ende der Verarbeitung erreicht, haben Sie die folgenden Optionen:

Verknüpfen mit einem Programm

Wenn Sie den URI in das Format `cics://PROGRAM/programm` ändern, wobei *programm* der Name des Zielanwendungsprogramms ist, übergibt CICS den aktuellen Kanal und seine Container oder seinen Kommunikationsbereich mithilfe des Befehls **EXEC CICS LINK** an das Programm.

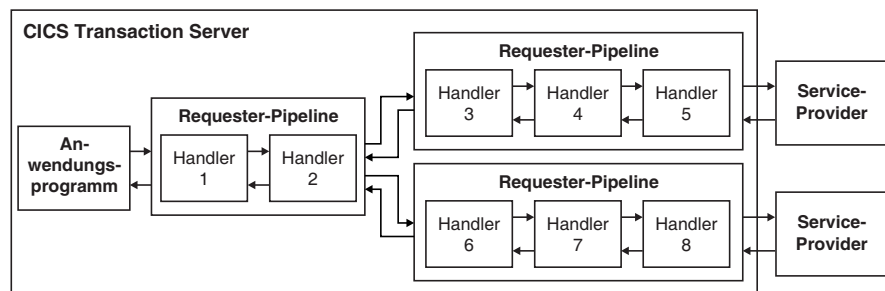
Diese Verarbeitung ähnelt der lokalen Optimierung, die auftritt, wenn sich die Service-Requester- und Service-Provider-Anwendungen in derselben CICS-Region befinden. Die Verwendung dieses URI-Formats bietet jedoch den Vorteil, dass die Anforderung zuerst die Pipeline und alle angepassten Nachrichtenhandler durchläuft. Das Zielanwendungsprogramm muss den Inhalt der Container oder des Kommunikationsbereichs bearbeiten können.

Starten einer weiteren Pipeline im Requestermodus

Wenn Sie den URI in das Format `cics://PIPELINE/`

pipeline?targetServiceUri=zielservice-uri ändern, wobei *pipeline* der Name einer PIPELINE-Ressource und *zielservice-uri* der URI ist, den Sie in den Container DFHWS-URI stellen möchten, übergibt CICS den aktuellen Kanal und seine Container an die angegebene Requester-Pipeline. Verwenden Sie diesen URI, wenn Sie zwei oder mehr Requester-Pipelines miteinander verknüpfen möchten, bevor Sie die Anforderung an den Service-Provider senden. Die Anzahl von Requester-Pipelines, die Sie miteinander verketteten können, ist unbegrenzt.

Im folgenden Beispiel unterstützt eine generische Requester-Pipeline eine Anwendung. Die Nachrichtenhändler 1 oder 2 können den URI für die einzelnen Anforderungen ändern, abhängig von den Anwendungsdaten im Container, und die Anforderungen an eine von zwei Requester-Pipelines senden, die verschiedene Nachrichtenhändler enthalten.



Obwohl in dem Beispiel nur eine Service-Requester-Anwendung gezeigt wird, könnten viele Anwendungen dieselbe generische Requester-Pipeline verwenden und ihre Anforderungen an verschiedene Requester-Pipelines senden, bevor die Anforderung abschließend an den entsprechenden Web-Service-Provider gesendet wird.

Senden der Anforderung direkt an die Pipeline im Providermodus

Wenn Sie den URI in das Format `cics://SERVICE/service?targetServiceUri=zielservice-uri` ändern, wobei *service* der Name des Zielservice und *zielservice-uri* der Pfad zum Service ist, löst CICS die Anforderung auf, indem es den Pfad mit einer URIMAP abgleicht und die Anforderung an die korrekte Provider-Pipeline übergibt. Verwenden Sie diese Option, wenn Sie die Anforderungsverarbeitung sowohl über die Requester- als auch über die Provider-Pipelines durchführen möchten, ohne das Netz zu verwenden.

Dieser URI kann auch nützlich sein, wenn die Requester- und Provider-Anwendungen in verschiedenen Sprachen geschrieben sind oder verschiedene Zuordnungsebenen verwenden und unterschiedliche Binärdaten erwarten.

Requester-Pipelineverarbeitung mit einem URI steuern

In Service-Requester-Pipelines kann ein Nachrichtenhändler bestimmen, wohin die Web-Service-Anforderung gesendet werden soll, indem der URI geändert wird. Indem Sie das URI-Format ändern, können Sie auswählen, dass bestimmte Optimierungen durchgeführt werden sollen, z. B. das Starten einer weiteren Requester-Pipeline oder das Starten einer Service-Provider-Pipeline, ohne die Anforderung über das Netz zu senden.

Before you begin

Entscheiden Sie, welche Optionen in Ihrer Requester-Pipeline implementiert werden sollen. Ausführliche Informationen finden Sie unter „Optionen für die Steuerung der Requester-Pipelineverarbeitung“ auf Seite 204.

About this task

Die Web-Service-Requester-Anwendung kann den Container DFHWS-URI mithilfe des Befehls **EXEC CICS INVOKE SERVICE** füllen oder, wenn von der Anwendung kein Wert bereitgestellt wird, füllt CICS den Container anhand des Werts in der Web-Service-Bindungsdatei. Zum Ändern des URI müssen Sie einen Nachrichtenhandler schreiben, der die Inhalte dieses Containers ändert.

Procedure

1. Schreiben Sie einen Nachrichtenhandler, um den Container DFHWS-URI entsprechend einem der folgenden URI-Formate zu ändern:
 - Verwenden Sie für die Verknüpfung eines Anwendungsprogramms den URI `cics://PROGRAM/programm`, wobei *programm* das Zielanwendungsprogramm ist. Es findet keine Datenumsetzung statt, deshalb müssen Sie sicherstellen, dass das Anwendungsprogramm die Inhalte der Container im aktuellen Kanal verarbeiten kann. Das Anwendungsprogramm kann entweder den aktuellen Kanal und die die aktuellen Container oder einen Kommunikationsbereich übergeben.
 - Um eine Provider-Pipeline zu starten, ohne das Netz dafür zu nutzen, verwenden Sie den URI `cics://SERVICE/service?targetServiceUri=zielservice-uri`, wobei *service* der Name des Service und *zielservice-uri* der Pfad des Service ist. Der Transporthandler verwendet den Pfad des Service, um die URIMAP-Ressource zu lokalisieren, die die Anforderung auflöst und an die korrekte Provider-Pipeline übergibt. CICS verwendet bei der Verarbeitung nicht den Namen des Service.
Es tritt ein Fehler auf, wenn keine URIMAP-Ressource für den Service installiert ist. Die URIMAP-Ressourcendefinition muss auch USAGE(PIPELINE) angeben. Der Transporthandler stellt den Wert des Parameters **targetServiceUri** in den Container DFHWS-URI und startet die Provider-Pipeline.
 - Um eine weitere Requester-Pipeline zu starten, verwenden Sie den URI `cics://PIPELINE/pipeline?targetServiceUri=zielservice-uri`, wobei *pipeline* der Name der PIPELINE-Ressource ist, die Sie starten möchten, und *zielservice-uri* der Wert ist, den Sie im Container DFHWS-URI an die nächste Pipeline übergeben möchten.

Jeder Typ von URI hat zusätzliche Parameter, die Sie als Abfragezeichenfolge hinzufügen können. Weitere Informationen zum Format dieser URIs und den Regeln für ihre Codierung finden Sie unter „Container DFHWS-URI“ auf Seite 181.

2. Verwenden Sie einen XML-Editor, um den Nachrichtenhandler zur Pipelinekonfigurationsdatei hinzuzufügen:

```
<service>
  <service_handler_list>
    <handler>
      <program>MYPROG</program>
    </handler>
  </service_handler_list>
</service>
```

3. Inaktivieren, löschen und installieren Sie die PIPELINE-Ressource für die Requester-Pipeline neu, um Ihr neues Nachrichtenhandlerprogramm in die Pipeline einzuschließen.
4. Installieren Sie das Nachrichtenhandlerprogramm in der CICS-Region.

Results

Die nächste Serviceanforderung, die über die Requester-Pipeline ausgeführt werden soll, wird von Ihrem neuen Nachrichtenhandler verarbeitet.

What to do next

Testen Sie die Änderungen an Ihrer Requester-Pipeline, um sicherzustellen, dass die Serviceanforderungen an der richtigen Stelle landen und dass Ihr Nachrichtenhandlerprogramm das erwartete Verhalten zeigt.

Unterstützung für Web-Service-Transaktionen

Die Spezifikationen WSAT (Web Services Atomic Transaction oder WS-AtomicTransaction) und WSC (Web Services Coordination oder WS-Coordination) definieren Protokolle für kurzfristige Transaktionen, mit deren Hilfe Transaktionsverarbeitungssysteme in einer Web-Service-Umgebung zusammenarbeiten können. Transaktionen, die WS-AtomicTransaction verwenden, besitzen die so genannten *ACID*-Eigenschaften (atomar, konsistent, isoliert und permanent).

Die Spezifikationen finden Sie unter OASIS.

Anmerkung: CICS unterstützt die Version der Spezifikationen vom November 2004.

CICS-Anwendungen, die als Web-Service-Provider oder Web-Service-Requester implementiert werden, können an verteilten Transaktionen mit anderen Web-Service-Implementierungen teilnehmen, die die Spezifikationen unterstützen.

Registrierungsservices

Registrierungsservices sind Teil des WS-Coordination-Modells, das einer Anwendung ermöglicht, sich für Koordinationsprotokolle zu registrieren. In einer verteilten Transaktion kommunizieren die Registrierungsservices in den teilnehmenden Systemen miteinander, um den verbundenen Anwendungen zu ermöglichen, an diesen Protokollen teilzunehmen.

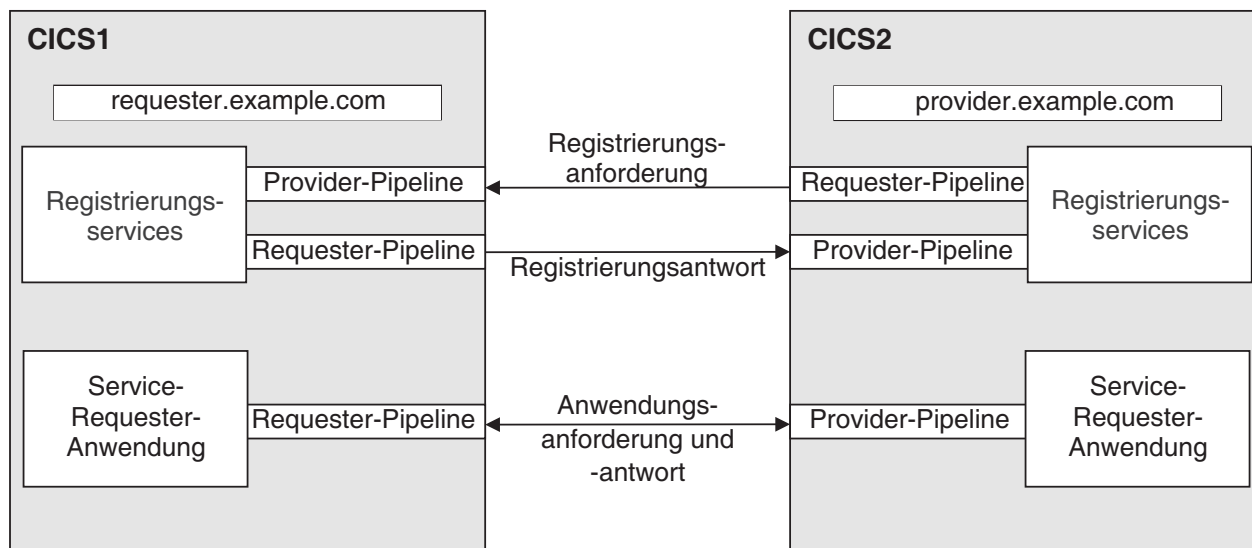


Abbildung 24. Registrierungsservices

Abb. 24 zeigt zwei CICS-Systeme, CICS1 und CICS2. Eine Service-Requester-Anwendung in CICS1 ruft eine Service-Provider-Anwendung in CICS2 auf. Die beiden CICS-Regionen und die Anwendungen werden so konfiguriert, dass die beiden Anwendungen an einer einzelnen verteilten Transaktion teilnehmen, unter Verwendung der WS-Coordination-Protokolle. Die Service-Requester-Anwendung ist der Koordinator und die Service-Provider-Anwendung ist der Teilnehmer.

Zur Unterstützung dieser Protokolle interagieren die Registrierungsservices in den beiden CICS-Regionen zu Beginn der Transaktion und dann erneut, während die Transaktion beendet wird. Während dieser Interaktionen können die Registrierungsservices in beiden Regionen zu unterschiedlichen Zeiten als Service-Provider und als Service-Requester fungieren. Deshalb verwenden die Registrierungsservices in jeder Region eine Service-Provider-Pipeline und eine Service-Requester-Pipeline. Die Pipelines werden für CICS mit dem PIPELINE-Element und zugehörigen Ressourcen definiert.

Die Registrierungsservices in den einzelnen Regionen sind einer Endpunktadresse zugeordnet. Deshalb haben in diesem Beispiel die Registrierungsservices in CICS1 die Endpunktadresse `requester.example.com` und in CICS2 die Endpunktadresse `provider.example.com`.

In einem CICSplex können Sie die Provider-Pipeline der Registrierungsservices auf eine andere Region verteilen. Dies wird in Abb. 25 auf Seite 209 gezeigt.

CICS für Web-Service-Transaktionen konfigurieren

Damit Web-Service-Requester- und Web-Service-Provider-Anwendungen an Web-Service-Transaktionen teilnehmen, müssen Sie CICS entsprechend konfigurieren, indem Sie eine Reihe von CICS-Ressourcen installieren.

Before you begin

Bevor Sie diese Ressourcen installieren können, müssen Sie die Position der Pipelinekonfigurationsdateien kennen, die CICS zur Unterstützung von Web-Service-Transaktionen bereitstellt. Standardmäßig werden die Konfigurationsdateien im Verzeichnis `/usr/lpp/cicsts/cicsts55/pipeline/configs` bereitgestellt, aber der Standarddateipfad wurde möglicherweise während der CICS-Installation geändert.

About this task

Die CICS-Unterstützung für Web-Service-Transaktionen verwendet einen von CICS bereitgestellten Registrierungsservice. Dieser Registrierungsservice besteht aus einem Service-Provider und einem Service-Requester. Sie müssen Ressourcen sowohl für den Service-Provider als auch für den Service-Requester installieren, selbst wenn Ihre Anwendungen alle Service-Provider oder alle Service-Requester sind.

Sie müssen auch eine Programmdefinition für das Header-Handlerprogramm installieren, das aufgerufen wird, wenn Sie Ihre Service-Provider- und Service-Requester-Anwendungen ausführen.

Die Ressourcen, die für die Konfiguration von CICS für Web-Service-Transaktionen erforderlich sind, werden alle in der Gruppe DFHWSAT bereitgestellt, außer DFHPIDIR, das in einer der folgenden Gruppen bereitgestellt wird: DFHPIVS, DFHPIVR oder DFHPICF. Die Gruppe DFHWSAT ist nicht in der Liste DFHLIST enthalten und wird deshalb nicht automatisch installiert. Sie können die von CICS in der Gruppe DFHWSAT bereitgestellten Ressourcen nicht ändern.

Gehen Sie wie folgt vor, um CICS für Web-Service-Transaktionen zu konfigurieren:

Procedure

1. Fügen Sie den Datensatz DFHPIDIR zu Ihrer Start-JCL hinzu. DFHPIDIR speichert eine Zuordnung zwischen Kontexten und Tasks.
 - a. Fügen Sie eine neue DD-Anweisung für den Datensatz DFHPIDIR zu Ihrer CICS-Start-JCL hinzu.
 - b. Erstellen Sie den Datensatz DFHPIDIR anhand von Informationen in DFHDEFDS.JCL. Die standardmäßige Datensatzgröße (RECORDSIZE) von DFHPIDIR ist 1 KB, das sich für die meisten Verwendungszwecke eignet. Sie können DFHPIDIR bei Bedarf mit einer größeren Datensatzgröße (RECORDSIZE) erstellen.
 - c. Installieren Sie die entsprechende Gruppe für den Datensatz in Ihrem CICS-System: DFHPIVS, DFHPIVR oder DFHPICF. Weitere Informationen zu diesen Gruppen finden Sie unter Defining the WS-AT data set.

Wenn Sie die Datei DFHPIDIR über CICS-Regionen hinweg teilen möchten, müssen die Regionen logisch über MRO verbunden sein. Sie müssen einen Datensatz pro Gruppe von Regionen erstellen, die als logischer Server fungieren.

Tip: Es wird empfohlen, keine Datensätze zwischen Regionen zu teilen, die nicht logisch miteinander verbunden sind.

2. Kopieren Sie den Inhalt der Gruppe DFHWSAT in eine andere Gruppe. Sie können die von CICS in der Gruppe DFHWSAT bereitgestellten Ressourcen nicht ändern. Sie müssen jedoch das Attribut CONFIGFILE in den PIPELINE-Ressourcen ändern.
3. Ändern Sie die PIPELINE-Ressource des Providers des Registrierungsservice. Die PIPELINE heißt DFHWSATP und gibt die Pipelinekonfigurationsdatei /usr/lpp/cicsts/cicsts55/pipeline/configs/registrationservicePROV.xml im Attribut CONFIGFILE an.
 - a. Ändern Sie das Attribut CONFIGFILE, um die Position der Datei in Ihrem System abzubilden.
 - b. Lassen Sie die anderen Attribute unverändert.

Verwenden Sie die Pipelinekonfigurationsdatei genau wie sie bereitgestellt wird. Ändern Sie ihren Inhalt nicht.

4. Installieren Sie die PIPELINE-Ressource. Die PIPELINE-Ressource des Providers der Registrierungsservices muss sich nicht in derselben CICS-Region wie Ihre Service-Requester- oder Service-Provider-Anwendungen befinden, aber sie muss mit dieser Region über eine geeignete MRO- oder APPC-Verbindung verbunden sein.
5. Installieren Sie die unveränderte URIMAP, die von Registrierungsservices-Provider verwendet wird, in derselben Region wie die PIPELINE. Die URIMAP heißt DFHRSURI.
6. Ändern Sie die PIPELINE-Ressource des Requesters des Registrierungsservice. Die PIPELINE heißt DFHWSATR und gibt die Pipelinekonfigurationsdatei /usr/lpp/cicsts/cicsts55/pipeline/configs/registrationserviceREQ.xml im Attribut CONFIGFILE an.
 - a. Ändern Sie das Attribut CONFIGFILE, um die Position der Datei in Ihrem System abzubilden.
 - b. Lassen Sie die anderen Attribute unverändert.

Verwenden Sie die Pipelinekonfigurationsdatei genau wie sie bereitgestellt wird. Ändern Sie ihren Inhalt nicht.

7. Installieren Sie die PIPELINE-Ressource. Die PIPELINE-Ressource des Registrierungsservices-Requester muss sich in derselben CICS-Region wie die Service-Requester- und Service-Provider-Anwendungen befinden.
8. Installieren Sie die Programme, die von der Provider-Pipeline des Registrierungsservice verwendet werden, in derselben Region wie Ihre PIPELINE-Ressourcen. Die Programme sind DFHWSATX, DFHWSATR und DFHPIRS. Wenn sich Ihre beiden PIPELINE-Ressourcen in verschiedenen Regionen befinden, müssen Sie diese Programme in beiden Regionen installieren.
9. Installieren Sie die PROGRAM-Ressourcendefinition für das Header-Handlerprogramm. Das Programm heißt DFHWSATH. Installieren Sie das PROGRAM in den Regionen, in denen Ihre Service-Provider- und Service-Requester-Anwendungen ausgeführt werden.

Results

CICS ist jetzt so konfiguriert, dass Ihre Service- Provider- und Service-Requester-Anwendungen in verteilten Transaktionen unter Verwendung von WS-AtomicTransaction- und WS-Coordination-Protokollen teilnehmen können.

What to do next

Jetzt müssen Sie die teilnehmenden Anwendungen einzeln konfigurieren.

Service-Provider für Web-Service-Transaktionen konfigurieren

Wenn eine Service-Provider-Anwendung an Web-Service-Transaktionen teilnehmen soll, muss die Pipelinekonfigurationsdatei ein `<headerprogram>`-Element und ein `<service_parameter_list>`-Element angeben.

Before you begin

Wenn Ihre Service-Provider-Anwendung an Web-Service-Transaktionen teilnehmen soll, muss sie SOAP-Protokolle für die Kommunikation mit dem Service-Requester verwenden, und Ihre Pipeline muss so konfiguriert sein, dass sie einen der von CICS bereitgestellten SOAP-Nachrichtenhandlern verwendet. Selbst wenn Sie Ihre Service-Provider-Anwendung korrekt konfiguriert haben, nimmt sie nur an Web-Service-Transaktionen mit dem Service-Requester teil, wenn die Requester-Anwendung ebenfalls für die Teilnahme eingerichtet wurde.

About this task

Zusätzlich zu den Pipelinekonfigurationsinformationen, die für Ihre Anwendung spezifisch sind, muss die Konfigurationsdatei Informationen enthalten, die CICS nutzt, um sicherzustellen, dass Ihre Anwendung an Web-Service-Transaktionen teilnimmt.

CICS stellt ein Beispiel für eine Pipelinekonfigurationsdatei bereit, die diese Informationen im Verzeichnis `/usr/lpp/cicsts/cicsts55/samples/pipelines/wsatprovider.xml` enthält (wobei `/usr/lpp/cicsts/cicsts55` das Standardinstallationsverzeichnis für CICS-Dateien unter z/OS UNIX) ist.

Procedure

1. Codieren Sie in der Definition Ihres Terminal-Handlers ein Element `<headerprogram>` im Element `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` oder `<cics_soap_1.2_handler_java>`. Codieren Sie die Elemente `<program_name>`, `<namespace>`, `<localname>` und `<mandatory>` genau wie in diesem Beispiel beschrieben:

```
<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>
```

Schließen Sie weitere `<headerprogram>`-Elemente ein, falls Ihre Anwendung diese benötigt.

2. Codieren Sie ein Element `<registration_service_endpoint>` in einem Element `<service_parameter_list>`. Codieren Sie das Element `<registration_service_endpoint>` wie folgt:

```
<registration_service_endpoint>
http://adresse:port/cicswsat/RegistrationService
</registration_service_endpoint>
```

adresse ist die IP-Adresse der CICS-Region, in der sich die Registrierungs-service-Provider-Pipeline befindet.

port ist die Portnummer, die von der Registrierungsservice-Provider-Pipeline verwendet wird.

Codieren Sie alles andere genau wie angezeigt. Die Zeichenfolge `cicswsat/RegistrationService` stimmt mit dem Attribut `PATH` von `URIMAP DFHRSURI` überein:

```
<registration_service_endpoint>
http://provider.example.com:7160/cicswsat/RegistrationService
</registration_service_endpoint>
```

Service-Requester für Web-Service-Transaktionen konfigurieren

Wenn eine Service-Requester-Anwendung an Web-Service-Transaktionen teilnehmen soll, muss die Pipelinekonfigurationsdatei ein `<headerprogram>`-Element und ein `<service_parameter_list>`-Element angeben.

Before you begin

Wenn Ihre Service-Requester-Anwendung an Web-Service-Transaktionen teilnehmen soll, muss sie SOAP-Protokolle für die Kommunikation mit dem Service-Provider verwenden, und Ihre Pipeline muss so konfiguriert sein, dass sie einen der von CICS bereitgestellten SOAP-Nachrichtenhandlern verwendet. Selbst wenn Sie Ihre Service-Requester-Anwendung korrekt konfiguriert haben, nimmt sie nur an Web-Service-Transaktionen mit dem Service-Provider teil, wenn die Provider-Anwendung ebenfalls für die Teilnahme eingerichtet wurde.

About this task

Zusätzlich zu den Pipelinekonfigurationsinformationen, die für Ihre Anwendung spezifisch sind, muss die Konfigurationsdatei Informationen enthalten, die CICS nutzt, um sicherzustellen, dass Ihre Anwendung an Web-Service-Transaktionen teilnimmt.

CICS stellt ein Beispiel für eine Pipelinekonfigurationsdatei bereit, die diese Informationen im Verzeichnis `/usr/lpp/cicsts/cicsts55/samples/pipelines/wsatrequester.xml` enthält (wobei `/usr/lpp/cicsts/cicsts55` das Standardinstallationsverzeichnis für CICS-Dateien unter z/OS UNIX) ist.

Procedure

1. Codieren Sie ein Element `<headerprogram>` im Element `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` oder `<cics_soap_1.2_handler_java>`. Codieren Sie die Elemente `<program_name>`, `<namespace>`, `<localname>` und `<mandatory>` genau wie im folgenden Beispiel beschrieben:

```
<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>
```

Sie können weitere `<headerprogram>`-Elemente einschließen, falls Ihre Anwendung diese benötigt.

2. Codieren Sie ein Element `<registration_service_endpoint>` in einem Element `<service_parameter_list>`. Codieren Sie das Element `<registration_service_endpoint>` wie folgt:

```
<registration_service_endpoint>
http://adresse:port/cicswsat/RegistrationService
</registration_service_endpoint>
```

adresse ist die IP-Adresse der CICS-Region, in der sich die Registrierungs-service-Provider-Pipeline befindet.

port ist die Portnummer, die von der Registrierungsservice-Provider-Pipeline verwendet wird.

Zwischen dem Beginn des Elements `<registration_service_endpoint>`, seinem Inhalt und dem Ende des Elements `<registration_service_endpoint>` darf kein Leerzeichen sein. Die Leerzeichen in diesem Beispiel wurden aus Gründen der Übersichtlichkeit eingefügt.

3. Wenn CICS einen neuen transaktionsorientierten Kontext für jede Anforderung erstellen soll, statt denselben Kontext für Anforderungen in derselben Arbeitseinheit zu verwenden, fügen Sie das leere Element, `<new_tx_context_required/>`, in einem Element `<service_parameter_list>` zu Ihrer Pipelinekonfigurationsdatei hinzu:

```
<service_parameter_list>
  <registration_service_endpoint>
  http://requester.example.com:7159/cicswsat/RegistrationService
  </registration_service_endpoint>
  <new_tx_context_required/>
</service_parameter_list>
```

Zwischen dem Beginn des Elements `<registration_service_endpoint>`, seinem Inhalt und dem Ende des Elements `<registration_service_endpoint>` darf kein Leerzeichen sein. Die Leerzeichen in diesem Beispiel wurden aus Gründen der Übersichtlichkeit eingefügt.

Die Einstellung `<new_tx_context_required/>` ist nicht die Standardeinstellung für CICS und nicht in der Beispielkonfigurationsdatei für die Pipeline, `wsatprovider.xml`, enthalten. Wenn Sie `<new_tx_context_required/>` in einem Element `<service_parameter_list>` zu Ihrer Pipelinekonfigurationsdatei hinzufügen, sind Loopback-Aufrufe von CICS zulässig, deshalb kann in dieser Situation ein Deadlock auftreten.

Feststellen, ob die SOAP-Nachricht Teil einer atomaren Transaktion ist

Wenn ein CICS-Web-Service in der atomaren Transaktionspipeline aufgerufen wird, muss die SOAP-Nachricht nicht zwangsläufig Teil einer atomaren Transaktion sein.

About this task

Das Element `<soapenv:Header>` enthält spezifische Informationen, wenn die SOAP-Nachricht Teil einer atomaren Transaktion ist. Um herauszufinden, ob die SOAP-Nachricht Teil einer atomaren Transaktion ist, können Sie Folgendes tun:

Procedure

- Durchsuchen Sie den Inhalt des Elements `<soapenv:Header>` mithilfe eines Trace.
 1. Führen Sie einen Hilfstrace unter Verwendung der Komponente PI aus und setzen Sie die Tracestufe auf 2.
 2. Suchen Sie nach dem Tracepunkt PI 0A31, der die Informationen für den Anforderungscontainer enthält. Suchen Sie insbesondere nach PIIS EVENT - REQUEST_CNT, das direkt vor dem Element `<cicswsa:Action>` angezeigt wird.
- Verwenden Sie ein benutzerdefiniertes Nachrichtenhandlerprogramm in der Pipeline DFHWSATP, um den Inhalt des Containers DFHREQUEST anzuzeigen,

wenn er die RECEIVE-REQUEST-Daten enthält. Wenn Sie sich für diesen Ansatz entscheiden, stellen Sie sicher, dass Sie das Nachrichtenhandlerprogramm in der Pipelinekonfigurationsdatei definieren.

Example

Das folgende Beispiel zeigt, welche Informationen Sie im SOAP-Envelope-Header für eine atomare Transaktion sehen können.

```
<soapenv:Header>
  <wscoor:CoordinationContext soapenv:mustUnderstand="1"> 1
    <wscoor:Expires>500</wscoor:Expires>
    <wscoor:Identifier>com.ibm.ws.wstx:
      0000010a2b5008c80000000200000019a75aab901a1758a4e40e2731c61192a10ad6e921
    </wscoor:Identifier>
    <wscoor:CoordinationType>http://schemas.xmlsoap.org/ws/2004/10/wsat</wscoor:CoordinationType> 2
    <wscoor:RegistrationService 3
      xmlns:wscoor="http://schemas.xmlsoap.org/ws/2004/10/wscoor">
        <cicswsa:Address xmlns:cicswsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
          http://clientIPaddress:clientPort/_IBMSYSAPP/wscoor/services/RegistrationCoordinatorPort
        </cicswsa:Address>
        <cicswsa:ReferenceProperties
          xmlns:cicswsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
          <websphere-wsat:txID
            xmlns:websphere-wsat="http://wstx.Transaction.ws.ibm.com/extension">com.ibm.ws.wstx:
              0000010a2b5008c80000000200000019a75aab901a1758a4e40e2731c61192a10ad6e921
          </websphere-wsat:txID>
          <websphere-wsat:instanceID
            xmlns:websphere-wsat="http://wstx.Transaction.ws.ibm.com/extension">com.ibm.ws.wstx:
              0000010a2b5008c80000000200000019a75aab901a1758a4e40e2731c61192a10ad6e921
          </websphere-wsat:instanceID>
        </cicswsa:ReferenceProperties>
      </wscoor:RegistrationService>
    </wscoor:CoordinationContext>
  </soapenv:Header>
```

1. 'CoordinationContext' gibt an, dass die SOAP-Nachricht an einer atomaren Transaktion teilnehmen soll. Das Element enthält die erforderlichen Informationen, damit der Web-Service-Provider Teil des Koordinationsservice sein kann, vorausgesetzt, der Provider ist für die Erkennung und Verarbeitung des Headers konfiguriert.
2. 'CoordinationType' gibt die Version der WS-AT-Spezifikation an, mit der der Koordinationsservice kompatibel ist.
3. 'RegistrationService' der Koordination beschreibt, wo der Registrierungspunkt des Koordinators ist und stellt die Informationen bereit, die der teilnehmende Web-Service an den Koordinator zurückgeben muss, wenn er versucht, sich als Komponente der atomaren Transaktion zu registrieren.

Fortschritt einer atomaren Transaktion prüfen

Wenn ein CICS-Web-Service als Teil einer atomaren Transaktion aufgerufen wird, durchläuft die Transaktion eine Reihe von Status. Diese Status geben an, ob die Transaktion erfolgreich war oder ob sie rückgängig gemacht werden musste.

About this task

Wenn Sie auf diese Informationen zugreifen müssen, können Sie Folgendes tun:

Procedure

- Durchsuchen Sie den Inhalt des Elements <cicswsa:Action> mithilfe eines Trace.
 1. Führen Sie einen Hilfstrace unter Verwendung der Komponente PI aus und setzen Sie die Tracestufe auf 2.

2. Suchen Sie nach dem Tracepunkt PI 0A31, der die Informationen für den Anforderungscontainer enthält. Suchen Sie insbesondere nach PIIS EVENT - REQUEST_CNT, das direkt vor dem Element <cicswsa:Action> angezeigt wird.
- Verwenden Sie ein benutzerdefiniertes Nachrichtenhandlerprogramm in den Pipelines DFHWSATR und DFHWSATP, um den Inhalt der DFHWS-SOAPACTION-Container anzuzeigen. Wenn Sie sich für diesen Ansatz entscheiden, stellen Sie sicher, dass Sie das Nachrichtenhandlerprogramm in den Pipelinekonfigurationsdateien definieren.

Example

Die Status für eine Transaktion, die erfolgreich abgeschlossen und festgeschrieben wurde, lauten wie folgt:

```
"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 "
```

Die Status für eine Transaktion, die rückgängig gemacht wurde, lauten:

```
"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"
```

Unterstützung für MTOM/XOP-Optimierung von Binärdaten

In standardmäßigen SOAP-Nachrichten sind Binärobjekte base64-codiert und in den Nachrichtenhauptteil eingebunden, was ihre Größe um 33 % erhöht. Bei sehr großen Binärobjekten kann dieser Größenzuwachs erhebliche Auswirkungen auf die Übertragungszeit haben. Die Implementierung von MTOM/XOP bietet eine Lösung für dieses Problem.

Die Spezifikationen von SOAP Message Transmission Optimization Mechanism (MTOM) und XML-binary Optimized Packaging (XOP), oft als MTOM/XOP bezeichnet, definieren eine Methode zur Optimierung der Übertragung von großen base64Binary-Datenobjekten in SOAP-Nachrichten.

- Die MTOM-Spezifikation definiert konzeptionell eine Methode zur Optimierung von SOAP-Nachrichten durch Trennen der binären Daten, die ansonsten base64-codiert würden, und das Senden dieser Daten in separaten binären Anhängen unter Verwendung einer MIME-Multipart/Related-Nachricht. Dieser Typ von MIME-Nachricht wird als *MTOM-Nachricht* bezeichnet. Das Senden der Daten im Binärformat reduziert ihre Größe beträchtlich und optimiert so die Übertragung der SOAP-Nachricht.
- Die XOP-Spezifikation definiert eine Implementierung zur Optimierung von XML-Nachrichten mithilfe von binären Anhängen in einem Paketformat, das MIME-Nachrichten einschließt, aber nicht darauf beschränkt ist.

CICS implementiert die Unterstützung für diese Spezifikationen sowohl in Requester- als auch in Provider-Pipelines, wenn das Transportprotokoll WebSphere MQ, HTTP oder HTTPS ist. Als Alternative zum Hinzufügen der base64Binary-Daten direkt in der SOAP-Nachricht können CICS-Anwendungen, die als Web-Service-Provider oder als Web-Service-Requester implementiert sind, diese Unterstützung verwenden, um MTOM-Nachrichten mit binären Anhängen zu senden und zu empfangen.

Sie können diese Unterstützung über zusätzliche Optionen in der Pipelinekonfigurationsdatei konfigurieren.

MTOM/XOP und SOAP

Wenn MTOM/XOP zur Optimierung einer SOAP-Nachricht verwendet wird, wird es unter Verwendung der XOP-Verarbeitung in eine MIME Multipart/Related-Nachricht serialisiert. Die base64Binary-Daten werden aus der SOAP-Nachricht extrahiert und als separate binäre Anhänge innerhalb der MIME-Nachricht gepackt, ähnlich wie E-Mail-Anhänge.

Die Größe der base64Binary-Daten wird deutlich reduziert, da die Anhänge im Binärformat codiert sind. Anschließend wird die XML in der SOAP-Nachricht in das XOP-Format konvertiert, indem die base64Binary-Daten durch ein spezielles `<xop:Include>`-Element ersetzt werden, das den relevanten MIME-Anhang mithilfe eines URI referenziert.

Die geänderte SOAP-Nachricht wird als *XOP-Dokument* bezeichnet und bildet das Stammdokument in der Nachricht. Das XOP-Dokument und die binären Anhänge bilden zusammen das *XOP-Paket*. Wenn das XOP-Paket auf die SOAP-MTOM-Spezifikation angewendet wird, handelt es sich um eine MIME-Nachricht im MTOM-Format.

Das Stammdokument wird angegeben, indem seine Inhalts-ID im übergeordneten 'content-type'-Header der MIME-Nachricht referenziert wird. Im Folgenden sehen Sie ein Beispiel für einen 'content-type'-Header:

```
Content-Type: Multipart/Related; boundary=MIME_boundary;  
type="application/soap+xml"; start="<claim@insurance.com>"
```

Der Parameter **start** enthält die Inhalts-ID des XOP-Dokuments. Wenn dieser Parameter nicht im 'content-type'-Header enthalten ist, wird von dem ersten Teil in der MIME-Nachricht angenommen, dass es sich um das XOP-Dokument handelt.

Die Reihenfolge der Anhänge in der MIME-Nachricht ist nicht wichtig. In manchen Nachrichten können die binären Anhänge beispielsweise vor dem XOP-Dokument vorkommen. Eine Anwendung, die MIME-Nachrichten verarbeitet, darf sich nicht darauf verlassen, in welcher Reihenfolge die Anhänge vorkommen. Ausführliche Informationen finden Sie in den MTOM/XOP-Spezifikationen.

Das folgende Beispiel veranschaulicht, wie eine einfache SOAP-Nachricht, die ein JPEG-Bild enthält, mithilfe der XOP-Verarbeitung optimiert wird. Die SOAP-Nachricht lautet wie folgt:

```
<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>
```

Eine MTOM/XOP-Version dieser SOAP-Nachricht lautet wie folgt:

```
MIME-Version: 1.0  
Content-Type: Multipart/Related; boundary=MIME_boundary;  
type="application/soap+xml"; start="<claim@insurance.com>" ❶  
  
--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. Der Parameter **start** gibt an, welcher Teil der MIME-Nachricht das XOP-Stammdokument ist.
2. Der Wert von 'Content-ID' gibt einen Teil der MIME-Nachricht an. In diesem Fall ist es das XOP-Stammdokument.
3. Das Element <xop:Include> referenziert die binäre JPEG-Anlage.
4. 'Content-ID' gibt das JPEG im binären Anhang an.

MTOM-Nachrichten und binäre Anhänge in CICS

CICS unterstützt und steuert die Verarbeitung von MTOM-Nachrichten sowohl in Web-Service-Provider- als auch in Web-Service-Requester-Pipelines mithilfe eines MTOM-Handlerprogramms und XOP-Verarbeitung.

Sie konfigurieren und aktivieren die MTOM-Unterstützung mithilfe der Pipelinekonfigurationsdatei. Wenn MTOM-Unterstützung für eine Pipeline aktiviert wird, entpackt MTOM eingehende MTOM-Nachrichten automatisch und packt ausgehende Nachrichten. Wenn MTOM-Unterstützung für eine Pipeline nicht aktiviert ist und CICS eine MTOM-Nachricht empfängt, akzeptieren Java-basierte Pipelines die eingehende MTOM-Nachricht. Andere SOAP-Pipelines weisen die eingehende MTOM-Nachricht jedoch mit einem SOAP-Fehler zurück.

Konfigurationsoptionen für Java-basierte Pipelines

Sie können eine Provider-Pipeline konfigurieren, um die folgenden Tasks auszuführen:

- Akzeptieren von MTOM-Nachrichten, aber kein Senden von MTOM-Antwortnachrichten.
- Akzeptieren von MTOM-Nachrichten und Senden von MTOM-Antwortnachrichten.
- Verarbeiten von XOP-Dokumenten und binären Anhängen im Axis2-Modus.

Sie können eine Requester-Pipeline konfigurieren, um die folgenden Tasks auszuführen:

- Kein Senden von MTOM-Nachrichten, aber Akzeptieren von MTOM-Antwortnachrichten.
- Senden von MTOM-Nachrichten und Akzeptieren von MTOM-Antwortnachrichten.
- Verarbeiten von XOP-Dokumenten und binären Anhängen im Axis2-Modus.

Konfigurationsoptionen für Pipelines, die Java nicht unterstützen

Sie können eine Provider-Pipeline konfigurieren, um die folgenden Tasks auszuführen:

- Akzeptieren von MTOM-Nachrichten, aber kein Senden von MTOM-Antwortnachrichten.
- Akzeptieren von MTOM-Nachrichten und Senden desselben Typs von Antwortnachricht.
- Akzeptieren von MTOM-Nachrichten, aber Senden von MTOM-Nachrichten nur, wenn binäre Anhänge vorhanden sind.
- Akzeptieren von MTOM-Nachrichten und Senden von MTOM-Antwortnachrichten.
- Verarbeiten von XOP-Dokumenten und binären Anhängen im Direktmodus oder im Kompatibilitätsmodus.

Sie können eine Requester-Pipeline konfigurieren, um die folgenden Tasks auszuführen:

- Kein Senden von MTOM-Nachrichten, aber Akzeptieren von MTOM-Antwortnachrichten.
- Senden von MTOM-Nachrichten nur, wenn binäre Anhänge vorhanden sind, und Akzeptieren von MTOM-Nachrichten.
- Senden von MTOM-Nachrichten und Akzeptieren von MTOM-Antwortnachrichten.
- Verarbeiten von XOP-Dokumenten und binären Anhängen im Direktmodus oder im Kompatibilitätsmodus.

Unterstützungsmodi

In der Pipeline werden drei Unterstützungsmodi für die Verarbeitung von XOP-Dokumenten und allen zugeordneten binären Anhängen bereitgestellt.

Axis2-Modus

Der Axis2-Modus wird verwendet, wenn der Terminal-Handler Ihrer Web-Service-Pipeline entweder der Nachrichtenhandler `<cics_soap_1.1_handler_java>` oder der Nachrichtenhandler `<cics_soap_1.2_handler_java>` ist.

Direktmodus

Im Direktmodus werden die binären Anhänge, die einer eingehenden oder ausgehenden MTOM-Nachricht zugeordnet sind, in Containern über die Pipeline übergeben und direkt von der Anwendung verarbeitet, ohne dass eine Datenkonvertierung ausgeführt werden muss.

Kompatibilitätsmodus

Der Kompatibilitätsmodus wird verwendet, wenn die Pipelineverarbeitung erfordert, dass die Nachricht im Standard-XML-Format vorliegt, wobei alle binären Daten als base64Binary-Felder in der Nachricht gespeichert sind. Für eingehende Nachrichten werden das XOP-Dokument und die binären Anhänge in einer XML-Standardnachricht zusammengefasst, entweder am Anfang der Pipeline, wenn Web Services Security aktiviert ist, oder am Ende der Pipeline, wenn die Web-Service-Validierung aktiviert ist. Für ausgehende Nachrichten wird eine XML-Standardnachricht erstellt und entlang der Pipeline weitergegeben. Diese XML-Nachricht wird vom MTOM-Handler in das XOP-Format konvertiert, unmittelbar bevor CICS sie sendet.

Der Kompatibilitätsmodus ist wesentlich weniger effizient als der Direktmodus, da Binärdaten in das base64-Format und wieder zurück konvertiert werden. Er ermöglicht jedoch, dass Ihre Web-Services mit anderen MTOM/XOP-Web-Service-Requestern und -Providern arbeiten, ohne dass Sie Ihre Anwendungen ändern müssen.

Eingehende MTOM-Nachrichtenverarbeitung für Pipelines, die Java nicht unterstützen:

Wenn der MTOM-Handler in Pipelines aktiviert ist, die Java nicht unterstützen, überprüft er die Header der eingehenden Nachricht im Container DFHREQUEST oder DFHRESPONSE, um das Format der Nachricht während der Transportverarbeitung zu bestimmen.

Wenn eine MIME Multipart/Related-Nachricht empfangen wird, entpackt der MTOM-Handler die Nachricht wie folgt:

1. Er stellt den Header und die Binärdaten aus den einzelnen binären Anhängen in separate Container.
2. Er stellt die Liste von Containern in den Container DFHWS-XOP-IN.
3. Er stellt das XOP-Dokument, das den Stamm dieser Nachricht bildet, zurück in den Container DFHREQUEST oder DFHRESPONSE und ersetzt so die ursprüngliche Nachricht.

Wenn keine binären Anhänge vorhanden sind, wird das XOP-Dokument als normale XML-Nachricht behandelt, und es ist keine XOP-Verarbeitung erforderlich. Wenn binäre Anhänge vorhanden sind, wird die XOP-Verarbeitung für die Nachricht aktiviert.

Wenn die XOP-Verarbeitung aktiviert ist, überprüft der MTOM-Handler die Pipelineeigenschaften, um festzustellen, ob die aktuelle Nachricht im Direktmodus oder im Kompatibilitätsmodus verarbeitet werden soll, und stellt diese Informationen in den Container DFHWS-MTOM-IN.

Im Providermodus erstellt der MTOM-Handler außerdem den Container DFHWS-MTOM-OUT, um zu bestimmen, wie die ausgehende Antwortnachricht verarbeitet werden soll.

Direktmodus

Wenn Sie CICS-Web-Service-Unterstützung nutzen, d. h. wenn eine Service-Provider-Pipeline den Anwendungshandler DFHPITP verwendet oder eine Service-Requester-Pipeline mithilfe von **INVOKE WEBSERVICE** aufgerufen wird, kann die Pipeline das XOP-Dokument und binäre Anhänge im Direktmodus verarbeiten.

In diesem Modus werden das XOP-Dokument und zugeordnete Container von dem MTOM-Handler an den nächsten Nachrichtenhandler in der Pipeline zur Verarbeitung übergeben. Die CICS-Web-Service-Unterstützung interpretiert die `<xop:Include>`-Elemente. Wenn das base64Binary-Feld als Container in der Anwendungsdatenstruktur dargestellt wird, wird der Anhangscontainername in der Struktur gespeichert. Wenn das Feld als Zeichenfolge mit variabler oder fester Länge dargestellt wird, werden die Inhalte des Containers in das relevante Anwendungsdatenstrukturfeld kopiert. Die Datenstruktur wird dann an das Anwendungsprogramm übergeben.

Kompatibilitätsmodus

Wenn Ihre Pipeline für die Verwendung eines angepassten Anwendungshandlers konfiguriert ist oder wenn Web Services Security ebenfalls aktiviert ist, wird die Nachricht im Kompatibilitätsmodus verarbeitet. In diesem Modus werden das XOP-Dokument und die binären Anhänge sofort mithilfe der XOP-Verarbeitung in einer SOAP-Nachricht neu zusammengestellt, sodass der Inhalt erfolgreich in der Pipeline verarbeitet werden kann. Die XOP-Verarbeitung führt die folgenden Tasks aus:

1. Sie scannt das XOP-Dokument nach `<xop:Include>`-Elementen und ersetzt jedes Vorkommen durch die binären Daten aus dem referenzierten Anhang im base64-codierten Format.
2. Sie verwirft den Container DFHWS-XOP-IN und alle Anhangscontainer.

Die neu zusammengestellte SOAP-Nachricht wird dann an den nächsten Handler in der Pipeline übergeben, um regulär verarbeitet zu werden.

Ist die Web-Service-Validierung aktiviert, wechselt die Pipeline in den Kompatibilitätsmodus, wenn die Nachricht den Anwendungshandler erreicht. Die Nachricht wird in einer SOAP-Nachricht neu zusammengestellt, validiert und an die Anwendung übergeben.

Ausgehende MTOM-Nachrichtenverarbeitung für Pipelines, die Java nicht unterstützen:

Wenn eine Pipeline, die Java nicht unterstützt, für das Senden von ausgehenden Nachrichten konfiguriert ist, werden die Web-Service- und Pipeline-Eigenschaften geprüft, um festzustellen, wie die Nachricht verarbeitet und gesendet werden soll.

Diese Eigenschaften sind in zwei Containern gespeichert: DFHWS-MTOM-OUT und DFHWS-XOP-OUT. In einer Pipeline im Requestermodus werden diese Container von CICS erstellt, wenn die Anwendung den Befehl **EXEC CICS INVOKE WEBSERVICE** ausgibt. In einer Pipeline im Providermodus ist der Container DFHWS-MTOM-OUT bereits mit den Optionen initialisiert, die beim Empfang der eingehenden Nachricht festgelegt wurden.

Wenn die ausgehende Nachricht im Direktmodus verarbeitet werden kann, findet die Optimierung der Nachricht unmittelbar statt. Wenn die ausgehende Nachricht im Kompatibilitätsmodus verarbeitet werden muss, findet die Optimierung ganz am Ende der Pipelineverarbeitung statt.

Wenn Sie Ihre Web-Service-Provider- oder -Requester-Anwendung nicht mithilfe des CICS-Web-Service-Assistenten implementiert haben oder wenn in Ihrer Pipeline die Web-Service-Validierung oder Web Services Security aktiviert ist, wird die ausgehende Nachricht im Kompatibilitätsmodus verarbeitet.

Direktmodus

Im Direktmodus findet die folgende Verarbeitung statt:

1. Ein XOP-Dokument wird aus der Datenstruktur der Anwendung im Container DFHWS-DATA erstellt. Alle binären Felder, die größer-gleich 1500 Bytes sind, werden identifiziert, und die binären Daten und MIME-Header, die den binären Anhang beschreiben, werden in separate Container gestellt. Wenn die binären Daten sich bereits in einem Container befinden, wird dieser Container direkt als

Anhang verwendet. Ein Element `<xop:Include>` wird dann statt der üblichen base64-codierten Binärdaten mithilfe einer generierten Inhalts-ID in die XML eingefügt. Beispiel:

```
<xop:Include href="cid:generierter_wert_der_inhalts-id"
xmlns:xop="http://www.w3.org/2004/08/xop/include">
```

2. Alle Container werden der Anhangsliste im Container DFHWS-XOP-OUT hinzugefügt.
3. Wenn der SOAP-Handler DFHWS-DATA verarbeitet hat, werden das XOP-Dokument und der SOAP-Envelope im Container DFHREQUEST oder DFHRESPONSE gespeichert und über die Pipeline verarbeitet.
4. Wenn der letzte Nachrichtenhandler beendet wurde, packt der MTOM-Handler das XOP-Dokument und die binären Anhänge in eine MIME-Multipart/Related-Nachricht und sendet sie an den Web-Service-Requester oder -Provider. Der Container DFHWS-XOP-OUT und alle zugehörigen Container werden dann gelöscht.

Kompatibilitätsmodus

Wenn die Pipeline das XOP-Dokument nicht direkt verarbeiten kann, findet die folgende Verarbeitung statt:

1. Der SOAP-Hauptteil wird in DFHWS-DATA aus der Anwendungsdatenstruktur erstellt und wie üblich in der Pipeline verarbeitet.
2. Wenn der finale Handler die Verarbeitung der Nachricht beendet hat, prüft der MTOM-Handler die Optionen im Container DFHWS-MTOM-OUT, um zu bestimmen, ob MTOM verwendet werden soll. Optional berücksichtigt er dabei, ob binäre Anhänge vorhanden sind. Wenn der MTOM-Handler feststellt, dass MTOM nicht erforderlich ist, findet keine XOP-Verarbeitung statt, und es wird, wie sonst auch, eine SOAP-Nachricht von CICS gesendet.
3. Wenn der MTOM-Handler feststellt, dass die ausgehende Nachricht im MTOM-Format gesendet werden soll, scannt die XOP-Verarbeitung die Nachricht auf infrage kommende Felder, um die Daten in binäre Anhänge aufzuteilen. Damit ein Feld infrage kommt, muss für das Element das MIME-Attribut **contentType** angegeben sein und der zugeordnete binäre Wert muss aus gültigen base64Binary-Daten im kanonischen Format bestehen. Die Daten müssen größer als 1500 Bytes sein. Die XOP-Verarbeitung erstellt die binären Anhänge und die Anhangsliste und ersetzt dann die Felder durch `<xop:Include>`-Elemente.
4. Der MTOM-Handler packt die XOP-Dokumente und binären Anhänge in eine MIME-Multipart/Related-Nachricht und CICS sendet sie an den Web-Service-Requester oder -Provider.

Einschränkungen bei der Verwendung von MTOM/XOP

Zur Unterstützung von MTOM/XOP können Sie entweder das Element `<mtom>` in Ihrer Pipelinekonfigurationsdatei angeben oder den MTOM-Handler in Ihrer Pipeline aktivieren. Für beide Methoden gibt es jedoch Einschränkungen.

Einschränkungen für Java-basierte Pipelines:

Wenn Sie das Element `<mtom>` in der Pipelinekonfigurationsdatei angeben, wird MTOM/XOP-Unterstützung für Ihre Java-basierte Pipeline aktiviert. Es gibt jedoch Einschränkungen bei dieser MTOM/XOP-Implementierung.

Anwendungshandler DFHPITP

Der Axis2-Modus der MTOM/XOP-Unterstützung kann nicht mit Pipelines verwendet werden, die DFHPITP als Anwendungshandler angeben.

WS-Security

Der Axis2-Modus der MTOM/XOP-Unterstützung kann nicht mit Pipelines verwendet werden, die WS-Security-Konfigurationen verwenden, die XML-Signaturen erfordern.

Befehl **INQUIRE PIPELINE**

Wenn ein Befehl **INQUIRE PIPELINE** für eine Java-basierte Pipeline mithilfe des Axis2-Modus der MTOM/XOP-Unterstützung ausgegeben wird, werden die Attribute **Mtomst**, **Sendmtomst**, **Mtomnoxopst**, **Xopsupportst** und **Xopdirectst** als Nomtom berichtet. Weitere Informationen finden Sie unter **INQUIRE PIPELINE**.

Einschränkungen für andere SOAP-Pipelines:

Durch die Aktivierung des MTOM-Handlers in der Pipeline können Sie Web-Service-Implementierungen mit MTOM/XOP-Optimierung unterstützen. Die Option 'Kompatibilitätsmodus' bedeutet, dass Sie mit diesen Web-Services zusammenarbeiten können, ohne dass Sie Ihre Web-Service-Anwendungen ändern müssen. Es gibt jedoch Situationen, in denen Sie MTOM/XOP nicht verwenden können oder die Verwendung von MTOM/XOP eingeschränkt ist.

CICS-Web-Service-Assistenten verwenden

Die Direktmodus-Optimierung für MTOM/XOP ist nur verfügbar, wenn Sie DFHWS2LS auf Zuordnungsebene 1.2 oder höher verwenden, und wenn das WSDL-Dokument mindestens ein Feld vom Typ 'xsd:base64Binary' enthält. Web-Services, die mit DFHLS2WS aktiviert sind, stehen nicht für eine XOP-Optimierung zur Verfügung.

Web-Services, die mit DFHLS2WS generiert wurden, wobei CHAR-VARYING=BINARY angegeben war, können für MTOM/XOP-Optimierungen infrage kommen. Andere mit DFHLS2WS generierte Web-Services enthalten keine binären Daten und stehen für MTOM/XOP-Optimierungen nicht zur Verfügung, funktionieren aber normal in einer PIPELINE, die MTOM/XOP unterstützt.

Provider-Pipelines

CICS stellt einen Standardanwendungshandler namens DFHPITP zur Verfügung, der in einer Provider-Pipeline konfiguriert werden kann. Dieser Anwendungshandler kann XOP-Dokumente verarbeiten und die erforderlichen Container zur Unterstützung der Pipelineverarbeitung sowohl im Direktmodus als auch im Kompatibilitätsmodus erstellen. Wenn Sie Ihren eigenen Anwendungshandler in einer Provider-Pipeline verwenden und MTOM/XOP aktivieren möchten, sollten Sie die Pipeline für die Ausführung im Kompatibilitätsmodus konfigurieren.

Requester-Pipelines

Wenn Ihre Anwendungen den Befehl **INVOKE WEBSERVICE** verwenden, übernimmt CICS die Optimierung der SOAP-Nachricht für Sie im Direktmodus und im Kompatibilitätsmodus. Wenn Sie das Programm DFHPIRT verwenden, um die Pipeline zu starten, können Sie MIME-Multipart/Related-Nachrichten nur im Kompatibilitätsmodus senden und empfangen.

Web Services Security

Wenn Sie für den MTOM-Handler in der Pipelinekonfigurationsdatei den Direktmodus aktivieren und außerdem den Web Services Security-Nachrichtenhandler aktivieren, unterstützt die Pipeline nur die Verarbeitung von MTOM-Nachrichten im Kompatibilitätsmodus.

Behandlung von binären Daten

Wenn Sie umfangreiche Binärdaten in Ihren Web-Service einschließen müs-

sen, z. B. eine Grafikdatei im JPEG-Format, können Sie MTOM/XOP zur Optimierung der Größe der Nachricht verwenden, die an den Service-Provider oder -Requester gesendet wird. Die Mindestgröße von Binärdaten, die mit MTOM/XOP optimiert werden können, ist 1500 Byte. Wenn die Binärdaten in einem Feld kleiner als 1500 Bytes sind, optimiert CICS das Feld nicht.

Wie in der XOP-Spezifikation angegeben, sollten in den 'base64Binary'-Daten keine Leerräume enthalten sein. Alle Anwendungsprogramme, die 'base64Binary'-Daten erzeugen, müssen das kanonische Format verwenden. Wenn die 'base64Binary'-Daten in einer ausgehenden Nachricht Leerräume enthalten, konvertiert CICS die Daten nicht in einen binären Anhang. Wenn 'base64Binary'-Daten von CICS generiert werden, werden die Felder in einem kanonischen Format bereitgestellt und enthalten deshalb keine Leerräume.

Das Attribut **contentType** muss in 'base64Binary'-Feldern vorhanden sein, damit XOP-Verarbeitung im Kompatibilitätsmodus für ausgehende Nachrichten ausgeführt werden kann. Das Attribut **contentType** darf in 'hexBinary'-Feldern nicht vorhanden sein.

Web-Service-Validierung

Wenn Sie die Web-Service-Validierung aktivieren, findet die folgende Pipeline-Verarbeitung statt:

- Wenn ein eingehendes XOP-Dokument über die Pipeline im Direktmodus übergeben wurde, wechselt CICS automatisch in den Kompatibilitätsmodus und konvertiert es zurück in das Standard-XML-Format, bevor die CICS-Web-Service-Unterstützung das Dokument validiert.
- Eine ausgehende SOAP-Nachricht wird als Standard-XML generiert und im Kompatibilitätsmodus verarbeitet.

Die zusätzliche Pipelineverarbeitung ist erforderlich, da die Validierung den Inhalt von XOP-Dokumenten nicht verarbeiten kann.

CICS für die Unterstützung von MTOM/XOP konvertieren

Damit MTOM-Nachrichten in CICS unterstützt werden, müssen Sie die korrekte MTOM/XOP-Unterstützung für Ihren Typ von Pipeline in Ihren Pipelinekonfigurationsdateien angeben.

MTOM/XOP-Unterstützung für Java-basierte Pipelines konfigurieren:

Zum Konfigurieren von MTOM/XOP-Unterstützung für Java-basierte Pipelines müssen Sie das <mtom>-Element zu Ihren Pipelinekonfigurationsdateien hinzufügen.

Before you begin

Zunächst müssen Sie aber die Pipelinekonfigurationsdateien angeben oder erstellen, in denen Sie Konfigurationsinformationen für MTOM/XOP hinzufügen werden.

About this task

Wenn das <mtom>-Element in Ihrer Pipelinekonfigurationsdatei definiert ist, wird MTOM-Unterstützung für alle eingehenden und ausgehenden Nachrichten aktiviert. Ist dieses Element jedoch nicht in der Pipelinekonfigurationsdatei angegeben, wird MTOM-Unterstützung nur für eingehende Nachrichten aktiviert.

Procedure

Fügen Sie ein `<mtom>`-Element zu Ihrer Pipelinekonfigurationsdatei hinzu. Dieses Element muss nach dem optionalen `<addressing>`-Element und vor dem optionalen `<headerprogram>`-Element definiert werden.

Beispiel

Für eine Pipeline im Provider- oder Requestermodus können Sie Folgendes angeben:

```
<cics_soap_1.2_handler_java>
  <jvmserver>JVMSEVR1</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>
```

MTOM/XOP für andere SOAP-Pipelines konfigurieren:

Zum Konfigurieren von MTOM/XOP-Unterstützung für Pipelines, die nicht die Handler `<cics_soap_1.1_handler_java>` oder `<cics_soap_1.2_handler_java>` verwenden, müssen Sie den MTOM-Handler zu Ihren Pipelinekonfigurationsdateien hinzufügen.

Before you begin

Zunächst müssen Sie aber die Pipelinekonfigurationsdateien angeben oder erstellen, in denen Sie Konfigurationsinformationen für MTOM/XOP hinzufügen werden.

Procedure

1. Fügen Sie ein `<cics_mtom_handler>`-Element zu Ihrer Pipelinekonfigurationsdatei hinzu. Dieses Element sollte das erste Element im `<provider_pipeline>`-Element und das letzte Element vor dem `<service_parameter_list>`-Element im `<requester_pipeline>`-Element sein. Codieren Sie die folgenden Elemente:

```
<cics_mtom_handler>
  <dfhmtom_configuration version="1">
  </dfhmtom_configuration>
</cics_mtom_handler>
```

Das `<dfhmtom_configuration>`-Element ist ein Container für die anderen Elemente in der Konfiguration. Wenn Sie die Standardeinstellungen für die MTOM/XOP-Verarbeitung akzeptieren, können Sie ein leeres Element wie folgt angeben.

```
<cics_mtom_handler/>
```

2. Optional: Codieren Sie ein `<mtom_options>`-Element. Sowohl in einer Service-Provider- als auch in einer Service-Requester-Pipeline gibt dieses Element an, ob die ausgehende Nachricht als MTOM-Nachricht gepackt werden soll.
 - a. Codieren Sie das Attribut **send_mtom**, um zu definieren, ob die ausgehende Nachricht als MTOM-Nachricht gesendet werden soll. Ausführliche Informationen zu diesem Attribut finden Sie unter „Pipelinekonfigurationselement `<mtom_options>`“ auf Seite 132.

- b. Codieren Sie das Attribut **send_when_no_xop**, um zu definieren, ob die ausgehende Nachricht als MTOM-Nachricht gesendet werden soll, wenn keine binären Anhänge vorhanden sind. Ausführliche Informationen zu diesem Attribut finden Sie unter „Pipelinekonfigurationselement <mtom_options>“ auf Seite 132.
3. Optional: Codieren Sie ein <xop_options>-Element mit einem Attribut **apphandler_supports_xop**. Dies gibt an, ob der Anwendungshandler XOP-Dokumente direkt verarbeiten kann. Wenn Sie dieses Attribut nicht angeben, hängt die Standardeinstellung davon ab, ob das Element <apphandler> DFHPITP oder ein anderes Programm angibt. Ausführliche Informationen zu diesem Attribut finden Sie unter „Pipelinekonfigurationselement <xop_options>“ auf Seite 133.
4. Optional: Codieren Sie ein <mime_options>-Element mit einem Attribut **content_id_domain**. Dies gibt den Domännennamen an, der verwendet werden soll, wenn MIME-Werte vom Typ 'content-ID' generiert werden, mit denen binäre Anhänge angegeben werden. Ausführliche Informationen zu diesem Attribut finden Sie unter „Pipelinekonfigurationselement <mime_options>“ auf Seite 135.

Beispiel

Das folgende Beispiel zeigt ein vollständiges <cics_mtom_handler>-Element an, in dem alle optionalen Elemente vorhanden sind.

```
<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>
```

Unterstützung für Web-Service-Adressierung

CICS unterstützt Services, die die Web-Service-Adressierungsspezifikationen (WS-Addressing) des Worldwide Web Consortium (W3C) verwenden. Diese Familie von Spezifikationen stellt transportunabhängige Mechanismen zur Adressierung von Web-Services bereit und ermöglicht eine End-to-End-Adressierung.

CICS stellt sicher, dass Ihre vorhandenen Web-Service-Anwendungen Anforderungen von Web-Services akzeptieren können, die WS-Addressing verwenden. Sie können auch neue Web-Services erstellen, die Endpunktreferenzen und Nachrichtenadressierungseigenschaften in SOAP-Nachrichten verwenden.

WS-Addressing fügt Adressierungsinformationen in Form von Nachrichtenadressierungseigenschaften (MAPs) zu SOAP-Nachrichtenheadern hinzu. MAPs enthalten Informationen zur Nachrichtenübertragung wie eine eindeutige Nachrichten-ID und Endpunktreferenzen, die detailliert angeben, woher die Nachricht stammt, wohin die Nachricht geht und wohin Antwort- oder Fehlnachrichten gesendet werden sollen. Eine Endpunktreferenz (EPR) ist ein spezifischer Typ von MAP, der die Zieladresse der Nachricht, optionale Referenzparameter zur Verwendung durch die Anwendung und optionale Metadaten enthält.

Funktionen der WS-Addressing-Unterstützung

CICS umfasst die folgenden Funktionen zur Unterstützung von WS-Addressing:

- Ihre Web-Service-Requester- und -Provider-Anwendungen können mit anderen Services interagieren, die WS-Addressing verwenden, ohne dass Sie diese erneut implementieren müssen. Ein neuer Nachrichtenhandler in der Pipeline, der Adressierungsnachrichtenhandler DFHWSADH, leitet Nachrichten, die WS-Addressing-Informationen enthalten, an den angegebenen Web-Service weiter.
- Sie können eine Anwendung schreiben, die WS-Addressing-API-Befehle verwendet, um eine Endpunktreferenz zu erstellen und um einen Adressierungskontext zu erstellen, zu aktualisieren, zu löschen und abzufragen.
- Sie können Antwortnachrichten an andere Endpunkte als den Requester-Endpunkt weiterleiten. Beispielsweise können Sie Fehlermeldungen an einen dedizierten Fehlerhandler weiterleiten.
- Sie können Referenzparameter als Teil der MAPs im SOAP-Header an Anwendungen übergeben.

Unterstützung für WS-Addressing-Spezifikationen und Interoperabilität

Standardmäßig unterstützt CICS die Empfehlungsspezifikationen:

- W3C WS-Addressing 1.0 - Core
- W3C WS-Addressing 1.0 - SOAP Binding
- W3C WS-Addressing 1.0 - Metadata

Diese Spezifikationen sind durch den Namensbereich <http://www.w3.org/2005/08/addressing> angegeben. Sofern nicht anders festgehalten, bezieht sich WS-Addressing-Semantik, die in dieser Dokumentation beschrieben ist, auf die Empfehlungsspezifikationen.

Für die Interoperabilität unterstützt CICS auch die Übermittlungsspezifikation:

- W3C WS-Addressing- Submission

Diese Spezifikation ist durch den Namensbereich <http://schemas.xmlsoap.org/ws/2004/08/addressing> angegeben. Verwenden Sie die Übermittlungsspezifikation nur, wenn Sie mit einem Client oder einem Web-Service-Provider interagieren müssen, der die Übermittlungsspezifikation implementiert.

Übersicht über die Web-Service-Adressierung

Die Web-Service-Adressierung (WS-Addressing) stellt ein Standardframework für die Angabe der Endpunkte einer SOAP-Nachricht bereit. Dieses Framework ist transportneutral und verbessert die Interoperabilität von Web-Services, die unterschiedliche Transportmechanismen verwenden. Die WS-Addressing-Spezifikation führt Nachrichtenadressierungseigenschaften und Endpunktreferenzen ein.

Die Web-Service-Adressierung (WS-Addressing) ist eine Spezifikation des Worldwide Web Consortium (W3C), die die Interoperabilität zwischen Web-Services verbessert, indem sie eine Standardvorgehensweise zur Adressierung von Web-Services und zur Implementierung von Adressierungsinformationen in SOAP-Nachrichten definiert. SOAP-Nachrichten können über eine Vielzahl von Transportmechanismen gesendet werden, darunter HTTP und WebSphere MQ, die Zielinformationen für die Nachricht auf unterschiedliche Weise speichern.

Vorhandene CICS-Web-Services, die in einer Pipeline implementiert sind, die für die Verwendung von WS-Addressing konfiguriert ist, können die WS-Addressing-Standard Einstellungen verwenden, ohne dass Änderungen erforderlich sind. Mit-

hilfe der WS-Addressing-API-Befehle können Sie die WS-Addressing-Funktionalität uneingeschränkt nutzen. Die WS-Addressing-Implementierung unterstützt einen SOAP-Fehler pro WSDL-Operation.

Nachrichtenadressierungseigenschaften

Nachrichtenadressierungseigenschaften (Message Addressing Properties, MAPs) sind ein Satz wohldefinierter WS-Addressing-Eigenschaften, die als Elemente in SOAP-Headern dargestellt werden können. MAPs stellen eine Standardvorgehensweise für das Übertragen von Informationen bereit, z. B. den Endpunkt, an den Nachrichtenantworten weitergeleitet werden müssen, oder Informationen zur Beziehung, in der die Nachricht zu anderen Nachrichten steht. Die von der WS-Addressing-Spezifikation definierten MAPs sind in der folgenden Tabelle zusammengefasst.

Tabelle 11. Von der WS-Addressing-Spezifikation definierte Nachrichtenadressierungseigenschaften

Abstrakter Name der WS-Addressing-MAP	SOAP-Name der WS-Addressing-MAP	MAP-Inhaltstyp	Verbindungsart	Beschreibung
[action]	<wsa:Action>	xs:anyURI	1..1	Ein absoluter URI, der die Semantik der Nachricht eindeutig angibt. Dieser Wert ist erforderlich.
[destination]	<wsa:To>	xs:anyURI in der SOAP-Nachricht EndpointReference im Adressierungskontext	0..1	Der absolute URI, der die Adresse des beabsichtigten Empfängers der Nachricht angibt. Wenn dieser Wert nicht angegeben ist, wird standardmäßig der anonyme URI verwendet, der in der Spezifikation definiert ist: http://www.w3.org/2005/08/addressing/anonymous . Im Adressierungskontext wird die MAP <wsa:To> als EPR dargestellt. Wenn <wsa:To> als Teil einer SOAP-Nachricht gesendet wird, wird es in die Adresse und die zugehörigen Referenzparameter aufgeteilt, wie durch das Schema definiert.
[reference parameters] *	[reference parameters]*	xs:any	0..unbegrenzt	Parameter, die den <wsa:ReferenceParameters>-Eigenschaften der Endpunktreferenz entsprechen, an die die Nachricht adressiert ist. Dieser Wert ist optional.
[source endpoint]	<wsa:From>	EndpointReference	0..1	Eine Referenz auf den Endpunkt, von dem die Nachricht stammt. Dieser Wert ist optional.
[reply endpoint]	<wsa:ReplyTo>	EndpointReference	0..1	Eine Endpunktreferenz für den geplanten Empfänger von Antworten auf diese Nachricht. Dieser Wert ist optional. Wenn dieser Wert nicht angegeben ist, wird standardmäßig http://www.w3.org/2005/08/addressing/anonymous verwendet.
[fault endpoint]	<wsa:FaultTo>	EndpointReference	0..1	Eine Endpunktreferenz für den geplanten Empfänger von Fehlern im Zusammenhang mit dieser Nachricht. Dieser Wert ist optional und nimmt standardmäßig den Wert der MAP <wsa:ReplyTo> an.

Tabelle 11. Von der WS-Addressing-Spezifikation definierte Nachrichtenadressierungseigenschaften (Forts.)

Abstrakter Name der WS-Addressing-MAP	SOAP-Name der WS-Addressing-MAP	MAP-Inhaltstyp	Verbindungsart	Beschreibung
[relationship] *	<wsa:RelatesTo>	xs:anyURI plus optionales Attribut vom Typ 'xs:anyURI'	0..unbegrenzt	Ein Wertepaar, das angibt, in welcher Beziehung diese Nachricht zu anderen Nachrichten steht. Der Inhalt dieses Elements übermittelt die <wsa:MessageID> der verwandten Nachricht. Ein optionales Attribut übermittelt den Beziehungstyp. Dieser Wert ist optional. Wenn dieser Wert nicht angegeben ist, wird standardmäßig http://www.w3.org/2005/08/addressing/reply verwendet.
[message id]	<wsa:MessageID>	xs:anyURI		Ein absoluter URI, der die Nachricht eindeutig angibt. Dieser Wert ist optional. Wenn keine Angabe gemacht wird, generiert CICS einen Wert für ausgehende Anforderungen und Antworten.

Das folgende Beispiel einer SOAP-Nachricht enthält WS-Addressing-MAPs:

```
<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>
```

Endpunktreferenzen

Eine Endpunktreferenz ist ein bestimmter Typ von MAP, der einen Standardmechanismus zur Kapselung von Informationen über bestimmte Endpunkte bereitstellt. Endpunktreferenzen können an andere Parteien gesendet werden und geben den jeweiligen Web-Service-Endpunkt an. In der folgenden Tabelle ist das Informationsmodell für Endpunktreferenzen zusammengefasst.

Tabelle 12. Informationsmodell für Endpunktreferenzen

Abstrakter Eigenschaftsname	Eigenschaftstyp	Verbindungsart	Beschreibung
[address]	xs:anyURI	1..1	Der absolute URI, der die Adresse des Endpunkts angibt.

Tabelle 12. Informationsmodell für Endpunktreferenzen (Forts.)

Abstrakter Eigenschaftsname	Eigenschaftstyp	Verbindungsart	Beschreibung
[reference parameters] *	xs:any	0..unbegrenzt	Informationen zu durch den Namensbereich qualifizierten Elementen, die für die Kommunikation mit dem Endpunkt erforderlich sind.
[metadata]	xs:any	0..unbegrenzt	Beschreibung des Verhaltens, der Richtlinien und der Funktionalität des Endpunkts.

Das folgende XML-Fragment stellt eine Endpunktreferenz dar. Das Element `<wsa:EndpointReference>` referenziert den Endpunkt an dem URI `http://example.ibm.com/enquiry` und enthält Metadaten, die die Schnittstelle angeben, auf die die Endpunktreferenz zeigt, und einige anwendungsspezifische Referenzparameter.

```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
  xmlns:example="http://example.ibm.com/namespace">
  <wsa:Address>http://example.ibm.com/enquiry</wsa:Address>
  <wsa:Metadata
    xmlns:wsdl="http://www.w3.org/ns/wsdl-instance"
    wsdl:wsdlLocation="http://example.ibm.com/wsdl/wsdl-location.wsdl">
    <wsam:InterfaceName>example:reservationInterface</wsam:InterfaceName>
  </wsa:Metadata>
  <wsa:ReferenceParameters>
    <example:AccountCode>123456789</example:AccountCode>
    <example:DiscountId>ABCDEFGH</example:DiscountId>
  </wsa:ReferenceParameters>
</wsa:EndpointReference>
```

WS-Addressing-MAPs vom Typ `wsa:EndpointReferenceType` sind: `<wsa:From>`, `<wsa:ReplyTo>` und `<wsa:FaultTo>`. Allerdings ist die `<wsa:To>`-MAP im dem WS-Addressing 1.0-Standard als Typ `xs:anyURI` definiert. Der Einfachheit halber behandelt CICS `<wsa:To>`-MAPs im Adressierungskontext als EPRs. Wenn eine `<wsa:To>`-MAP als Teil einer SOAP-Nachricht gesendet wird, teilt CICS sie in die Adresse und die Referenzparameter auf, wie vom Standard vorgegeben.

Standardnamensbereiche

Auf das folgende Präfix und die entsprechenden Namensbereiche wird in der WS-Addressing-Dokumentation Bezug genommen:

Tabelle 13. Präfix und entsprechender Namensbereich

Standardpräfix	Namensbereich
xs	<code>http://www.w3.org/2001/XMLSchema</code>
wsa	<code>http://www.w3.org/2005/08/addressing</code> (Empfehlungsschema) <code>http://schemas.xmlsoap.org/ws/2004/08/addressing</code> (Übermittlungsschema)
wsam	<code>http://www.w3.org/2007/05/addressing/metadata</code>

Requester-Pipeline für die Web-Service-Adressierung konfigurieren

Um eine Requester-Pipeline für die Unterstützung von Web-Service-Adressierung (WS-Addressing) zu konfigurieren, müssen Sie einen Adressierungshandler zu Ihrer Pipelinekonfigurationsdatei hinzufügen.

Before you begin

Sie müssen die Pipelinekonfigurationsdatei identifizieren oder erstellen, um die Konfigurationsinformationen für WS-Addressing hinzuzufügen. Und Sie müssen entscheiden, welche WS-Addressing-Spezifikationen verwendet werden sollen. Verwenden Sie die Spezifikation *W3C WS-Addressing 1.0 Core*, wenn möglich.

About this task

Sie haben zwei Möglichkeiten, Unterstützung für WS-Addressing hinzuzufügen:

- Wenn die SOAP-Pipeline Java verwendet, wird die SOAP-Adressierung von Axis2 übernommen, und Sie können die von dieser Technologie bereitgestellte Unterstützung verwenden, um Anforderungen zu verarbeiten, die WS-Addressing nutzen. Die gesamte Headerverarbeitung wird von Axis2 verarbeitet und es ist wichtig, dass Sie nicht das Programm zur Headerverarbeitung DFHWSADH zur Pipeline hinzufügen. Sie können Ihre eigenen Programme zur Headerverarbeitung verwenden. Um eine bessere Leistung zu erzielen, schreiben Sie Axis2-Handler in Java, wenn Sie SOAP-Header verarbeiten wollen.
- Wenn die SOAP-Pipeline Java nicht verwendet, müssen Sie das von CICS bereitgestellte Programm zur Headerverarbeitung DFHWSADH hinzufügen, um die Anforderungen zu verarbeiten.

Procedure

- Wenn die SOAP-Pipeline ein Element `<cics_soap_1.1_handler_java>` oder `<cics_soap_1.2_handler_java>` verwendet, fügen Sie ein Element `<addressing>` zur Pipelinekonfigurationsdatei hinzu. Schließen Sie ein Element `<namespace>` ein, das die Spezifikation enthält, die Sie für die Anforderungsnachricht verwenden möchten. Diese Spezifikation kann sich von der Antwortnachricht unterscheiden. Sie können beispielsweise immer eine Anforderungsnachricht senden, die mit der W3C-Core-Spezifikation kompatibel ist, auch wenn die Antwortnachricht die Übermittlungsspezifikation verwendet. Axis2 unterstützt beide WS-Addressing-Spezifikationen für eingehende Nachrichten.

Das folgende Beispiel stellt dar, wie Sie die Requester-Pipeline konfigurieren können:

```
<requester_pipeline>
  <service>
    <service_handler_list>
      <cics_soap_1.1_handler_java>
        <jvmserver>JVMSERV1</jvmserver>
        <addressing>
          <namespace>http://www.w3.org/2005/08/addressing</namespace>
        </addressing>
      </cics_soap_1.1_handler_java>
    </service_handler_list>
  </service>
</requester_pipeline>
```

Das Element `<jvmserver>` enthält den Namen der JVMSERVER-Ressource, die Axis2 unterstützt.

- Wenn die SOAP-Pipeline Java nicht unterstützt, fügen Sie ein Headerprogramm für CICS-Adressierung in `<cics_soap_1.1_handler>` oder `<cics_soap_1.2_handler>` zur Pipelinekonfigurationsdatei hinzu. Das folgende Beispiel stellt dar, wie Sie die Requester-Pipeline konfigurieren können:

```
<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>
```

Codieren Sie die Elemente `<program_name>`, `<localname>` und `<mandatory>` genau wie beschrieben. Setzen Sie `<namespace>` auf `http://www.w3.org/2005/08/addressing`, um die Spezifikation *W3C WS-Addressing 1.0 Core* zu verwenden, oder auf `http://schemas.xmlsoap.org/ws/2004/08/addressing`, um die Spezifikation *W3C WS-Addressing Submission* zu verwenden.

Die Reihenfolge von Programmen zur Headerverarbeitung ist nicht garantiert. Wenn Sie andere Programme zur Headerverarbeitung definieren, fügen Sie sie in einem nachfolgenden CICS-SOAP-Handlerelement in Ihrem Element `<service_handler_list>` hinzu. Der Header-Handler DFHWSADH muss in dem ersten SOAP-Handlerelement vorhanden sein.

Results

Ihre Requester-Pipeline ist jetzt für die Unterstützung von WS-Addressing konfiguriert.

What to do next

Erstellen Sie eine PIPELINE-Ressource, die auf die Konfigurationsdatei verweist. Wenn Sie eine Java-basierte SOAP-Pipeline verwenden, stellen Sie sicher, dass eine JVMSERVER-Ressource für die Axis2-Verarbeitung aktiviert ist.

Provider-Pipeline für die Web-Service-Adressierung konfigurieren

Um eine Provider-Pipeline für die Unterstützung von Web-Service-Adressierung (WS-Addressing) zu konfigurieren, müssen Sie einen Adressierungshandler zu Ihrer Pipelinekonfigurationsdatei hinzufügen.

Before you begin

Sie müssen die Pipelinekonfigurationsdatei identifizieren oder erstellen, um die Konfigurationsinformationen für WS-Addressing hinzuzufügen. Und Sie müssen entscheiden, welche WS-Addressing-Spezifikationen verwendet werden sollen. Verwenden Sie die Spezifikation *W3C WS-Addressing 1.0 Core*, wenn möglich.

About this task

Sie haben zwei Möglichkeiten, Unterstützung für WS-Addressing hinzuzufügen:

- Wenn die SOAP-Pipeline Java verwendet, wird die SOAP-Adressierung von Axis2 übernommen, und Sie können die von dieser Technologie bereitgestellte

Unterstützung verwenden, um Anforderungen zu verarbeiten, die WS-Addressing nutzen. Die gesamte Headerverarbeitung wird von Axis2 verarbeitet und es ist wichtig, dass Sie nicht das Programm zur Headerverarbeitung DFHWSADH zur Pipeline hinzufügen. Sie können Ihre eigenen Programme zur Headerverarbeitung verwenden. Um eine bessere Leistung zu erzielen, schreiben Sie Axis2-Handler in Java, wenn Sie SOAP-Header verarbeiten wollen.

- Wenn die SOAP-Pipeline Java nicht verwendet, müssen Sie das von CICS bereitgestellte Programm zur Headerverarbeitung DFHWSADH hinzufügen, um die Anforderungen zu verarbeiten.

Procedure

- Wenn die SOAP-Pipeline ein Element `<cics_soap_1.1_handler_java>` oder `<cics_soap_1.2_handler_java>` verwendet, fügen Sie ein Element `<addressing>` zur Pipelinekonfigurationsdatei hinzu. Sie können optional ein oder mehrere `<namespace>`-Elemente einschließen. Dieses Element enthält die Spezifikation, die Sie für die ausgehende Nachricht verwenden möchten. Diese Spezifikation kann sich von der eingehenden Nachricht unterscheiden. Sie können beispielsweise immer eine ausgehende Nachricht senden, die mit der W3C-Core-Spezifikation kompatibel ist, auch wenn die eingehende Nachricht die Übermittlungsspezifikation verwendet. Wenn Sie dieses Element ausschließen, verwendet Axis2 dieselbe Spezifikation für die ausgehende und die eingehende Nachricht. Axis2 unterstützt beide WS-Addressing-Spezifikationen für eingehende Nachrichten.

Das folgende Beispiel stellt dar, wie Sie die Provider-Pipeline konfigurieren können:

```
<provider_pipeline>
  <terminal_handler>
    <cics_soap_1.1_handler_java>
      <jvmserver>JVMSESV1</jvmserver>
      <addressing>
        <namespace>http://www.w3.org/2005/08/addressing</namespace>
      </addressing>
    </cics_soap_1.1_handler_java>
  </terminal_handler>
</provider_pipeline>
```

Das Element `<jvmserver>` enthält den Namen der JVMSERVER-Ressource, die Axis2 unterstützt.

- Wenn die SOAP-Pipeline Java nicht verwendet, fügen Sie das Headerprogramm für CICS-Adressierung DFHWSADH zum SOAP-Handler in der Pipelinekonfigurationsdatei hinzu. Das folgende Beispiel stellt dar, wie Sie die Provider-Pipeline konfigurieren können:

```
<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>
```

Codieren Sie die Elemente `<program_name>`, `<localname>` und `<mandatory>` genau wie beschrieben. Setzen Sie `<namespace>` auf `http://www.w3.org/2005/08/addressing`, um die Spezifikation *W3C WS-Addressing 1.0 Core* zu verwenden,

oder auf <http://schemas.xmlsoap.org/ws/2004/08/addressing>, um die Spezifikation *W3C WS-Addressing Submission* zu verwenden.

Die Reihenfolge von Programmen zur Headerverarbeitung ist nicht garantiert. Wenn Sie andere Programme zur Headerverarbeitung definieren, fügen Sie sie in einem anderen CICS-SOAP-Handlerelement in Ihrem Element `<service_handler_list>` hinzu. Der Header-Handler DFHWSADH muss in dem letzten SOAP-Handlerelement vorhanden sein.

Results

Ihre Provider-Pipeline ist jetzt für die Unterstützung von WS-Addressing konfiguriert.

What to do next

Erstellen Sie eine PIPELINE-Ressource, die auf die Konfigurationsdatei verweist. Wenn Sie eine Java-basierte SOAP-Pipeline verwenden, stellen Sie sicher, dass eine JVMSERVER-Ressource für die Axis2-Verarbeitung aktiviert ist.

Web-Service erstellen, der WS-Addressing verwendet

Um einen Web-Service aus einem WSDL-Dokument zu erstellen, das Web-Service-Adressierung (WS-Addressing) einsetzt, verwenden Sie Parameter im Web-Service-Assistenten, um die Konvertierung von XML in Sprachstrukturen zu handhaben.

About this task

Sie können den Job des Web-Service-Assistenten, DFHWS2LS, verwenden, um zu steuern, wie eine Endpunktreferenz (EPR) im WSDL-Dokument gehandhabt wird, und um zu bestimmen, ob CICS Standardeingabe-, -ausgabe- und -fehleraktionen erstellt.

Procedure

1. Legen Sie den Parameter **MINIMUM-RUNTIME** im Web-Service-Assistenten DFHWS2LS auf 3.0 oder höher fest. Eine Laufzeitebene von 3.0 oder höher stellt sicher, dass alle generierten Web-Service-Bindungen WS-Addressing vollständig unterstützen und mit anderen Web-Service-Plattformen interagieren.
2. Legen Sie den Parameter **MAPPING-LEVEL** im Web-Service-Assistenten DFHWS2LS auf 3.0 oder höher fest.
3. Legen Sie den Parameter **WSADDR-EPR-ANY** auf TRUE fest, wenn Sie Elemente vom Typ `wsa:EndpointReferenceType` in den Anforderungs- oder Antwortnachrichten verwenden möchten. Endpunktreferenzen können in Anwendungsdaten eingeschlossen werden und Sie haben die Möglichkeit, die EPR in API-Befehlen wie **WSACONTEXT BUILD** zu verwenden. Indem Sie den Parameter **WSADDR-EPR-ANY** auf TRUE festlegen, geben Sie an, dass CICS die EPR-Daten zur Laufzeit nicht in eine Struktur umsetzen darf. Stattdessen behandelt CICS die EPR-Daten als `<xsd:any>`-Element und speichert sie in einem benannten Container.

Dieses WSDL-Beispielfragment zeigt an, wie eine `<wsa:To>`-Zuordnung als ein Element vom Typ `wsa:EndpointReferenceType` weitergegeben wird:

```
<?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>

```

Wenn das Element '`<xs:element name="myEpr" type="wsa:EndpointReferenceType"/>`' **1** von DFHWS2LS verarbeitet wird, wobei der Parameter **WSADDR-EPR-ANY** auf TRUE gesetzt ist, werden die myEpr-Elementdaten in einem benannten Container als `<xsd:any>`-Element gespeichert, und ein Verweis auf den Container wird zur generierten Sprachstruktur hinzugefügt.

Im folgenden Beispiel sehen Sie die von DFHWS2LS generierte COBOL-Sprachstruktur für das Element myEpr:

```

09 myEpr.
   12 myEpr-xml-cont          PIC X(16).
   12 myEpr-xmlns-cont       PIC X(16).

```

Der Container myEpr-xml-cont speichert den Namen des Containers, der die myEpr-Daten enthält. myEpr-xmlns-cont ist ein optionaler Container, der mit allen XML-Namensbereichsdeklarationen in diesem Bereich gefüllt wird.

4. Speichern und übergeben Sie den Job DFHWS2LS.

Results

CICS erstellt eine Web-Service-Bindung, um die Datenumsetzung abzuwickeln, und Sprachstrukturen, die Sie zum Erstellen der Service-Requester- oder -Provider-Anwendung verwenden können.

What to do next

Aktivieren Sie den Web-Service, indem Sie einen Pipeline-Scan ausführen, um die erforderlichen CICS-Ressourcen zu erstellen.

Standardendpunktreferenzen:

Die meisten WSDL-Dokumente enthalten die Adresse, unter der der Web-Service gehostet wird. In WS-Addressing kann das WSDL-Dokument auch eine Endpunktreferenz für den Web-Service enthalten. Diese Endpunktreferenz kann zusätzliche Metadaten enthalten, die die Kommunikation zwischen Requester- und Provider-Anwendungen erleichtern.

Wenn Sie DFHWS2LS verwenden, um die WSDL zu verarbeiten, wird die Endpunktreferenz (EPR) in der Web-Service-Bindung gespeichert und von CICS zum Senden von Anforderungs- und Antwortnachrichten verwendet. Alle Referenzpara-

meter, <wsa:ReferenceParameters>, die in der EPR definiert sind, werden in die SOAP-Nachricht eingeschlossen. Diese EPR wird als Standard-EPR bezeichnet, weil sie von einer Anwendung überschrieben werden kann. Wenn die Anwendung keine explizite Endpunktreferenz bereitstellt, wird die Standard-EPR aus der WSDL verwendet.

Das folgende WSDL 1.1-Fragment enthält eine Standard-EPR: <soap:address location="http://example.ibm.com:12345/exampleTest" />.. Das Element <port> enthält ein untergeordnetes Element, <wsa:EndpointReference>, die von dem untergeordneten Element **2** angegebene Adresse muss mit der im übergeordneten Element **1** angegebenen Adresse übereinstimmen:

```
<service name="exampleService">
  <port name="examplePort" binding="s0:createBinding">
    <soap:address location="http://example.ibm.com:12345/exampleTest" /> 1

    <wsa:EndpointReference
      xmlns:example="http://example.ibm.com/namespace"
      xmlns:wsdl="http://www.w3.org/2006/01/wsdl-instance"
      wsdl:wsdlLocation="http://example.ibm.com/location "
      title="http://example.ibm.com/example/example_wsdl"
      class="http://example.ibm.com/example/example.wsdl">
        <wsa:Address>http://example.ibm.com:12345/exampleTest</wsa:Address> 2
        <wsa:Metadata>
          <wsam:InterfaceName>example:Inventory</wsam:InterfaceName>
        </wsa:Metadata>
        <wsa:ReferenceParameters>
          <example:AccountCode>123456789</example:AccountCode>
          <example:DiscountId>ABCDEFGH</example:DiscountId>
        </wsa:ReferenceParameters>
      </wsa:EndpointReference>
    </port>
  </service>
```

Explizite Aktionen:

WSDL-Dokumente können die Werte der <wsa:Action>-Eigenschaften explizit definieren. Wenn das WSDL-Dokument keine explizit definierten <wsa:Action>-Eigenschaften enthält, erstellt CICS Standardaktionen, wenn die WSDL von DFHWS2LS verarbeitet wird.

WSDL 1.1

Das folgende WSDL 1.1-Fragment stellt ein Buchungssystem dar, das explizit definierte <wsa:Action>-Eigenschaften enthält:

```
<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>
```

In diesem Beispiel ist die Eingabeaktion der Operation makeBooking explizit als http://example.ibm.com/namespace/makeBooking definiert, und die Ausgabeaktion ist explizit als http://example.ibm.com/namespace/bookingResponse definiert.

WSDL 2.0

Das folgende WSDL 2.0-Fragment stellt ein Buchungssystem dar, das explizit definierte `<wsa:Action>`-Eigenschaften enthält:

```
<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>
```

In diesem Beispiel ist die Eingabeaktion der Operation `makeBooking` explizit als `http://example.ibm.com/namespace/makeBooking` definiert, und die Ausgabeaktion ist als `http://example.ibm.com/namespace/makeBookingResponse` definiert.

Weitere Informationen finden Sie in der Spezifikation W3C WS-Addressing 1.0 Metadata.

Standardaktionen für WSDL 1.1:

Wenn ein WSDL 1.1-Dokument nicht explizit angegebene `<wsa:Action>`-Eigenschaften enthält, erstellt CICS Standardeingabe-, -ausgabe- und -fehleraktionen, wenn die WSDL von DFHWS2LS verarbeitet wird.

Standardeingabe- und -ausgabeaktionen für WSDL 1.1

Das folgende Muster wird von CICS in WSDL 1.1-Dokumenten verwendet, die entweder dem Empfehlungsschema oder dem Übergabeschema folgen, um eine Standardeingabe- oder -ausgabeaktion zu erstellen:

```
[zielnamensbereich]/[name_des_porttyps]/[name_von_eingabe|ausgabe]
```

Standardfehleraktionen für WSDL 1.1

Wenn Sie dem Empfehlungsschema folgen, unterscheidet sich die Art, wie CICS die Standardfehleraktion erstellt, von dem im Schema beschriebenen Verhalten. Das folgende Muster wird von CICS in WSDL 1.1-Dokumenten verwendet, die dem Empfehlungsschema folgen, um eine Standardfehlernachricht zu erstellen. Beachten Sie, dass der Fehlername weggelassen wird.

```
[zielnamensbereich]/[name_des_porttyps]/[operationsname]/Fault/
```

Wenn Sie dem Übergabeschema folgen, entspricht die Art, wie CICS die Standardfehleraktion erstellt, dem im Schema beschriebenen Verhalten. Das folgende Muster wird von CICS in WSDL 1.1-Dokumenten verwendet, die dem Übergabeschema folgen, um eine Standardfehlernachricht zu erstellen:

```
[zielnamensbereich]/[name_des_porttyps]/[operationsname]/Fault/[fehlername]
```

Beispiel für die Standardaktionen, die von CICS für ein WSDL 1.1-Dokument generiert werden

Dieses Beispiel eines Buchungssystems veranschaulicht, wie CICS Standardaktionen aus einem WSDL 1.1-Dokument erstellt:

```

<beschreibung_zielnamensbereich="http://example.ibm.com/namespace" ...>
...
<name_des_porttyps="bookingInterface">
  <operationsname="makeBooking">
    <eingabeelement="tns:makeBooking" name="MakeBooking"/>
    <ausgabeelement="tns:bookingResponse" name="BookingResponse"/>
    <fehlernachricht="tns:InvalidBooking" name="InvalidBooking"/>
  </operation>
</interface>
...
</definitions>

```

Das WSDL-Fragment hat die folgenden Adressierungseigenschaften:

Eigenschaftsname	Wert
[zielnamensbereich]	http://example.ibm.com/namespace
[name_des porttyps]	bookingInterface
[operationsname]	makeBooking
[eingabename]	MakeBooking
[ausgabename]	BookingResponse
[fehlername]	InvalidBooking

Die folgenden Aktionen werden aus diesen Werten erstellt:

Aktion	Wert
Eingabeaktion	<p>http://example.ibm.com/namespace/bookingInterface/MakeBooking</p> <p>Wenn der [eingabename] nicht angegeben ist, wird stattdessen der Wert des [operationsnamens] mit angehängtem "Request" verwendet. In diesem Beispiel ist die Eingabeaktion http://example.ibm.com/namespace/bookingInterface/makeBookingRequest.</p> <p>http://example.ibm.com/namespace/bookingInterface/BookingResponse</p>
Ausgabeaktion	<p>Wenn der [ausgabename] nicht angegeben ist, wird stattdessen der Wert des [operationsnamens] mit angehängtem "Response" verwendet. In diesem Beispiel ist die Ausgabeaktion http://example.ibm.com/namespace/bookingInterface/makeBookingResponse.</p>
Fehleraktion (Empfehlungsschema)	<p>http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/</p> <p>Beachten Sie, dass der [fehlername] weggelassen wird.</p>
Fehleraktion (Übergabeschema)	<p>http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/InvalidBooking</p>

Weitere Informationen finden Sie in der Spezifikation W3C WS-Addressing 1.0 Metadata.

Standardaktionen für WSDL 2.0:

Wenn ein WSDL 2.0-Dokument nicht explizit angegebene <wsa:Action>-Eigenschaften enthält, erstellt CICS Standardeingabe-, -ausgabe- und -fehleraktionen, wenn die WSDL von DFHWS2LS verarbeitet wird.

Standardeingabe- und -ausgabeaktionen für WSDL 2.0

Das folgende Muster wird von CICS in WSDL 2.0-Dokumenten verwendet, die dem Empfehlungsschema folgen, um Standardaktionen für Eingabe und Ausgabe zu erstellen:

[zielnamensbereich]/[schnittstellename]/[operationsname] [anweisungstoken]

Standardfehleraktionen für WSDL 2.0

Wenn Sie dem Empfehlungsschema folgen, unterscheidet sich die Art, wie CICS die Standardaktion für WS-Addressing erstellt, von dem im Schema beschriebenen Verhalten. Wenn Sie dem Übergabeschema folgen, folgt die Art, wie CICS die Standardaktion für WS-Addressing erstellt, dem im Schema beschriebenen Verhalten.

Das folgende Muster wird von CICS in WSDL 2.0-Dokumenten verwendet, die dem Empfehlungsschema folgen, um eine Standardaktion für Fehler zu erstellen. Beachten Sie, dass der Fehlername weggelassen wird.

[zielnamensbereich]/[schnittstellename]/

Das folgende Muster wird von CICS in WSDL 2.0-Dokumenten verwendet, die dem Übergabeschema folgen, um eine Standardaktion für Fehler zu erstellen:

[zielnamensbereich]/[schnittstellename]/[fehlername]

Beispiel für die Standardaktionen, die von CICS für ein WSDL 2.0-Dokument generiert werden

In diesem Beispiel wird veranschaulicht, wie CICS Standardaktionen für ein WSDL 2.0-Dokument entsprechend dem Empfehlungsschema erstellt:

```
<beschreibung_zielnamensbereich="http://example.ibm.com/namespace" ...>
...
<schnittstellename="bookingInterface">
  <operationsname="makeBooking" pattern="http://www.w3.org/ns/wsd1/in-out">
    <eingabeelement="tns:makeBooking" messageLabel="In"/>
    <ausgabeelement="tns:bookingResponse" messageLabel="Out"/>
  </operation>
</interface>
...
</definitions>
```

Das WSDL-Fragment hat die folgenden Adressierungseigenschaften:

Eigenschaftsname	Wert
[zielnamensbereich]	http://example.ibm.com/namespace
[schnittstellename]	bookingInterface
[operationsname]	makeBooking
[anweisungstoken]	Entweder Request oder Response

Die folgenden Eingabe- und Ausgabeaktionen werden aus diesen Werten erstellt:

Aktion	Wert
Eingabeaktion	http://example.ibm.com/namespace/bookingInterface/makeBookingRequest
Ausgabeaktion	http://example.ibm.com/namespace/bookingInterface/makeBookingResponse

Weitere Informationen finden Sie in der Spezifikation W3C WS-Addressing 1.0 Metadata.

Nachrichtenaustausch

Die Web-Service-Adressierung (WS-Addressing) unterstützt diesen Nachrichtenaustausch: unidirektionale, bidirektionale Anforderung/Antwort, synchrone Anforderung/Antwort und asynchrone Anforderung/Antwort.

Der Nachrichtenaustausch per Web-Service-Adressierung beinhaltet Nachrichtenadressierungseigenschaften (MAPs) und Endpunktreferenzen (EPRs).

Zur Laufzeit stellt CICS sicher, dass der SOAP-Header der Anforderungsnachricht die relevante WS-Addressing-Nachrichteninformationen enthält; die Requester-Anwendung muss die WS-Addressing-Header nicht festlegen und weiß möglicherweise nicht einmal, dass WS-Addressing verwendet wird.

Unidirektional

Diese einfache unidirektionale Nachricht ist als reine Eingabeoperation definiert. Die Web Service Description Language (WSDL) für diese Operation hat das folgende Format:

```
<operation name="myOperation">
  <input message="tns:myInputMessage"/>
</operation>
```

Wenn Sie WS-Addressing verwenden, fügt CICS die <wsa:Action>-MAPs und die <wsa:MessageID>-MAP zum SOAP-Nachrichtenheader der WS-Addressing-Anforderungsnachricht zur Laufzeit hinzu, um Kompatibilität mit der WS-Addressing-Spezifikation sicherzustellen.

Die <wsa:MessageID>-MAP ist eine eindeutige ID; ist sie nicht angegeben, generiert CICS diese ID automatisch.

Die <wsa:Action>-MAPs werden aus der WSDL abgeleitet und in der WSBind-Datei gespeichert.

Sie können die Werte dieser MAPs mithilfe der WS-Addressing-API-Befehle von CICS überschreiben.

Bidirektionale Anforderung/Antwort

Dieser bidirektionale Austausch umfasst eine Anforderungsnachricht und eine Antwortnachricht. Der Antwortteil der Operation kann als Ausgabenachricht, als Fehlermeldung oder beides definiert werden. Die WSDL-Definition für eine Anforderung/Antwort-Operation hat das folgende Format:

```
<operation name="myOperation">
  <input message="tns:myInputMessage"/>
  <output message="tns:myOutputMessage"/>
  <fault="tns:myFaultMessage"/>
</operation>
```

Antworten auf Anforderungen oder Fehler, die von Anforderungen generiert werden, die an Endpunkte übertragen werden, richten sich an die <wsa:ReplyTo>-MAP oder die <wsa:FaultTo>-MAP, je nachdem, ob der Antworttyp normal oder ein Fehler ist.

Geben Sie eine <wsa:ReplyTo>- oder <wsa:FaultTo>-MAP in der Anforderungsnachricht an, um anzugeben, wohin die Antwort gesendet werden muss.

Wenn Sie die Empfehlungsspezifikationen verwenden und keinen Wert für die MAP <wsa:ReplyTo> angeben, nimmt die MAP <wsa:ReplyTo> standardmäßig eine Endpunktreferenz an, die den anonymen URI (<http://www.w3.org/2005/08/addressing/anonymous>) enthält, der dafür sorgt, dass CICS die Antwort zurück an den Requester sendet.

Wenn Sie die Empfehlungsspezifikationen verwenden und keinen Wert für die MAP <wsa:FaultTo> angeben, nimmt die MAP <wsa:FaultTo> standardmäßig den Wert der MAP <wsa:ReplyTo> an.

Wenn der Requester MAPs erstellt, die fehlerhaft sind und Validierungsfehler verursachen, sendet CICS die Fehlnachricht zurück an den Requester statt an die in der MAP <wsa:FaultTo> angegebene Adresse.

Synchrone Anforderung/Antwort

Standardmäßig wird der Antwortteil einer bidirektionalen Nachricht entsprechend dem zugrunde liegenden Protokoll zurückgegeben. Im Fall einer HTTP-Anforderung wird die Antwort synchron in der HTTP-Antwort zurückgegeben.

Asynchrone Anforderung/Antwort

Eine asynchrone Antwort richtet sich an einen anderen Web-Service und wird nicht an die ursprüngliche Requester-Anwendung zurückgegeben. Im Fall einer HTTP-Anforderung wird die Verbindung mit dem anfordernden Client mit einer Antwort vom Typ HTTP 202 geschlossen. Wenn der Web-Service-Provider auf einem CICS-System ausgeführt wird, empfängt die Requester-Anwendung eine leere Antwortnachricht. Wenn der Web-Service-Provider auf einem WebSphere MQ-System ausgeführt wird, empfängt die Requester-Anwendung keine Antwort.

Um das Ziel des Antwortteils einer bidirektionalen Nachricht zu ändern, müssen Sie die entsprechenden Adressen in der MAP <wsa:ReplyTo> oder den MAPs <wsa:ReplyTo> und <wsa:FaultTo> angeben.

.

Eine vollständige Liste der obligatorischen MAPs in WSDL 1.1 und WSDL 2.0 finden Sie unter „Obligatorische Nachrichtenadressierungseigenschaften (MAPs) für WS-Addressing“.

Obligatorische Nachrichtenadressierungseigenschaften (MAPs) für WS-Addressing

Die WS-Addressing 1.0-Metadatenspezifikation gibt an, welche Nachrichtenadressierungseigenschaften (MAPs) in WSDL 1.1- und WSDL 2.0-Dokumenten enthalten sein müssen. Die CICS-Implementierung von WS-Addressing hilft Ihnen dabei, die WS-Addressing-Spezifikationen einzuhalten, indem automatisch Werte für diese obligatorischen MAPs bereitgestellt werden.

Sie können Ihre eigenen Werte für MAPs in der von Ihnen bereitgestellten WSDL angeben und Sie können diese Werte im Adressierungskontext mithilfe der API-Befehle für CICS WS-Addressing aktualisieren. Wenn Sie keine Werte für die obligatorischen MAPs angeben, generiert CICS an Ihrer Stelle Werte.

In der folgenden Tabelle ist aufgelistet, welche MAPs für die unterstützten Nachrichtenaustauschmuster (MEPs) mit WSDL 1.1 und WSDL 2.0 obligatorisch sind:

Tabelle 14. Obligatorische Nachrichtenadressierungseigenschaften (MAPs) für WS-Addressing

Name der WS-Addressing-MAP	Beschreibung	Obligatorisch in WSDL 1.1	Obligatorisch in WSDL 2.0
<wsa:To>	Die Adresse des beabsichtigten Empfängers der Nachricht.	Nein	Nein
<wsa:Action>	Die WS-Addressing-Aktion: Eingabe, Ausgabe oder Fehler.	Obligatorisch für die folgenden MEPs: Unidirektional Bidirektional (Anforderung) Bidirektional (Antwort)	Obligatorisch für die folgenden MEPs: in-only robust in-only (Ein) robust in-only (Fehler) in-out (Ein) in-out (Aus) in-optional-out (Ein) in-optional-out (Aus)
<wsa:From>	Der Endpunkt, von dem die Nachricht stammt.	Nein	Nein
Dieser	Wert	ist nicht	erforderlich
<wsa:ReplyTo>	Der Endpunkt des geplanten Empfängers von Antworten auf die Nachricht.	Nein	Nein
<wsa:FaultTo>	Der Endpunkt des geplanten Empfängers von Fehlern in Bezug auf die Nachricht.	Nein	Nein
<wsa:MessageID>	Eine eindeutige Nachrichten-ID.	Obligatorisch für die folgenden MEPs: Bidirektional (Anforderung)	Obligatorisch für die folgenden MEPs: robust in-only (Ein) in-out (Ein) in-optional-out (Ein)
<wsa:RelatesTo>	Ein Wertepaar, das angibt, in welcher Beziehung diese Nachricht zu anderen Nachrichten steht. Dieses Element enthält die <wsa:MessageID> der zugehörigen Nachricht und ein optionales Attribut übermitteln den Beziehungstyp.	Obligatorisch für die folgenden MEPs: Bidirektional (Antwort)	Obligatorisch für die folgenden MEPs: robust in-only (Fehler) in-out (Aus) in-optional-out (Aus)

Weitere Informationen finden Sie in der Spezifikation *W3C WS-Addressing 1.0 Metadata* unter <http://www.w3.org/TR/ws-addr-metadata/>.

Hinweise:

- Wenn kein Wert für das Adressenelement der MAP <wsa:ReplyTo> angegeben ist, wird für die Adresse der anonyme URI festgelegt: <http://www.w3.org/2005/08/addressing/anonymous>. Der anonyme URI gibt an, dass Antworten an den Requester zurückgesendet werden.

- Wenn kein Wert für das Adressenelement der MAP <wsa:FaultTo> angegeben ist, legt CICS für diese Adresse denselben Wert fest wie für das Adressenelement der MAP <wsa:ReplyTo>.

Beachten Sie, dass wenn der Requester MAPs erstellt, die fehlerhaft sind und zu Validierungsfehlern führen, sendet CICS die Fehlnachricht zurück an den Requester statt an die in der MAP <wsa:FaultTo> angegebene Adresse.

- Wenn kein Wert für die MAP <wsa:To> angegeben ist, legt CICS für die Adresse den anonymen URI fest: <http://www.w3.org/2005/08/addressing/anonymous>. Der anonyme URI gibt an, dass die Anforderung an die Adresse gesendet werden soll, die im Container DFHWS-URI angegeben ist. Weitere Informationen finden Sie unter „Container DFHWS-URI“ auf Seite 181.
- Sie können die <wsa:Action>-MAPs explizit in Ihrem WSDL-Dokument definieren oder Sie können sie von CICS automatisch generieren lassen.
- CICS legt automatisch einen eindeutigen Wert für die MAP <wsa:MessageID> zur Laufzeit für Anforderungsnachrichten fest, die eine Antwort erwarten, ebenso wie für Antwortnachrichten.
- Die MAP <wsa:RelatesTo> ist obligatorisch für Antwortnachrichten. Der Beziehungstyp der Nachricht ist optional. Informationen zum Standardwert finden Sie unter '<http://www.w3.org/2005/08/addressing/reply>'.

Sicherheit der Web-Service-Adressierung

Die Kommunikation über ein öffentliches Netz mit Web-Service-Adressierung (WS-Addressing) muss angemessen sicher sein und es muss ein ausreichendes Maß an Vertrauen zwischen den kommunizierenden Parteien aufgebaut werden. Es wird empfohlen, Sicherheit auf Transportebene zu verwenden, z. B. SSL oder HTTPS, um Ihre Kommunikation zu sichern.

Sicherheit auf Transportebene, z. B. SSL oder HTTPS, ist die direkteste Methode, um sicherzustellen, dass Ihre WS-Addressing-Kommunikation sicher ist. Wenn Sicherheit auf Transportebene nicht verfügbar ist, können Sie Ihre Nachrichten signieren, indem Sie die WS-Addressing-Nachrichtenadressierungseigenschaften signieren und die Endpunktreferenzen verschlüsseln.

CICS kann keine Header signieren, die WS-Addressing-MAPs enthalten, und auch keine Endpunktreferenzen verschlüsseln. Aber CICS kann Signaturen in eingehenden Nachrichten überprüfen und Verschlüsselte Header entschlüsseln. Wenn Sie Ihre Kommunikation mithilfe von Signaturen und Verschlüsselung sichern möchten, müssen Sie ein externes Sicherheitsgateway verwenden, z. B. IBM WebSphere DataPower XML Security Gateway. Weitere Informationen finden Sie unter IBM WebSphere DataPower XML Security Gateway.

Beispiel für Web-Service-Adressierung

Dieses Beispiel gibt eine allgemeine Übersicht über den Prozess, der ausgeführt wird, wenn ein Kunde eine Bestellung bei einem Unternehmen aufgibt, das Web-Service-Adressierung zum Senden von Nachrichten verwendet.

Ein internationales Unternehmen, das elektronische Komponenten verkauft, nutzt die Web-Service-Adressierung. Die Infrastruktur dieses Unternehmens besteht aus einem Client für Bestellungen, einer Gruppe von Verteilungsservices, einem Fulfillment-Service und einem Konfigurationsservice.

Die Verwendung von WS-Addressing bietet dem Unternehmen die folgenden Vorteile:

- WS-Addressing stellt einen transportunabhängigen Mechanismus für die Übertragung von Nachrichten bereit. Dies fördert die Interoperabilität zwischen Web-

Services, die auf verschiedenen Plattformen ausgeführt werden. In diesem Beispiel werden die Verteilungsservices des Unternehmens auf einer Vielzahl von Plattformen ausgeführt. WS-Addressing vereinfacht die Interoperabilität zwischen verschiedenen Plattformen, da es für die Web-Service-Requester und -Provider nicht von Bedeutung ist, auf welcher Plattform der Service ausgeführt wird, mit dem sie Nachrichten austauschen.

- WS-Addressing kann verwendet werden, um das Ziel der Antwortnachricht zu ändern, indem die EPR in der MAP `<wsa:ReplyTo>` aktualisiert wird. In diesem Beispiel ändert der the Fulfillment-Service das Ziel der Antwortnachricht, wenn er den Verteilungsservice auswählt, an den die Nachricht umgeleitet wird.

Das Unternehmen hat verschiedene Verteilungszentren in einer Reihe unterschiedlicher Länder. Jedes dieser Verteilungszentren wird in diesem Beispiel durch einen Verteilungsservice dargestellt und ist beim Konfigurationsservice registriert.

Der Fulfillment-Service wählt aus, welcher Verteilungsservice sich am besten eignet, um die Bestellung zu verarbeiten, basieren auf einer Reihe von Faktoren, darunter die Verfügbarkeit von Elementen und die Entfernung des Verteilungszentrums vom Kunden.

Adressierungsinformationen werden an den und vom Konfigurationsservice übergeben. Der Konfigurationsservice speichert die Adressen der verfügbaren Services in Form von Endpunktreferenzen. Neue Services werden beim Konfigurationsservice registriert, indem eine EPR mithilfe des Befehls **WSAEPR CREATE** erstellt und an den Konfigurationsservice gesendet wird. Der Konfigurationsservice erfordert, dass die EPR in Form eines XML-Blocks vorliegt, sodass der Parameter **WSADDR-EPR-ANY** in DFHWS2LS auf TRUE gesetzt werden muss. Die Option **WSADDR-EPR-ANY=TRUE** wird verwendet, um CICS anzuweisen, die EPR als ein `<xsd:any>`-Element zu behandeln. CICS muss sie in einen Container stellen, statt sie zur Laufzeit in eine Sprachstruktur umzuwandeln.

Die Art und Weise, in der diese Services interagieren, wird im folgenden Diagramm dargestellt. Das Diagramm zeigt andere Services, die von der Task ausgeschlossen wurden, aber in einer Geschäftsanwendung relevant sein könnten:

- Ein Überwachungsservices, der von allen anderen Services mit dem Status der Bestellung aktualisiert werden kann.
- Ein Fehlerbehebungsservice für die Verarbeitung aller auftretenden Fehlermeldungen.
- Ein Rückrufservice für Bestellclients zur Verarbeitung aller an den Bestellclient gerichteten Antwortnachrichten.

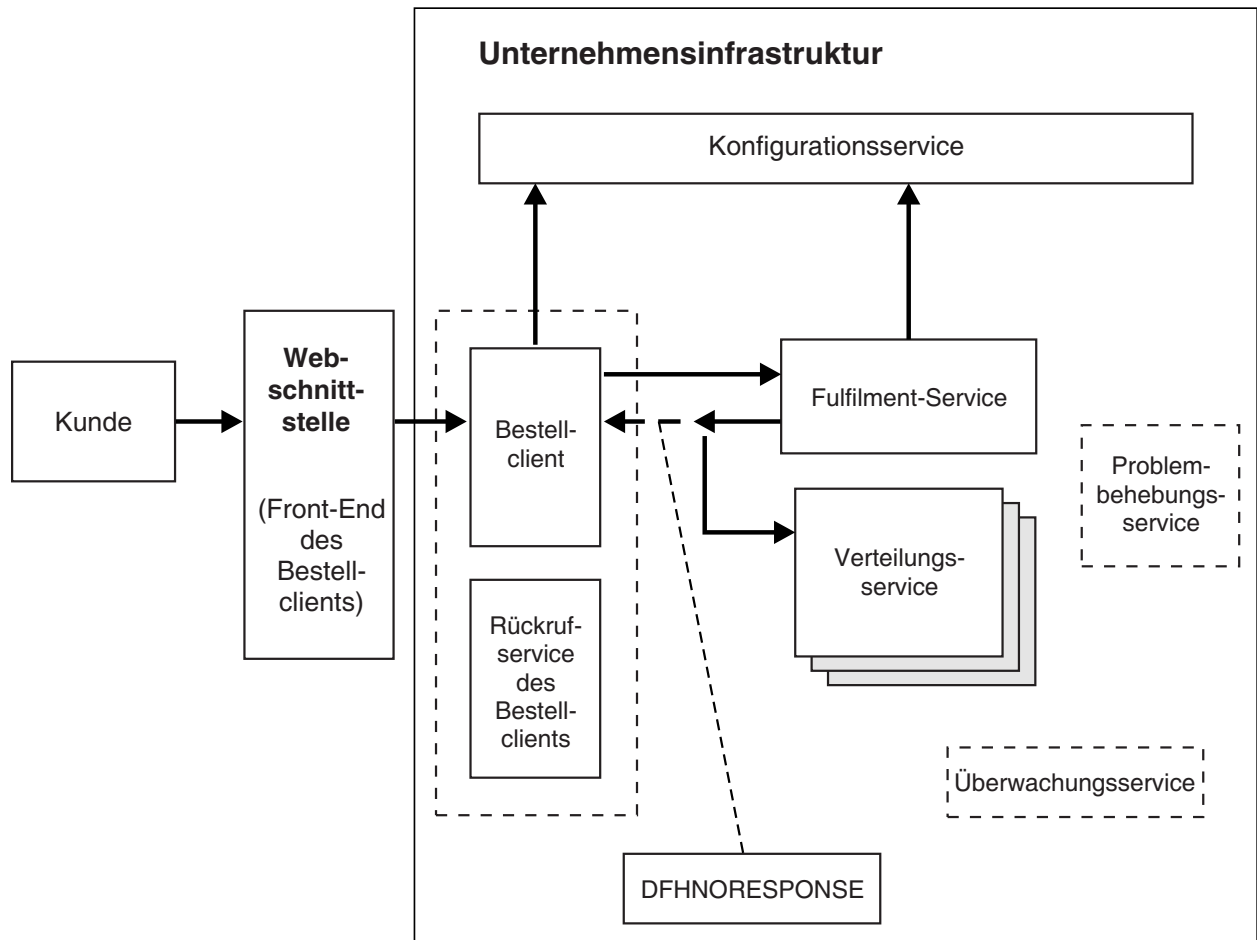


Abbildung 26. Unternehmensinfrastruktur

Die folgenden Schritte beschreiben den Prozess, der ab dem Zeitpunkt, an dem ein Kunde eine Bestellung aufgibt, bis zu dem Zeitpunkt, an dem die Bestellung verarbeitet wird, stattfindet.

1. Ein Kunde gibt eine Bestellung bei dem Unternehmen auf.
 - a. Der Kunde gibt die Bestellung auf der Website des Unternehmens auf, bei der es sich um das Front-End des Bestellclients handelt.
 - b. Der Bestellclient erfasst die Kundenkontaktdetails als Teil der Bestellung.
 - c. Der Bestellclient gibt eine Bestätigung und eine eindeutige Bestellreferenz über die Webschnittstelle an den Kunden zurück.
2. Der Bestellclient sendet die Bestellanforderung an den Fulfillment-Service.
 - a. Wenn der Bestellclient nicht bereit die EPR für den Fulfillment-Service kennt, fordert er sie vom Konfigurationsservice an. Der involvierte Prozess, wenn der Bestellclient die EPR des Fulfillment-Service vom Konfigurationsservice anfordert, ist im Abschnitt Beispiel für <wsa:To> detailliert beschrieben.
 - b. Der Bestellclient gibt den Befehl **INVOKE SERVICE** für den Fulfillment-Service aus. WS-Addressing leitet die Nachricht an die Adresse weiter, die von der 'To'-EPR im Anforderungsadressierungskontext angegeben ist.
3. Der Fulfillment-Service wählt einen Verteilungsservice für die Verarbeitung der Bestellung aus und leitet die Antwortnachricht an diesen Service weiter.

- a. Der Fulfillment-Service verwendet einen Befehl **WSACONTEXT GET**, um die Bestellreferenz und andere Adressierungseigenschaften aus dem Adressierungskontext zu extrahieren.
- b. Der Fulfillment-Service wählt den geeignetsten Verteilungsservice aus dem Konfigurationsservice aus.
- c. Die <wsa:ReplyTo>EPR wird zum Adressierungskontext hinzugefügt:

```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Address>http://www.example.ibm.com/DistributionService</wsa:Address>
</wsa:EndpointReference>
```

Der Fulfillment-Service verwendet den Befehl **WSACONTEXT BUILD**, um die 'ReplyTo'-EPR des ausgewählten Verteilungsservice zum Anforderungsadressierungskontext hinzuzufügen.

- d. Der Fulfillment-Service verwendet den Befehl **WSACONTEXT BUILD** wiederholt, um die Bestellreferenz und andere Informationen zum Anforderungsadressierungskontext hinzuzufügen.
 - e. Ein DFHNORESPONSE-Container wird zur Bestellclient-Pipeline hinzugefügt, um den Bestellclient anzuweisen, dass er keine Antwort empfangen wird, und die Antwortnachricht wird in Form einer Anforderungsnachricht an den Verteilungsservices weitergeleitet.
4. Der Verteilungsservice empfängt die weitergeleitete Antwortnachricht und verarbeitet die Bestellung.
 - a. Der Verteilungsservice verwendet einen Befehl **WSACONTEXT GET**, um die Bestellreferenz und Adressierungsdetails aus dem Anforderungsadressierungskontext zu extrahieren.
 - b. Der Verteilungsservice verarbeitet die Bestellung.

Beispiel für <wsa:To>

1. Der Bestellclient fordert die EPR des Service an, an den eine Nachricht vom Konfigurationsservice gesendet werden soll. In diesem Beispiel fordert der Bestellclient die EPR des Fulfillment-Service an.
2. Der Konfigurationsservice erstellt und sendet eine Antwortnachricht.
 - a. Der Konfigurationsservice erstellt die angeforderte <wsa:To>-EPR für den Fulfillment-Service mithilfe des API-Befehls **WSAEPR CREATE: EXEC CICS WSAEPR CREATE**.
 - b. Der Konfigurationsservice schreibt die Ausgabe des Befehls **WSAEPR CREATE** in einen Container: **EXEC CICS PUT CONTAINER(work-cont)**.
 - c. Der Konfigurationsservice kopiert den Containernamen in das Element **myEpr-xml-cont: MOVE work-cont TO myEpr-xml-cont**.
 - d. Der Konfigurationsservice sendet eine Antwortnachricht an den Bestellclient. Diese Nachricht enthält den Inhalt des Containers, der vom Container **myEpr-xml-cont** benannt wird. In diesem Beispiel wird der Inhalt des Containers **work-cont** an den Bestellclient im Element <wsa:myEpr> gesendet:

```
<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>

```

Abb. 27 zeigt den Anforderung/Antwort-Nachrichtenaustausch zwischen dem Bestellclient und dem Konfigurationsservice. Dieser Nachrichtenaustausch bezieht zwei typische Web-Service-Pipelines ein.

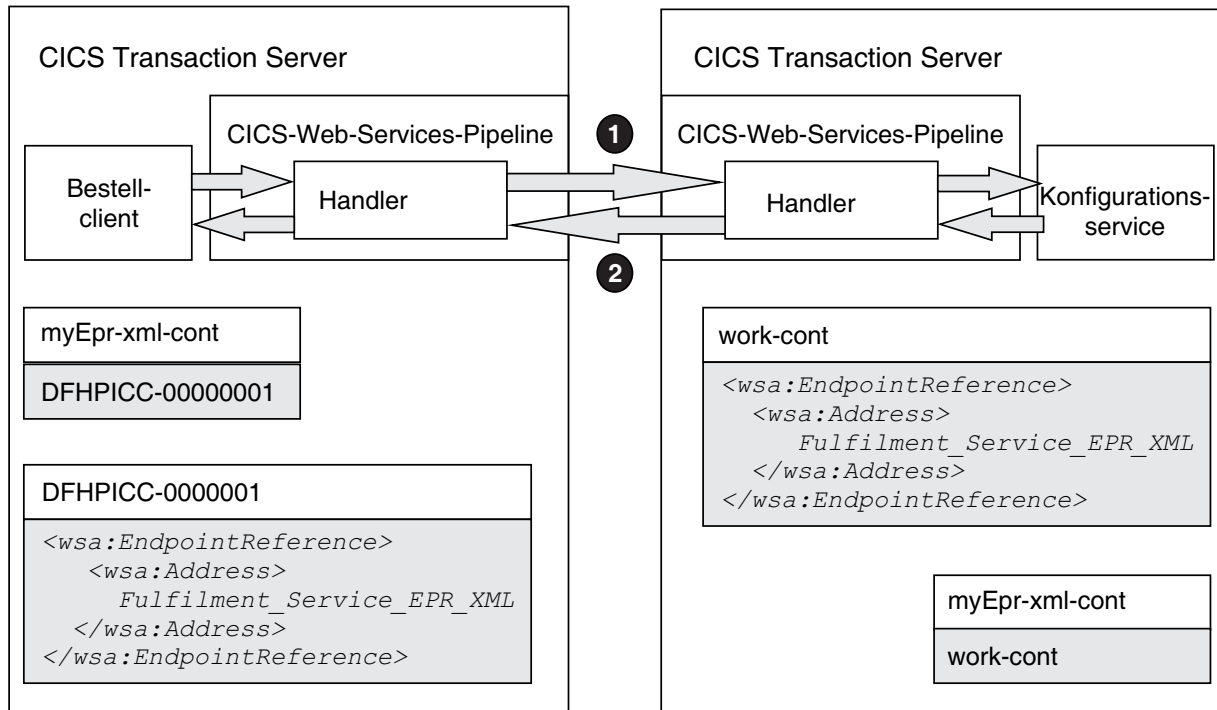


Abbildung 27. Anforderung/Antwort-Nachrichtenaustausch zwischen dem Bestellclient und dem Konfigurationsservice

3. Der Bestellclient empfängt die Antwortnachricht, erstellt die <wsa:To>-EPR und sendet eine Anforderung an den Fulfillment-Service:
 - a. Der Bestellclient extrahiert die <wsa:To>-EPR-Daten aus der Antwortnachricht.
 - b. CICS füllt einen eindeutigen Container, in diesem Beispiel den Container DFHPICC-00000001, mit den <wsa:To>-EPR-Daten.
 - c. CICS kopiert den Namen des Containers, in diesem Beispiel DFHPICC-00000001, in das Element myEpr-xml-cont.
 - d. Der Bestellclient liest den Inhalt des durch das Element myEpr-xml-cont angegebenen Containers und stellt ihn als Eingabe für den API-Befehl **WSACONTEXT BUILD** bereit. Der Befehl **WSACONTEXT BUILD** verwendet diese Eingabe, um die <wsa:To>-EPR für den Fulfillment-Service zu erstellen.
 - e. Der Bestellclient gibt einen Befehl **INVOKE SERVICE** aus, der die Pipelineverarbeitung initialisiert.
 - f. Der Adressierungshandler für CICS-Web-Services, DFHWSADH, in der ausgehenden Pipeline konvertiert die <wsa:To>-EPR in eine Adresse und ein optionales Set von Referenzparametern, die es im Header der SOAP-Anforderungsnachricht platziert, die an den Fulfillment-Service gesendet wird:

```

<env:Header>
  <wsa:To>http://example.ibm.com/Fulfilment_Service</wsa:To>
</env:Header>

```

Abb. 28 zeigt die Anforderung des Bestellclients an den Fulfillment-Service. Diese Anforderung umfasst eine Web-Service-Pipeline, die den Adressierungshandler für CICS-Web-Services, DFHWSADH, enthält.

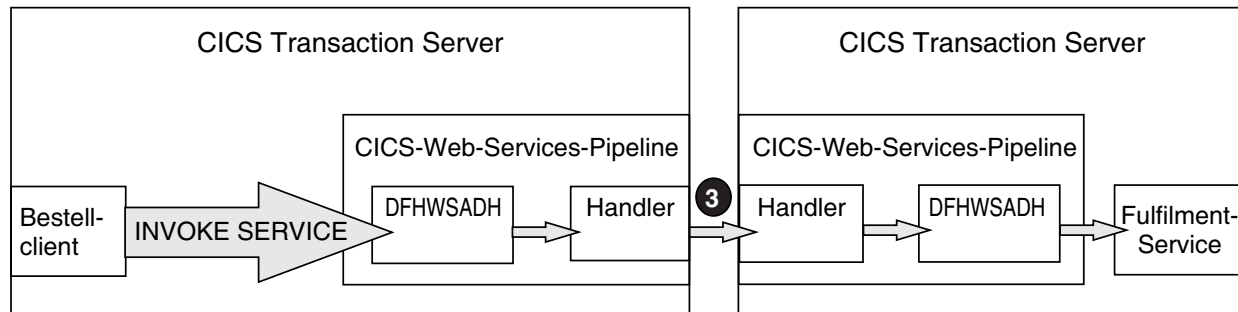


Abbildung 28. Anforderung des Bestellclients an den Fulfillment-Service

Terminologie der Web-Service-Adressierung

Begriffe zur Erläuterung der WS-Addressing-Unterstützung (Web-Service-Adressierung).

Adressierungskontext

Ein XML-Dokument, das WS-Addressing-Nachrichtenadressierungseigenschaften (MAPs) speichert, bevor sie in SOAP-Anforderungsnachrichten gesendet werden und nachdem sie von SOAP-Anforderungs- und Antwortnachrichten empfangen wurden.

Endpunktreferenz (EPR)

Eine XML-Struktur, die Adressierungsinformationen enthält, die verwendet werden, um eine Nachricht an einen Web-Service weiterzuleiten. Diese Adressierungsinformationen enthalten die Zieladresse der Nachricht, optionale Referenzparameter zur Verwendung durch die Anwendung sowie optionale Metadaten.

Nachrichtenadressierungseigenschaft (MAP)

Ein XML-Element, das Adressierungsinformationen für eine bestimmte Web-Service-Nachricht übermittelt, z. B. eine eindeutige Nachrichten-ID, das Ziel der Nachricht und die Endpunktreferenzen der Nachricht.

Unterstützung für SAML

CICS unterstützt die Verwendung von Security Assertion Markup Language (SAML), um Sicherheitsinformationen zu beschreiben und zwischen Online-Geschäftspartnern auszutauschen.

CICS unterstützt die Standards SAMLCore1.1 und SAML Core2.0. Es unterstützt nicht die Protokolle, die in diesen Standard beschrieben sind.

Sie können Provider- und Requester-Pipelines konfigurieren, um SAML-Token zu verwenden, müssen aber zunächst den CICS-Sicherheitstokenservice (STS) implementieren. Weitere Informationen zum Konfigurieren Ihrer CICS-Installation zur Unterstützung von SAML finden Sie unter *Configuring CICS for SAML*.

Kapitel 3. Web-Services entwickeln

JSON-Web-Service erstellen

Sie können vorhandene CICS-Anwendungen als JSON-Web-Services verfügbar machen und neue CICS-Anwendungen erstellen, die als JSON-Web-Service-Provider fungieren.

Before you begin

Vor dem Erstellen eines JSON-Web-Service müssen Sie Ihr CICS-System für die Unterstützung von JSON-Web-Services konfigurieren. Weitere Informationen finden Sie unter *Creating the CICS infrastructure for a JSON service provider*.

About this task

Der CICS-JSON-Assistent ist ein bereitgestelltes Dienstprogramm, das Sie beim Erstellen der erforderlichen Artefakte für eine neue Anwendung eines JSON-Web-Service-Providers bzw. beim Aktivieren einer vorhandenen Anwendung als JSON-Web-Service-Provider unterstützt.

Der CICS-JSON-Assistent kann ein JSON-Schema aus einer Struktur einer höheren Programmiersprache bzw. eine Struktur einer höheren Programmiersprache aus einem vorhandenen JSON-Schema erstellen. Er unterstützt COBOL, C/C++ und PL/I. Er generiert außerdem Informationen, die zum Aktivieren der automatischen Laufzeitkonvertierung der JSON-Nachrichten in Container und Kommunikationsbereiche und umgekehrt dienen. Diese Informationen werden von der JSON-Web-Service-Unterstützung in CICS während der Pipelineverarbeitung verwendet.

Erstellen Sie Ihren JSON-Web-Service wie in der folgenden Prozedur beschrieben und prüfen Sie, ob er korrekt funktioniert:

Procedure

1. Erstellen Sie einen JSON-Web-Service. Verwenden Sie den JSON-Assistenten, um das JSON-Schema oder die Sprachstrukturen zu erstellen und in CICS zu implementieren. Setzen Sie den Befehl **PIPELINE SCAN** ab, um automatisch die erforderlichen CICS-Ressourcen zu erstellen.
2. Starten Sie den JSON-Web-Service, um zu testen, ob er wie gewünscht funktioniert.

What to do next

Diese Schritte werden in den folgenden Themen ausführlicher erläutert.

CICS JSON-Assistent

Der CICS JSON-Assistent besteht aus einer Gruppe von Stapeldienstprogrammen, die eine Zuordnung zwischen Strukturen eines JSON-Schemas und Sprachstrukturen erstellen. Diese Zuordnung wird von CICS während der Laufzeit verwendet, um die Umsetzung zwischen JSON und Anwendungsdaten auszuführen. Der Assistent unterstützt eine schnelle Implementierung von CICS-Anwendungen für die Verwendung in Service-Providern und Service-Requestern, mit minimalem Programmieraufwand.

Wenn Sie den JSON-Assistenten für CICS verwenden, müssen Sie nicht Ihren eigenen Code für das Parsen von eingehenden Nachrichten und für das Zusammenstellen von ausgehenden Nachrichten schreiben. CICS ordnet Daten zwischen der JSON-Nachricht und der Datenstruktur eines Anwendungsprogramms zu.

Der Assistent kann ein JSON-Schema aus einer Struktur einer höheren Programmiersprache oder aus einem vorhandenen JSON-Schema erstellen und unterstützt COBOL, C/C++ und PL/I. Er generiert außerdem Informationen, die zum Aktivieren einer automatischen Laufzeitkonvertierung der JSON-Nachrichten in Container und Kommunikationsbereiche und umgekehrt dienen.

Der CICS JSON-Assistent umfasst zwei Dienstprogramme:

DFHLS2JS

Generiert eine Web-Service-Bindungsdatei aus einer Sprachstruktur. Das Dienstprogramm generiert außerdem ein JSON-Schema.

DFHJS2LS

Generiert eine Web-Service-Bindungsdatei aus einem JSON-Schema. Dieses Dienstprogramm generiert außerdem eine Sprachstruktur, die Sie in Ihren Anwendungsprogrammen verwenden können.

Die JCL-Prozeduren zum Ausführen beider Programme befinden sich in der Bibliothek *HLQ.XDFHINST*.

Der relevante Verwendungsmodus für die Prozedur DFHLS2JS oder DFHJS2LS hängt von Ihren Anforderungen ab:

- DFHLS2JS: Konvertierung einer höheren Programmiersprache in ein JSON-Schema für eine verknüpfbare Schnittstelle
- DFHJS2LS: Konvertierung eines JSON-Schemas in eine höhere Programmiersprache für eine verknüpfbare Schnittstelle
- DFHLS2JS: Konvertierung einer höheren Programmiersprache in ein JSON-Schema für Anforderung/Antwort-Services
- DFHJS2LS: Konvertierung eines JSON-Schemas in eine höhere Programmiersprache für Anforderung/Antwort-Services
- DFHJS2LS: Konvertierung eines JSON-Schemas in eine höhere Programmiersprache für REST-konforme Services

Weitere Informationen zu den Dienstprogrammen und Datenzuordnungen des JSON-Assistenten finden Sie in den folgenden Themen.

DFHLS2JS: Konvertierung einer höheren Programmiersprache in ein JSON-Schema für Anforderung/Antwort-Services

Die Prozedur DFHLS2JS generiert aus einer Datenstruktur einer höheren Programmiersprache ein JSON-Schema. Sie können DFHLS2JS verwenden, wenn Sie ein CICS-Anwendungsprogramm als Service-Provider verfügbar machen.

Die JCL-Prozedur DFHLS2JS wird in der Datei *HLQ.XDFHINST* installiert, wobei *HLQ* das übergeordnete Qualifikationsmerkmal für die Installationsposition von CICS ist.

Jobsteueranweisungen für DFHLS2JS

JOB Startet den Job.

EXEC Gibt den Prozedurnamen an (DFHLS2JS).

INPUT.SYSUT1 DD

Gibt die Eingabe an. Die Eingabeparameter werden in der Regel im Eingabedatenstrom angegeben. Sie können jedoch in einer Datei oder in einem Member einer partitionierten Datei definiert werden.

Symbolische Parameter

Die folgenden symbolischen Parameter sind in DFHLS2JS definiert:

JAVADIR = *pfad*

Gibt den Namen des Java-Verzeichnisses an, das von DFHLS2JS verwendet wird. Der Wert dieses Parameters wird an /usr/lpp/ angehängt, um den vollständigen Pfadnamen /usr/lpp/*pfad* zu erstellen.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **JAVADIR** angegeben wurde.

PATHPREF = *präfix*

Gibt ein Präfix an, das den z/OS UNIX-Verzeichnispfad erweitert, der in anderen Parametern verwendet wird, bzw. '' (leere Zeichenfolge), wenn kein Präfix verwendet wird.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **PATHPREF** angegeben wurde.

SERVICE = *wert*

Verwenden Sie diesen Parameter nur, wenn sie vom IBM Support dazu aufgefordert werden.

TMPDIR = *tmp-verzeichnis*

Gibt die Position eines Verzeichnisses in z/OS UNIX an, das DFHLS2JS als temporären Arbeitsbereich verwendet. Die Benutzer-ID, mit der der Job ausgeführt wird, muss über Lese- und Schreibberechtigungen für dieses Verzeichnis verfügen.

Der Standardwert ist /tmp.

TMPFILE = *tmp-präfix*

Gibt ein Präfix an, das DFHLS2JS verwendet, um die Namen der temporären Arbeitsbereichsdateien zu erstellen.

Der Standardwert ist LS2JS.

USSDIR = *pfad*

Gibt den Namen des CICS TS-Verzeichnisses im UNIX System Services-Dateisystem an. Der Wert dieses Parameters wird an /usr/lpp/cicsts/ angehängt, um den vollständigen Pfadnamen /usr/lpp/cicsts/*pfad* zu erstellen. Er muss als '.' (Punkt) angegeben werden, wenn der Standardwert verwendet wird.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **USSDIR** angegeben wurde.

Der temporäre Arbeitsbereich

DFHLS2JS erstellt die folgenden drei temporären Dateien zur Laufzeit:

tmp-verzeichnis/tmp-präfix.in

tmp-verzeichnis/tmp-präfix.out

tmp-verzeichnis/tmp-präfix.err

Dabei gilt Folgendes:

tmp-verzeichnis ist der Wert, der im Parameter **TMPDIR** angegeben ist.

tmp-prefix ist der Wert, der im Parameter **TMPFILE** angegeben ist.

Die Standardnamen für die Dateien lauten wie folgt, wenn **TMPDIR** und **TMPFILE** nicht angegeben sind:

/tmp/LS2JS.in

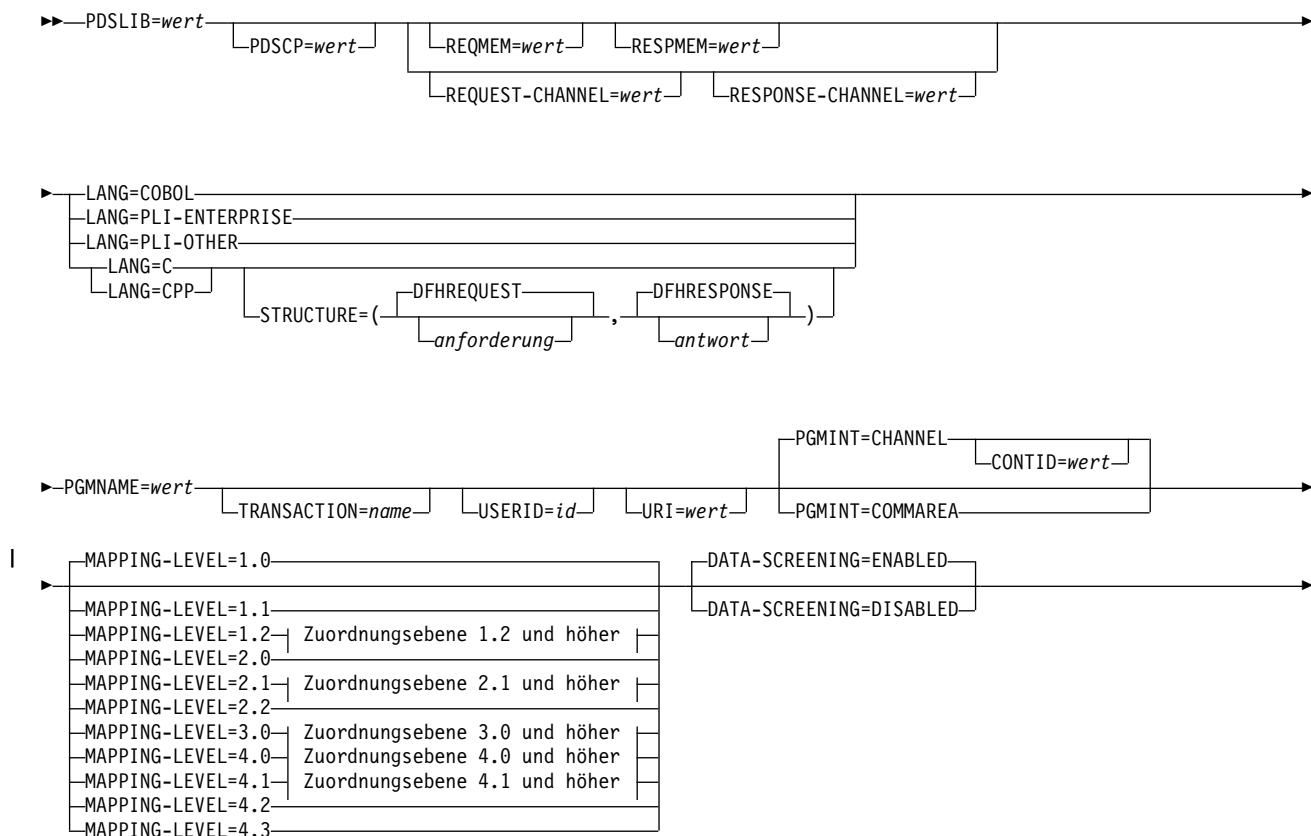
/tmp/LS2JS.out

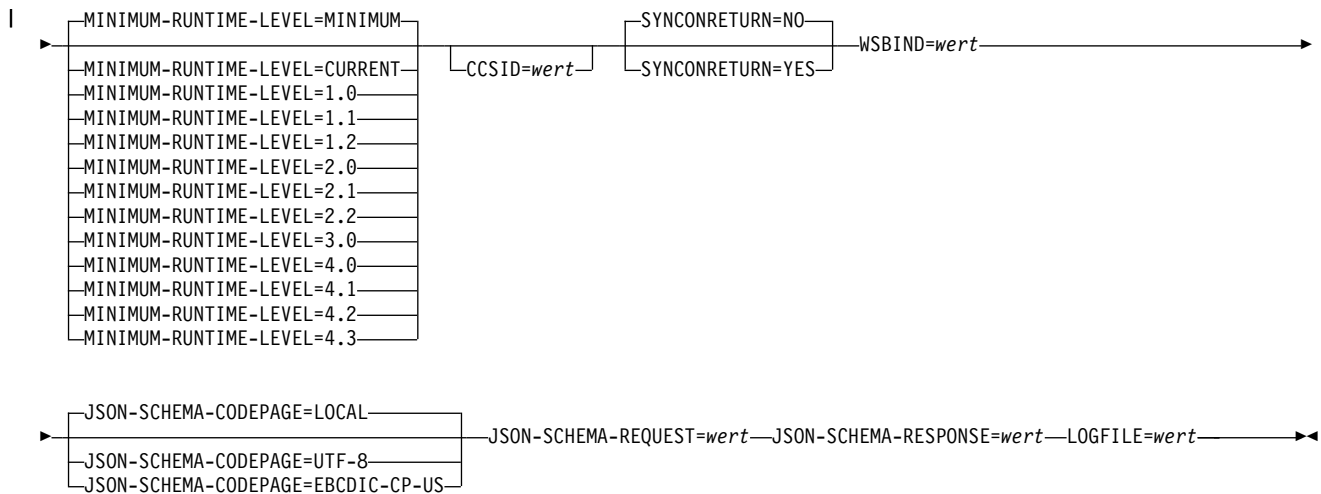
/tmp/LS2JS.err

Important: DFHLS2JS sperrt nicht den Zugriff auf die z/OS UNIX-Dateien oder -Dateimember. Wenn zwei oder mehr Instanzen von DFHLS2JS gleichzeitig ausgeführt werden und dieselben temporären Arbeitsbereichsdateien verwenden, kann nicht verhindert werden, dass ein Job die Arbeitsbereichsdateien überschreibt, während ein anderer Job sie verwendet, was zu unvorhersehbaren Fehlern führen kann.

Deshalb wird empfohlen, eine Namenskonvention und Betriebsprozeduren auszuarbeiten, die diese Situation verhindern. Sie können beispielsweise den symbolischen Systemparameter **SYSUID** verwenden, um Arbeitsbereichsdateinamen zu generieren, die für einen einzelnen Benutzer eindeutig sind. Diese temporären Dateien werden gelöscht, bevor der Job beendet wird.

Eingabeparameter für DFHLS2JS

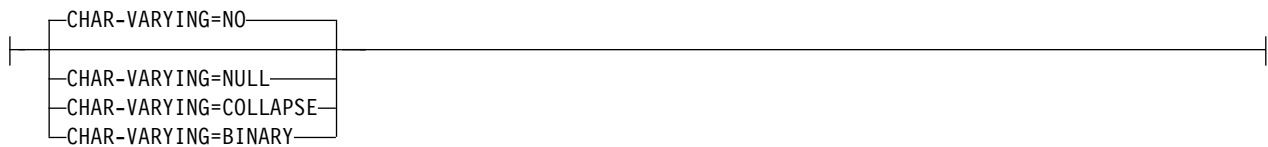




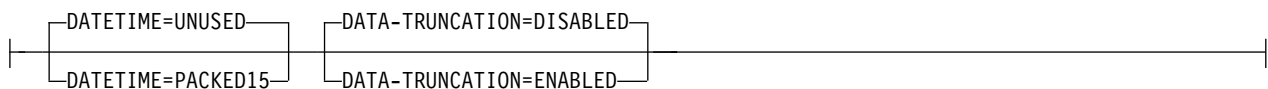
Zuordnungsebene 1.2 und höher:



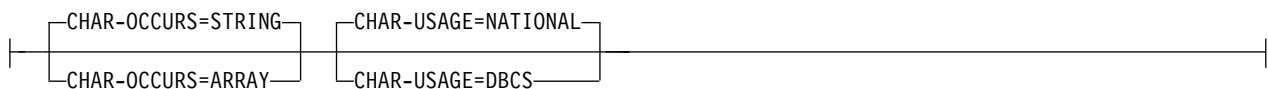
Zuordnungsebene 2.1 und höher:



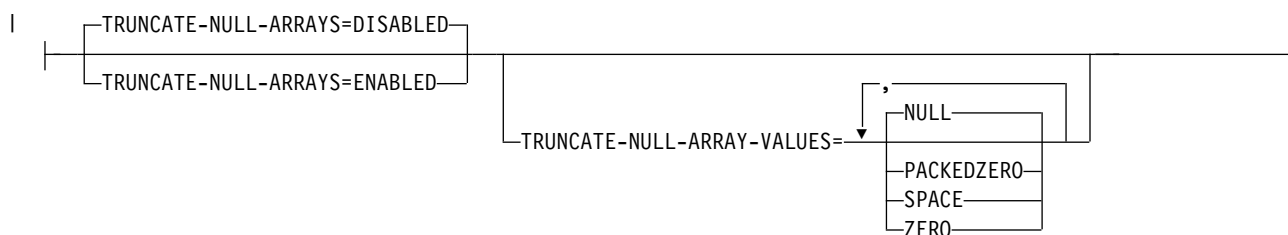
Zuordnungsebene 3.0 und höher:



Zuordnungsebene 4.0 und höher:



Zuordnungsebene 4.1 und höher:



Parameterverwendung

- Sie können die Eingabeparameter in beliebiger Reihenfolge angeben.
- Jeder Parameter muss in einer neuen Zeile eingegeben werden.
- Ein Parameter (und ggf. das Fortsetzungszeichen) darf nicht über Spalte 72 hinausgehen; die Spalten 73-80 müssen Leerzeichen enthalten.
- Wenn ein Parameter zu lang ist, um in eine einzelne Zeile zu passen, verwenden Sie am Ende der Zeile einen Stern (*), um anzugeben, dass der Parameter in der nächsten Zeile fortgesetzt wird. Alles, einschließlich der Leerzeichen, vor dem Stern wird als Teil des Parameters betrachtet. Beispiel:

```
WSBIND=wsbinddir*  
      /app1
```

ist äquivalent zu

```
WSBIND=wsbinddir/app1
```

- Ein #-Zeichen an der ersten Zeichenposition der Zeile ist ein Kommentarzeichen. Die Zeile wird ignoriert.
- Ein Komma an der letzten Zeichenposition der Zeile ist ein optionales Zeilentrennzeichen und wird ignoriert.

Parameterbeschreibungen

CCSID = *wert*

Gibt die CCSID an, die zur Laufzeit verwendet wird, um Zeichendaten in der Anwendungsdatenstruktur zu codieren. Der Wert dieses Parameters überschreibt den Wert des Systeminitialisierungsparameters **LOCALCCSID**. *wert* muss eine EBCDIC-CCSID sein, die von Java und z/OS Unicode Services User's Guide and Reference unterstützt wird. Wenn Sie diesen Parameter nicht angeben, wird die Anwendungsdatenstruktur unter Verwendung der im Systeminitialisierungsparameter angegebenen CCSID codiert.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Gibt an, wie Zeichenfelder in der Sprachstruktur auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Ein Zeichenfeld in COBOL ist eine Picture-Klausel vom Typ X, z. B. PIC(X) 10; ein Zeichenfeld in C/C++ ist ein Zeichenarray. Sie haben folgende Optionen:

NO Zeichenfelder werden einer JSON-Zeichenfolge zugeordnet und als Felder mit fester Länge verarbeitet. Die maximale Länge der Daten ist gleich der Länge des Felds. **NO** ist der Standardwert für den Parameter **CHAR-VARYING** für COBOL und PL/I auf den Zuordnungsebenen 2.0 und niedriger.

Dieser Wert gilt nicht für Enterprise PL/I und andere PL/I-Sprachstrukturen.

NULL Zeichenfelder werden einer JSON-Zeichenfolge zugeordnet und als auf null endende Zeichenfolgen verarbeitet. CICS fügt bei der Umsetzung aus einer JSON-Nachricht ein abschließendes Nullzeichen (NULL) hinzu. Die maximale Länge der Zeichenfolge wird als ein Zeichen weniger als die in der Sprachstruktur angegebene Länge berechnet. NULL ist der Standardwert für den Parameter **CHAR-VARYING** für C/C++.

Dieser Wert gilt nicht für Enterprise PL/I und andere PL/I-Sprachstrukturen.

COLLAPSE

Zeichenfelder werden einer JSON-Zeichenfolge zugeordnet. Abschließende und eingebettete Leerzeichen in dem Feld werden nicht in die JSON-Nachricht eingeschlossen, z. B. wird
<leerzeichen>AB<leerzeichen><leerzeichen><leerzeichen>C<leerzeichen>
zu AB<leerzeichen>C. Die eingehende JSON-Nachricht wird geparkt, um alle führenden, abschließenden und eingebetteten Leerzeichen zu entfernen. COLLAPSE ist der Standardwert für den Parameter **CHAR-VARYING** für COBOL und PL/I ab Zuordnungsebene 2.1.

BINARY

Zeichenfelder werden einer JSON-Zeichenfolge zugeordnet, die Base64-codierte Daten enthält, und als Felder mit fester Länge verarbeitet. Der BINARY-Wert im Parameter **CHAR-VARYING** ist nur ab Zuordnungsebene 2.1 verfügbar.

Weitere Informationen zur Behandlung von Werten mit variabler Länge und Leerzeichen finden Sie unter „Unterstützung für Werte variabler Länge und Leerzeichen“ auf Seite 393.

CHAR-OCCURS = { STRING | **ARRAY** }

Gibt an, wie Zeichenarrays in der Sprachstruktur auf Zuordnungsebene 4.0 oder höher zugeordnet werden. Beispiel: PIC X OCCURS 20. Dieser Parameter kann nur in der COBOL-Sprache verwendet werden.

ARRAY

Zeichenarrays werden einem JSON-Array zugeordnet. Das heißt, dass jedes Zeichen als einzelnes JSON-Element zugeordnet wird. Dies ist auch das Verhalten auf Zuordnungsebene 3.0 und niedriger.

STRING

Zeichenarrays werden einer JSON-Zeichenfolge zugeordnet. Das heißt, dass das gesamte COBOL-Array als einzelnes JSON-Element zugeordnet wird.

CHAR-USAGE = { NATIONAL | **DBCS** }

In COBOL kann der nationale Datentyp, PIC N, für UTF-16- oder DBCS-Daten verwendet werden. Diese Einstellung wird durch die Compileroption NSYMBOL gesteuert. Sie müssen den Parameter **CHAR-USAGE** im Assistenten auf denselben Wert setzen wie die Compileroption NSYMBOL, um sicherzustellen, dass die Daten richtig verarbeitet werden. Dieser Wert wird in der Regel auf CHAR-USAGE=NATIONAL gesetzt, wenn Sie UTF-16 verwenden.

DBCS Daten aus PIC(*n*)-Feldern werden als DBCS-codierte Daten behandelt.

NATIONAL

Daten aus PIC(*n*)-Feldern werden als UTF-16-codierte Daten behandelt.

CONTID = *wert*

Gibt in einem Service-Provider den Namen des Containers an, der die übergeordnete Datenstruktur enthält, die zur Darstellung einer JSON-Nachricht verwendet wird.

Die Länge des Containers, die CICS an das Zielanwendungsprogramm übergibt, ist die größere der beiden Längen des Anforderungscontainers bzw. des Antwortcontainers.

DATA-SCREENING = { **ENABLED** | **DISABLED** }

Gibt an, ob die von Anwendungen bereitgestellten Daten auf Fehler überprüft werden.

ENABLED

Alle von Anwendungen bereitgestellten Laufzeitdaten, die nicht mit der Sprachstruktur konsistent sind, werden als Fehler behandelt, und es wird die Nachricht DFHPI1010 ausgegeben. Eine Fehlerantwort wird an die Anwendung zurückgegeben.

DISABLED

Werte in den von Anwendungen bereitgestellten Laufzeitdaten, die nicht konsistent mit der Sprachstruktur sind, werden durch Standardwerte ersetzt. Eine Null ersetzt beispielsweise einen fehlerhaften Wert in einem numerischen Feld. Die Nachricht DFHPI1010 wird nicht ausgegeben und es wird eine normale Antwort an die Anwendung zurückgegeben. Dieses Feature kann verwendet werden, um die Fehlerantworten **INVALID_PACKED_DEC** und **INVALID_ZONED_DEC** zu vermeiden, die von nicht initialisierten Ausgabefeldern generiert werden.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Gibt an, ob Daten mit variabler Länge in einer Feldstruktur mit fester Länge toleriert werden:

DISABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, weist CICS die abgeschnittenen Daten zurück und gibt eine Fehlermeldung aus.

ENABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, toleriert CICS die abgeschnittenen Daten und verarbeitet die fehlenden Daten als Nullwerte.

DATETIME = { **UNUSED** | **PACKED15** }

Gibt an, ob potenzielle ABSTIME-Felder in der Struktur einer höheren Programmiersprache als Zeitmarken zugeordnet werden:

PACKED15

Gepackte Dezimalfelder mit einer Länge von 15 (8 Byte) werden als CICS ABSTIME-Felder behandelt und als Zeitmarken zugeordnet.

UNUSED

Gepackte Dezimalfelder mit einer Länge von 15 (8 Byte) werden nicht als Zeitmarken behandelt.

Sie können diesen Parameter auf der Zuordnungsebene 3.0 festlegen.

JSON-SCHEMA-CODEPAGE = { **LOCAL** | **UTF-8** | **EBCDIC-CP-US** }

Gibt die Codepage an, die zum Generieren der JSON-Schemadokumente verwendet wird.

LOCAL

Gibt an, dass die JSON-Schemas unter Verwendung der Standardcodepage für das Dateisystem generiert werden.

UTF-8 Gibt an, dass die JSON-Schemas unter Verwendung der UTF-8-Codepage generiert werden.

EBCDIC-CP-US

Gibt an, dass die JSON-Schemas unter Verwendung der US EBCDIC-Codepage generiert werden.

JSON-SCHEMA-REQUEST = *wert*

Dies ist ein obligatorischer Parameter.

Der Wert gibt die UNIX System Services-Position an, an der das JSON-Anforderungsschema gespeichert ist.

JSON-SCHEMA-RESPONSE = *wert*

Dies ist ein obligatorischer Parameter.

Der Wert gibt die UNIX System Services-Position an, an der das JSON-Antwortschema gespeichert ist.

LANG = **COBOL** | **PLI-ENTERPRISE** | **PLI-OTHER** | **C** | **CPP**

Gibt die Datenstruktur der höheren Programmiersprache an:

COBOL

COBOL

PLI-ENTERPRISE

Enterprise PL/I

PLI-OTHER

Andere PL/I-Ebene als Enterprise PL/I

C

C

CPP

C++

LOGFILE = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die DFHLS2JS das Aktivitätenprotokoll und die Traceinformationen schreibt. DFHLS2JS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur.

In der Regel verwenden Sie diese Datei nicht, aber dies kann von der IBM Serviceorganisation angefordert werden, wenn Sie Probleme mit DFHLS2JS feststellen.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 }

Gibt die Zuordnungsebene an, die DFHLS2JS bei der Generierung der Web-Service-Bindungsdatei und des JSON-Schemas verwendet. Sie haben folgende Optionen:

1.0 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

1.1 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

1.2 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

- 2.0 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.1 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.2 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 3.0 Verwenden Sie diese Zuordnungsebene, um ein JSON-Schema unter Verwendung aller verfügbaren Optionen zu generieren.
- 4.0 Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher. Auf dieser Zuordnungsebene können Sie OCCURS DEPENDING ON-COBOL-Felder und den Parameter **CHAR-OCCURS** verwenden.
- 4.1 Zur Unterstützung abschneidbarer Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher.
- 4.2 Keine signifikanten Änderungen. Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.
- 4.3 Keine signifikanten Änderungen. Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.

Weitere Informationen zu Zuordnungsebenen finden Sie unter Zuordnungsebenen für die CICS JSON-Assistenten.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 | CURRENT }

Gibt die älteste CICS-Laufzeitumgebung an, in der die Web-Service-Bindungsdatei implementiert werden kann. Wenn Sie eine Version auswählen, die nicht mit den anderen Parametern übereinstimmt, die Sie angegeben haben, erhalten Sie eine Fehlermeldung. Sie haben folgende Optionen:

MINIMUM

Die niedrigste mögliche Laufzeitebene von CICS wird anhand der von Ihnen ausgewählten Parameter automatisch zugewiesen.

- 1.0 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 1.1 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 1.2 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.0 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.1 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

- 2.2 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 3.0 Die generierte Web-Service-Bindungsdatei wird in einer Region unter CICS TS 4.1 oder höher implementiert.

Anmerkung: JSON-Unterstützung ist erst ab CICS TS 4.2 verfügbar.

- 4.0 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.0 oder niedriger verwenden. Sie können alle optionalen Parameter auf dieser Ebene verwenden.
- 4.1 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.1 oder niedriger verwenden.
- 4.2 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.2 oder niedriger verwenden.
- 4.3 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.3 oder niedriger verwenden.

CURRENT

Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS-Region auf derselben Laufzeitebene implementiert, die Sie zum Generieren der Web-Service-Bindungsdatei verwenden.

PDSLIB = *wert*

Gibt den Namen der partitionierten Datei an, die die zu verarbeitenden Datenstrukturen der höheren Programmiersprachen enthält. Die Dateimember, die für die Anforderung und Antwort verwendet werden, sind in den Parametern **REQMEM** und **RESPMEM** angegeben.

Restriction: Die Datensätze in der partitionierten Datei müssen eine feste Länge von 80 Bytes haben.

PDSCP = *wert*

Gibt die Codepage an, die in den Mitgliedern einer partitionierten Datei verwendet wird, die in den Parametern **REQMEM** und **RESPMEM** angegeben sind. Dabei ist *wert* eine CCSID-Nummer oder eine Java-Codepagenummer. Wenn dieser Parameter nicht angegeben wird, wird die Codepage von z/OS UNIX System Services verwendet. Beispielsweise können Sie **PDSCP** = 037 angeben.

PGMINT = { **CHANNEL** | **COMMAREA** }

Gibt für einen Service-Provider an, wie CICS Daten an das Ziellanwendungsprogramm übergibt.

CHANNEL

CICS verwendet eine Kanalschnittstelle, um Daten an das Ziellanwendungsprogramm zu übergeben.

- Auf Zuordnungsebenen niedriger als 3.0 kann der Kanal nur einen Container enthalten, der sowohl für Eingaben als auch für Ausgaben

verwendet wird. Mit dem Parameter **CONTID** geben Sie den Namen des Containers an. Der Standardname ist DFHWS-DATA.

- Auf Zuordnungsebene 3.0 kann der Kanal mehrere Container enthalten. Verwenden Sie die Parameter **REQUEST-CHANNEL** und **RESPONSE-CHANNEL**. Geben Sie nicht **PDSLIB**, **REQMEM** oder **RESPMEM** an.

COMMAREA

CICS verwendet einen Kommunikationsbereich, um Daten an das Zielanwendungsprogramm zu übergeben.

Wenn das Zielanwendungsprogramm die Anforderung verarbeitet hat, muss es denselben Mechanismus verwenden, um die Antwort zurückzugeben. Wenn die Anforderung in einem Kommunikationsbereich empfangen wurde, muss die Antwort im Kommunikationsbereich zurückgegeben werden. Wenn die Anforderung in einem Container empfangen wurde, muss die Antwort in einem Container zurückgegeben werden. Die Länge des Kommunikationsbereichs oder des Containers, die CICS an das Zielanwendungsprogramm zurückgibt, ist größer als die Länge des Anforderungskommunikationsbereichs oder -containers und des Antwortkommunikationsbereichs oder -containers.

PGMNAME = *wert*

Gibt den Namen der PROGRAM-Ressource von CICS für das Zielanwendungsprogramm an, das als Web-Service zugänglich gemacht wird. Der CICS-Web-Service-Support stellt eine Verknüpfung zu diesem Programm bereit.

REQMEM = *wert*

Gibt den Namen des Members der partitionierten Datei an, die die Struktur der höheren Programmiersprache für die Web-Service-Anforderung enthält. Für einen Service-Provider ist die Web-Service-Anforderung die Eingabe für das Anwendungsprogramm.

REQUEST-CHANNEL = *wert*

Gibt den Namen und die Position eines Kanalbeschreibungsdokuments an. In der Kanalbeschreibung werden die Container beschrieben, die die Web-Service-Provider-Anwendung in ihrer Schnittstelle verwenden kann, wenn sie eine JSON-Nachricht von einem Web-Service-Requester empfängt. Die Kanalbeschreibung ist ein XML-Dokument, das mit dem von CICS bereitgestellten Kanalschema übereinstimmen muss. Weitere Informationen finden Sie unter „Kanalbeschreibungsdokument erstellen“ auf Seite 297.

Sie können diesen Parameter nur auf Zuordnungsebene 3.0 verwenden.

RESPMEM = *wert*

Gibt den Namen des Members der partitionierten Datei an, die die Struktur der höheren Programmiersprache für die Web-Service-Antwort enthält. Für einen Service-Provider ist die Web-Service-Antwort die Ausgabe des Anwendungsprogramms.

RESPONSE-CHANNEL = *wert*

Gibt den Namen und die Position eines Kanalbeschreibungsdokuments an. In der Kanalbeschreibung werden die Container beschrieben, die die Web-Service-Provider-Anwendung in ihrer Schnittstelle verwenden kann, wenn sie eine JSON-Nachricht an einen Web-Service-Requester sendet. Die Kanalbeschreibung ist ein XML-Dokument, das mit dem von CICS bereitgestellten Kanalschema übereinstimmen muss. Weitere Informationen finden Sie unter „Kanalbeschreibungsdokument erstellen“ auf Seite 297.

Sie können diesen Parameter nur auf Zuordnungsebene 3.0 verwenden.

STRUCTURE = (*anforderung* , *antwort*)

Gibt nur für C und C++ die Namen der übergeordneten Strukturen an, die in

den Mitgliedern der partitionierten Dateien enthalten sind, die wiederum in den Parametern **REQMEM** und **RESPMEM** angegeben sind:

anforderung

Gibt den Namen der übergeordneten Struktur an, die die Anforderung enthält, wenn der Parameter **REQMEM** angegeben wird. Der Standardwert ist **DFHREQUEST**.

Das Mitglied der partitionierten Datei muss eine übergeordnete Struktur mit dem von Ihnen angegebenen Namen oder eine Struktur namens **DFHREQUEST** enthalten, wenn Sie keinen Namen angeben.

antwort

Gibt den Namen der übergeordneten Struktur an, die die Antwort enthält, wenn der Parameter **RESPMEM** angegeben wird. Der Standardwert ist **DFHRESPONSE**.

Wenn Sie einen Wert angeben, muss das Mitglied der partitionierten Datei eine übergeordnete Struktur mit dem von Ihnen angegebenen Namen oder eine Struktur namens **DFHRESPONSE** enthalten, wenn Sie keinen Namen angeben.

SYNCONRETURN = { **NO** | **YES** }

Gibt an, ob der ferne Web-Service einen Synchronisationspunkt absetzen kann.

NO Der ferne Web-Service kann keinen Synchronisationspunkt absetzen. Dies ist der Standardwert. Wenn der ferne Web-Service einen Synchronisationspunkt absetzt, schlägt er mit einem ADPL-Abbruch fehl.

YES Der ferne Web-Service kann einen Synchronisationspunkt absetzen. Wenn Sie **YES** auswählen, wird die ferne Task als separate Arbeitseinheit festgeschrieben, wenn die Steuerung vom fernen Web-Service zurückgegeben wird. Wenn der ferne Web-Service eine wiederherstellbare Ressource aktualisiert und ein Fehler auftritt, nachdem sie zurückgegeben wurde, kann die Aktualisierung dieser Ressource nicht zurückgesetzt werden.

TRANSACTION = *name*

In einem Service-Provider gibt dieser Parameter den ein bis vier Zeichen langen Namen einer Aliastransaktion an, die die Pipeline starten kann. Der Wert dieses Parameters wird verwendet, um das Attribut **TRANSACTION** der URIMAP-Ressource zu definieren, wenn es automatisch mithilfe des **PIPELINE**-Scanfehls erstellt wird.

Zulässige Zeichen:

A-Z a-z 0-9 \$ @ # _ < >

TRUNCATE-NULL-ARRAYS = { **DISABLED** | **ENABLED** }

Gibt an, wie strukturierte Arrays auf Zuordnungsebene 4.1 oder höher verarbeitet werden. Wenn diese Option aktiviert ist, versucht CICS, leere Datensätze innerhalb eines Arrays zu erkennen (weitere Informationen zur Identifizierung leerer Datensätze finden Sie unter **TRUNCATE-NULL-ARRAY-VALUES**). Wenn fünf aufeinanderfolgende leere Arraydatensätze erkannt werden, wird das Array bei der Generierung von XML/JSON beim ersten solchen Datensatz abgeschnitten. Diese Abschneidefunktion ist nur für Arrays mit strukturiertem Inhalt aktiviert, Arrays aus einfachen primitiven Feldern werden nicht abgeschnitten. Das Abschneiden von Arrays kann zu einer kürzeren Darstellung der Daten in JSON/XML führen, dies ist jedoch nicht ohne Risiko. Wenn fünf aufeinanderfolgende Datensätze fehlerhaft als nicht initialisierter Speicher

identifiziert werden (beispielsweise weil sie legitim niedrige Werte enthalten), kann es zu Datenverlusten kommen. Wenn TRUNCATE-NULL-ARRAYS aktiviert und TRUNCATE-NULL-ARRAY-VALUES nicht festgelegt ist, wird der Standardwert für TRUNCATE-NULL-ARRAY-VALUES verwendet.

TRUNCATE-NULL-ARRAY-VALUES = { NULL | **PACKEDZERO**|**SPACE** | **ZERO** }

Gibt an, welche Werte für die Verarbeitung von TRUNCATE-NULL-ARRAYS auf Zuordnungsebene 4.1 oder höher als leer behandelt werden. Standardmäßig wird der Nullwert ('0x00' oder 'low-values') als leer behandelt. Wenn alle Bytes des Speichers in einem Datensatz eines strukturierten Arrays Nullen enthalten, wird der gesamte Datensatz als leer betrachtet. Ein Wert oder mehrere Werte vom Typ NULL, PACKEDZERO, SPACE und ZERO können in einer durch Kommas getrennten Liste angegeben werden.

NULL Impliziert ein Nullzeichen (0x00).

PACKEDZERO

Impliziert eine gepackte Dezimalnull mit positivem Vorzeichen (0x0C), eine gepackte Dezimalnull mit negativem Vorzeichen (0x0D) oder eine gepackte Dezimalnull ohne Vorzeichen (0x0F).

SPACE

Impliziert einen SBCS EBCDIC-Bereich (0x40).

ZERO Impliziert eine gezonte Dezimalnull ohne Vorzeichen (0xF0).

Jede passende Kombination der ausgewählten Bytes innerhalb eines strukturierten Arraydatensatzes führt dazu, dass der gesamte Datensatz als leer erkannt wird.

Wenn für TRUNCATE-NULL-ARRAY-VALUES ein Wert definiert ist, muss TRUNCATE-NULL-ARRAYS aktiviert sein.

URI = *wert*

Dieser Parameter gibt den relativen oder absoluten URI an, den ein Client für den Zugriff auf den Web-Service verwendet. CICS verwendet den angegebenen Wert, wenn eine URIMAP-Ressource aus der Web-Service-Bindungsdatei generiert wird, die von DFHLS2JS erstellt wird. Der Parameter gibt die Pfadkomponente des URI an, für den die URIMAP-Definition gilt.

USERID = *id*

In einem Service-Provider gibt dieser Parameter eine ein bis acht Zeichen lange Benutzer-ID an, die von jedem Web-Client verwendet werden kann. Für eine anwendungsgenerierte Antwort oder einen Web-Service wird die Aliastransaktion unter dieser Benutzer-ID angehängt. Der Wert dieses Parameters wird verwendet, um das Attribut USERID der URIMAP-Ressource zu definieren, wenn es automatisch mithilfe des **PIPELINE**-Scanbefehls erstellt wird.

Zulässige Zeichen:

A-Z a-z 0-9 \$ @ #

WSBIND = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Web-Service-Bindungsdatei. DFHLS2JS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur. Die Dateierweiterung ist .wsbind.

Weitere Informationen

- Die Benutzer-ID, die DFHLS2JS für die Ausführung verwendet, muss für UNIX System Services konfiguriert werden. Die Benutzer-ID muss über Leseberechtigung

gungen für die z/OS UNIX-Dateistruktur und PDS-Bibliotheken unter CICS und über Schreibberechtigung für die in den Parametern **LOGFILE**, **WSBIND** und im **JSON-Schema** angegebenen Verzeichnisse verfügen.

- Der Benutzer-ID muss ausreichend Speicher zugeordnet sein, um Java auszuführen.
- Die JCL weist eine maximale Parameterlänge von 100 Zeichen auf. Die Länge des Parameters kann mithilfe der Anweisung **STDPARM** erhöht werden. Weitere Informationen finden Sie unter z/OS UNIX System Services User's Guide.

Beispiel

```
//LS2JS JOB '  
abrechnungsdaten  
,  
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: Konvertierung eines JSON-Schemas in eine höhere Programmiersprache für Anforderung/Antwort-Services

Die Prozedur DFHJS2LS generiert aus einem JSON-Schema eine Datenstruktur einer höheren Programmiersprache und eine Web-Service-Bindungsdatei. Sie können DFHJS2LS verwenden, wenn Sie die Erstellung eines CICS-Anwendungsprogramms als Service-Provider vorbereiten. In diesem Thema werden die Jobsteueranweisungen, symbolischen Parameter, Eingabeparameter und ihre Beschreibungen für DFHJS2LS aufgelistet.

Die JCL-Prozedur DFHJS2LS wird in der Datei *HLQ.XDFHINST* installiert, wobei *HLQ* das übergeordnete Qualifikationsmerkmal für die Installationsposition von CICS ist.

Jobsteueranweisungen für DFHJS2LS

JOB Startet den Job.

EXEC Gibt den Prozedurnamen an (DFHJS2LS).

INPUT.SYSUT1 DD

Gibt die Eingabe an. Die Eingabeparameter werden in der Regel im Eingabedatenstrom angegeben. Sie können jedoch in einer Datei oder in einem Member einer partitionierten Datei definiert werden.

Symbolische Parameter

Die folgenden symbolischen Parameter sind in DFHJS2LS definiert:

JAVADIR = *pfad*

Gibt den Namen des Java-Verzeichnisses an, das von DFHJS2LS verwendet wird. Der Wert dieses Parameters wird an /usr/lpp/ angehängt, um den vollständigen Pfadnamen /usr/lpp/*pfad* zu erstellen.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **JAVADIR** angegeben wurde.

PATHPREF = *präfix*

Gibt ein Präfix an, das den z/OS UNIX-Verzeichnispfad erweitert, der in anderen Parametern verwendet wird, bzw. ' ' (eine leere Zeichenfolge), wenn kein Präfix verwendet wird.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **PATHPREF** angegeben wurde.

SERVICE = *wert*

Verwenden Sie diesen Parameter nur, wenn sie vom IBM Support dazu aufgefordert werden.

TMPDIR = *tmp-verzeichnis*

Gibt die Position eines Verzeichnisses in z/OS UNIX an, das DFHJS2LS als temporären Arbeitsbereich verwendet. Die Benutzer-ID, unter der der Job ausgeführt wird, muss über Lese- und Schreibberechtigungen für dieses Verzeichnis verfügen.

Der Standardwert ist /tmp.

TMPFILE = *tmp-präfix*

Gibt ein Präfix an, das DFHJS2LS verwendet, um die Namen der temporären Arbeitsbereichsdateien zu erstellen.

Der Standardwert ist JS2LS.

USSDIR = *pfad*

Gibt den Namen des CICS TS-Verzeichnisses im UNIX System Services-Dateisystem an. Der Wert dieses Parameters wird an /usr/lpp/cicsts/ angehängt, um den vollständigen Pfadnamen /usr/lpp/cicsts/*pfad* zu erstellen. Er muss als ' .' (Punkt) angegeben werden, wenn der Standardwert verwendet wird.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **USSDIR** angegeben wurde.

Der temporäre Arbeitsbereich

DFHJS2LS erstellt die folgenden drei temporären Dateien zur Laufzeit:

tmp-verzeichnis/tmp-präfix.in

tmp-verzeichnis/tmp-präfix.out

tmp-verzeichnis/tmp-präfix.err

Dabei gilt Folgendes:

tmp-verzeichnis ist der Wert, der im Parameter **TMPDIR** angegeben ist.

tmp-präfix ist der Wert, der im Parameter **TMPFILE** angegeben ist.

Die Standardnamen für die Dateien lauten wie folgt, wenn **TMPDIR** und **TMPFILE** nicht angegeben sind:

/tmp/JS2LS.in


```

/tmp/JS2LS.out
/tmp/JS2LS.err

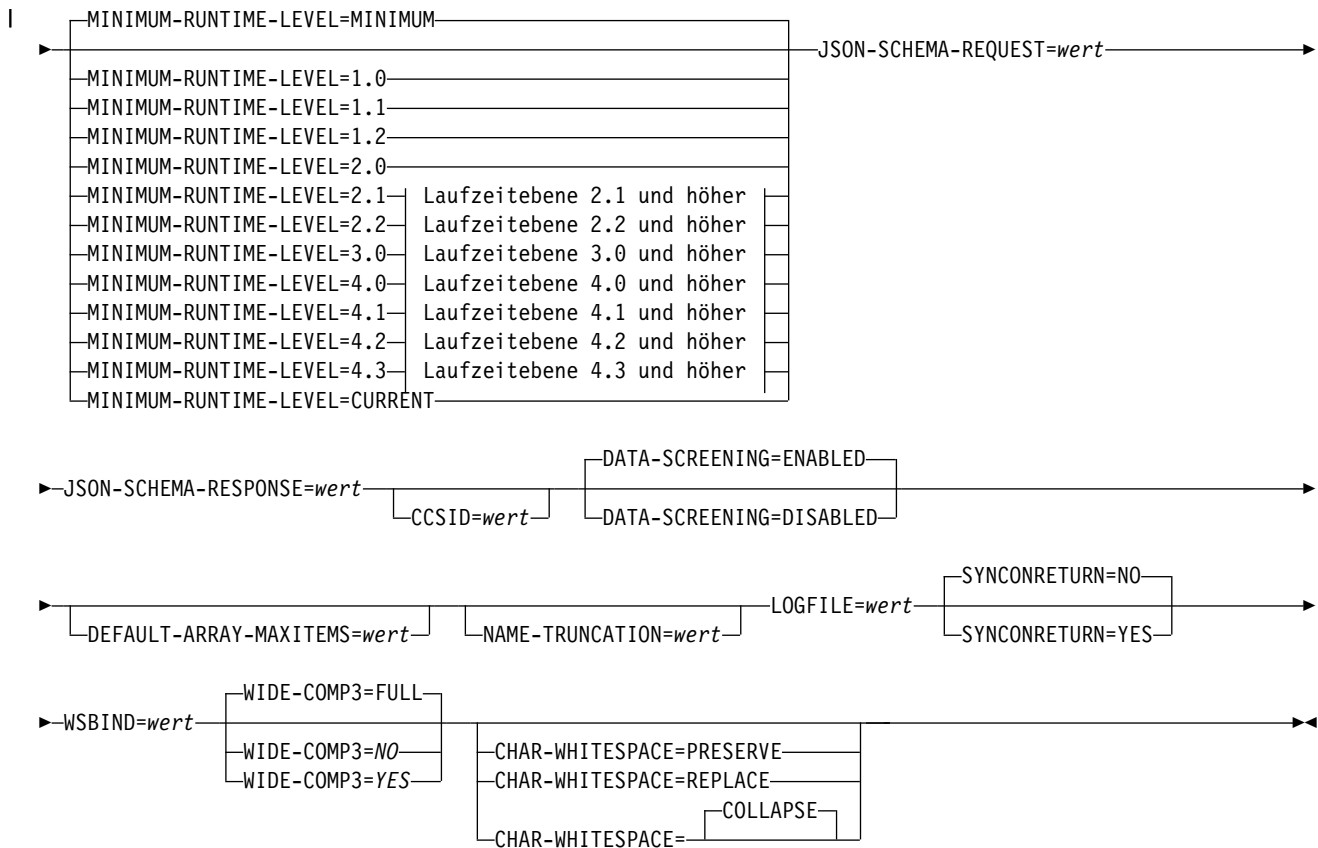
```

Important: DFHJS2LS sperrt nicht den Zugriff auf die z/OS UNIX-Dateien oder -Dateimember. Wenn folglich zwei oder mehr Instanzen von DFHJS2LS gleichzeitig ausgeführt werden und dieselben temporären Arbeitsbereichsdateien verwenden, kann nicht verhindert werden, dass ein Job die Arbeitsbereichsdateien überschreibt, während ein anderer Job sie verwendet, was zu unvorhersehbaren Fehlern führt.

Deshalb wird empfohlen, eine Namenskonvention und Betriebsprozeduren auszu-
arbeiten, die diese Situation verhindern. Sie können beispielsweise den symboli-
schen Systemparameter **SYSUID** verwenden, um Arbeitsbereichsdateinamen zu ge-
nerieren, die für einen einzelnen Benutzer eindeutig sind. Diese temporären
Dateien werden gelöscht, bevor der Job beendet wird.

Eingabeparameter für DFHJS2LS

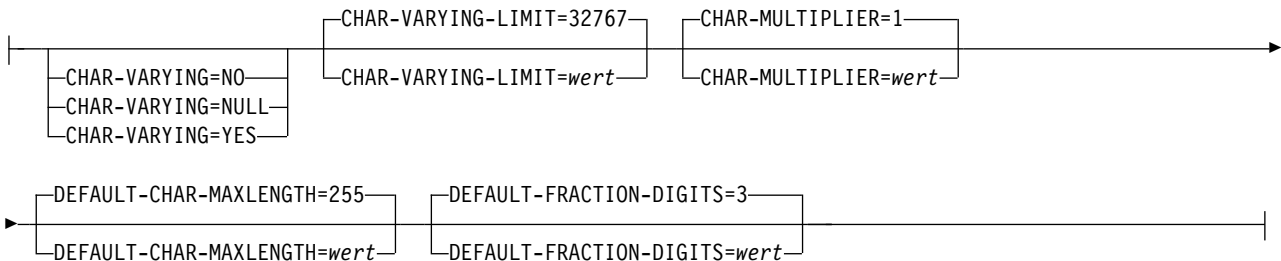




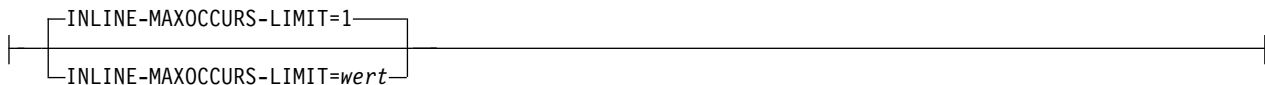
PGMINT:



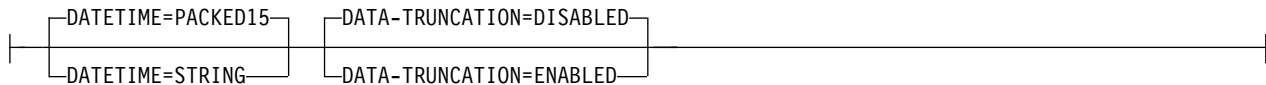
Zuordnungsebene 1.2 und höher:



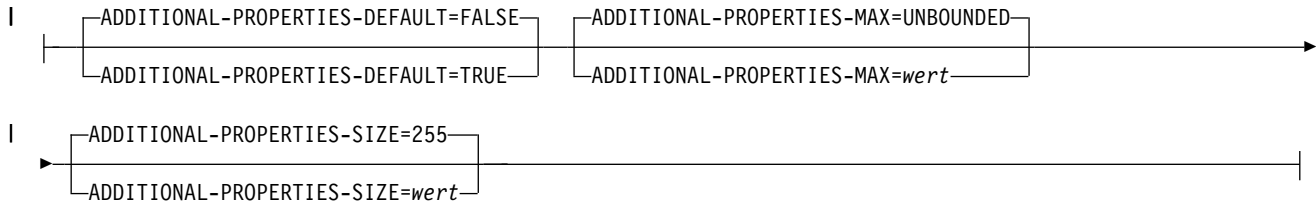
Zuordnungsebene 2.1 und höher:



Zuordnungsebene 3.0 und höher:



Zuordnungsebene 4.2 und höher:



Parameterverwendung

- Sie können die Eingabeparameter in beliebiger Reihenfolge angeben.
- Jeder Parameter muss in einer neuen Zeile eingegeben werden.
- Ein Parameter (und ggf. das Fortsetzungszeichen) darf nicht über Spalte 72 hinausgehen; die Spalten 73-80 müssen Leerzeichen enthalten.
- Wenn ein Parameter zu lang ist, um in eine einzelne Zeile zu passen, verwenden Sie am Ende der Zeile einen Stern (*), um anzugeben, dass der Parameter in der nächsten Zeile fortgesetzt wird. Alles, einschließlich der Leerzeichen, vor dem Stern wird als Teil des Parameters betrachtet. Beispiel:

```
WSBIND=wsbinddir*  
      /app1
```

ist äquivalent zu

```
WSBIND=wsbinddir/app1
```

- Ein #-Zeichen an der ersten Zeichenposition der Zeile ist ein Kommentarzeichen. Die Zeile wird ignoriert.
- Ein Komma an der letzten Zeichenposition der Zeile ist ein optionales Zeilentrennzeichen und wird ignoriert.

Parameterbeschreibungen

ADDITIONAL-PROPERTIES-DEFAULT = { **true** | **false** }

Gibt an, ob JSON-Schemaobjekte, die zusätzliche Eigenschaften nicht explizit unterstützen, als unterstützende Elemente interpretiert werden oder nicht. Zusätzliche JSON-Eigenschaften sind alle Eigenschaften in einem JSON-Objekt, die im JSON-Schema nicht vordefiniert sind. Diese Eigenschaften werden in der Regel vom Datenumsetzungsmechanismus als unerwartete Zusatzdaten zurückgewiesen. Wenn **ADDITIONAL-PROPERTIES-DEFAULT** auf TRUE gesetzt ist, oder wenn das JSON-Schema explizit `additionalProperties:true` für ein Objekt festlegt, wird ein Bereich für solche Werte in den generierten Copybooks reserviert. Anwendungen können mit diesen Werten interagieren, indem sie die zugeordneten Felder in den Copybooks verwenden.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

ADDITIONAL-PROPERTIES-MAX = { **0-20** | **UNBOUNDED** }

Gibt an, wie viele zusätzliche Eigenschaften für ein entsprechendes JSON-Objekt unterstützt werden. Siehe **ADDITIONAL-PROPERTIES-DEFAULT**. Die generierten

Copybooks enthalten Strukturen, die sich für die Adressierung aller zusätzlichen Eigenschaften eignen. Standardmäßig gibt es keine maximale Einschränkung für die Anzahl der unterstützten Eigenschaften. Die Copybooks werden in ähnlicher Weise wie Arrays ohne Einschränkungen generiert und verwenden Container. Mit diesem Parameter kann eine maximale Einschränkung angewendet werden, die in Kombination mit dem Parameter **INLINE-MAXOCCURS-LIMIT** dafür sorgt, dass ein Array mit fester Länge für die maximale Anzahl von Eigenschaften zugeordnet werden kann, wodurch keine Container erforderlich sind.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

ADDITIONAL-PROPERTIES-SIZE = { **16-32767** | **255** }

Gibt die maximale Größe für jede der zusätzlichen JSON-Eigenschaften an. Wenn ein JSON-Objekt zusätzliche Eigenschaften unterstützt, wie in **ADDITIONAL-PROPERTIES-DEFAULT** definiert, dann haben die generierten Copybooks Bindungen, um Eigenschaften bis zu der von **ADDITIONAL-PROPERTIES-MAX** angegebenen Anzahl zu unterstützen. Standardmäßig beträgt der für jede zusätzliche Eigenschaft unterstützte Maximalwert 255 Zeichen. Ein Feld dieser Größe wird in den erstellten Copybooks generiert. Diese Größe kann durch Festlegen des Parameters **ADDITIONAL-PROPERTIES-SIZE** angepasst werden. So wird beispielsweise ein JSON-Objekt verarbeitet, das folgende Eigenschaft enthält:

```
"example": { "notes": "this extra property was not defined in the JSON Schema"
}
```

Wenn die Copybooks generiert wurden, um zusätzliche Eigenschaften zu unterstützen, wird der gesamte Wert zur Verarbeitung an die Anwendung übergeben. Der Wert beginnt mit dem führenden Anführungszeichen vor dem Eigenschaftsschlüssel und endet mit der abschließenden rechten geschweiften Klammer im Eigenschaftswert. In diesem Beispiel sind es etwa 100 Zeichen. Der Wert für **ADDITIONAL-PROPERTIES-SIZE** muss groß genug sein, um den größtmöglichen Wert enthalten zu können. Wenn der zugeordnete Puffer für den verarbeiteten Wert zu klein ist, wird eine Fehlerantwort generiert.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

CCSID = *wert*

Gibt die CCSID an, die zur Laufzeit verwendet wird, um Zeichendaten in der Anwendungsdatenstruktur zu codieren. Der Wert dieses Parameters überschreibt den Wert des Systeminitialisierungsparameters **LOCALCCSID**. *wert* muss eine EBCDIC-CCSID sein, die von Java und z/OS Unicode Services User's Guide and Reference unterstützt wird. Wenn Sie diesen Parameter nicht angeben, wird die Anwendungsdatenstruktur unter Verwendung der im Systeminitialisierungsparameter angegebenen CCSID codiert.

CHAR-MULTIPLIER = { **1** | *wert* }

Gibt die Anzahl von Bytes an, die für die einzelnen Zeichen auf Zuordnungsebene 1.2 oder höher zulässig sein sollen. Der Wert (*wert*) dieses Parameters kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein. Alle nicht numerischen, zeichenbasierten Zuordnungen unterliegen diesem Multiplikator. Binäre, numerische, in Zonen eingeteilte und gepackte Dezimalfelder unterliegen diesem Multiplikator nicht.

Dieser Parameter kann nützlich sein, wenn Sie beispielsweise planen, DBCS-Zeichen zu verwenden, bei denen Sie sich für einen Multiplikator von 3 entscheiden können, um zur Laufzeit Platz zu lassen für mögliche SO- und SI-Zeichen (shift-out = Umschalttaste nicht gedrückt, shift-in = Umschalttaste gedrückt) um die einzelnen Doppelbytezeichen herum.

Wenn Sie **CCSID=1200** (gibt UTF-16 an) festlegen, sind nur 2 oder 4 gültige Werte für **CHAR-MULTIPLIER**. Wenn Sie UTF-16 verwenden, ist der Standardwert 2. Verwenden Sie **CHAR-MULTIPLIER=2**, wenn Sie erwarten, dass Anwendungsdaten Zeichen enthalten, die eine UTF-16-Codierungseinheit erfordern. Verwenden Sie **CHAR-MULTIPLIER=4**, wenn Sie erwarten, dass Anwendungsdaten Zeichen enthalten, die zwei UTF-16-Codierungseinheiten erfordern.

Anmerkung: Wenn Sie **CHAR-MULTIPLIER** auf 1 festlegen, schließt dies nicht aus, dass DBCS-Zeichen verwendet werden können. Und wenn Sie den Parameter auf 2 festlegen, schließt dies nicht aus, dass UTF-16-Ersatzzeichenpaare verwendet werden können. Wenn jedoch regelmäßig Breitzichen verwendet werden, werden einige gültige Werte nicht in das zugeordnete Feld passen. Wenn ein größerer Wert für **CHAR-MULTIPLIER** verwendet wird, ist es möglich, mehr Zeichen in dem zugeordneten Feld zu speichern als in der XML gültig sind. Achten Sie darauf, die passenden Bereichseinschränkungen einzuhalten.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Gibt an, wie Zeichendaten mit variabler Länge auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Binärdatentypen mit variabler Länge werden immer entweder einem Container oder einer variierenden Struktur zugeordnet. Wenn Sie diesen Parameter nicht angeben, hängt die Standardzuordnung von der angegebenen Sprache ab. Sie haben folgende Optionen:

- NO** Zeichendaten mit variabler Länge werden als Zeichenfolgen mit fester Länge zugeordnet.
- NULL** Zeichendaten mit variabler Länge werden auf null endenden Zeichenfolgen zugeordnet.
- YES** Zeichendaten mit variabler Länge werden in PL/I einem CHAR VARYING-Datentyp zugeordnet. In den COBOL-, C- und C++-Sprachen werden Zeichendaten mit variabler Länge einer äquivalenten Darstellung zugeordnet, die zwei verwandte Elemente umfasst: die Datenlänge und die Daten.

CHAR-VARYING-LIMIT = { **32767** | *wert* }

Gibt die maximale Größe von Binärdaten und Zeichendaten mit variabler Länge an, die der Sprachstruktur auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Wenn die Zeichen oder Binärdaten größer sind als der in diesem Parameter angegebene Wert, werden sie einem Container zugeordnet, und der Containername wird in der generierten Sprachstruktur verwendet. Der Wert kann zwischen 0 und dem Standardwert 32.767 Bytes liegen.

CHAR-WHITESPACE = **COLLAPSE** | **REPLACE** | **PRESERVE**

Gibt an, wie Leerzeichen in Werten vom Typ Zeichenfolge von CICS verarbeitet werden.

COLLAPSE

Führende, abschließende und eingebettete Leerzeichen werden entfernt und alle Tabulatoren, neuen Zeilen und aufeinanderfolgenden Leerzeichen werden durch einzelne Leerzeichen ersetzt.

REPLACE

Alle Tabulatoren oder neuen Zeilen werden durch die entsprechende Anzahl von Leerzeichen ersetzt.

PRESERVE

Alle Leerzeichen in dem Datenwert werden beibehalten.

Wird der Parameter **CHAR-WHITESPACE** nicht festgelegt, werden die Leerzeichen komprimiert.

Anmerkung: Dieser Parameter gilt nicht für Felder mit dem Format `date-time`, `uri`, `base64Binary` oder `hexBinary`, bei denen Leerzeichen immer komprimiert werden.

CONTID = *wert*

Gibt in einem Service-Provider den Namen des Containers an, der die übergeordnete Datenstruktur enthält, die zur Darstellung einer JSON-Nachricht verwendet wird.

Die Länge des Containers, die CICS an das Zielanwendungsprogramm übergibt, ist die größere der beiden Längen des Anforderungscontainers bzw. des Antwortcontainers.

DATA-SCREENING = { **ENABLED** | **DISABLED** }

Gibt an, ob die von Anwendungen bereitgestellten Daten auf Fehler überprüft werden.

ENABLED

Alle von Anwendungen bereitgestellten Laufzeitdaten, die nicht mit der Sprachstruktur konsistent sind, werden als Fehler behandelt, und es wird die Nachricht DFHPI1010 ausgegeben. Eine Fehlerantwort wird an die Anwendung zurückgegeben.

DISABLED

Werte in den von Anwendungen bereitgestellten Laufzeitdaten, die nicht konsistent mit der Sprachstruktur sind, werden durch Standardwerte ersetzt. Eine Null ersetzt beispielsweise einen fehlerhaften Wert in einem numerischen Feld. Die Nachricht DFHPI1010 wird nicht ausgegeben und es wird eine normale Antwort an die Anwendung zurückgegeben. Dieses Feature kann verwendet werden, um die Fehlerantworten `INVALID_PACKED_DEC` und `INVALID_ZONED_DEC` zu vermeiden, die von nicht initialisierten Ausgabefeldern generiert werden.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Gibt an, ob Daten mit variabler Länge in einer Feldstruktur mit fester Länge toleriert werden:

DISABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, weist CICS die abgeschnittenen Daten zurück und gibt eine Fehlerantwort aus.

ENABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, toleriert CICS die abgeschnittenen Daten und verarbeitet die fehlenden Daten als Nullwerte.

DATETIME = { **PACKED15** | **STRING** }

Gibt an, wie JSON-Datum/Zeit-Elemente der Sprachstruktur zugeordnet werden.

PACKED15

Standardmäßig wird jedes JSON-Datum/Zeit-Element als Zeitmarke verarbeitet und dem Format CICS `ABSTIME` zugeordnet.

STRING

Das JSON-Datum/Zeit-Element wird als Text verarbeitet.

DEFAULT-ARRAY-MAXITEMS = *wert*

Gibt die maximale Arraygrenze an, die angewendet werden soll, wenn im JSON-Schema keine Angabe zur maximalen Anzahl von Vorkommen (maxItems) eingeschlossen ist. Wenn dieser Parameter nicht festgelegt ist, wird kein Maximalwert angewendet. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein. Dieser Parameter kann mit dem Parameter **INLINE-MAXOCCURS-LIMIT** kombiniert werden, um die Zuordnung von JSON-Arrays in die Sprachstrukturen zu beeinflussen.

DEFAULT-CHAR-MAXLENGTH = { 255 | *wert* }

Gibt die Standardarraylänge von Zeichendaten in Zeichen für Zuordnungen an, in deren Web-Service-Beschreibungsdokument keine Länge eingeschlossen ist, wenn die Zuordnungsebene 1.2 oder höher ist. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein.

DEFAULT-FRACTION-DIGITS = { 3 | *wert* }

Gibt die Standardanzahl an Bruchteilstellen für einen JSON-Dezimalschematyp an. Der Standardwert ist 3. Für COBOL liegt der gültige Bereich zwischen 0 und 17, wenn der Parameter **WIDE-COMP3** verwendet wird. Für C oder PLI liegt der gültige Bereich zwischen 0 und 30.

INLINE-MAXOCCURS-LIMIT = { 1 | *wert* }

Gibt auf Basis des JSON-Schemaschlüsselworts maxItems an, ob variabel wiederkehrende Inline-Inhalte verwendet werden. Variabel wiederkehrende Inhalte, die inline zugeordnet sind, werden zusammen mit der generierten Sprachstruktur im aktuellen Container platziert. Der variabel wiederkehrende Inhalt wird in zwei Teilen gespeichert: als Zähler, der die Anzahl der Datenvorkommen speichert, und als Array, in dem alle Datenvorkommen gespeichert werden. Die alternative Zuordnung für variabel wiederkehrenden Inhalt ist eine containerbasierte Zuordnung, die die Anzahl von Datenvorkommen und den Namen des Containers speichert, in dem die Daten platziert werden. Das Speichern der Daten in einem separaten Container wirkt sich auf die Leistung aus, weshalb eine Inline-Zuordnung die bevorzugte Lösung ist.

Der Parameter **INLINE-MAXOCCURS-LIMIT** ist nur ab Zuordnungsebene 2.1 verfügbar. Der Wert von **INLINE-MAXOCCURS-LIMIT** kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein. Der Wert 0 gibt an, dass die Inline-Zuordnung nicht verwendet wird. Der Wert 1 gibt an, dass optionale Elemente inline zugeordnet werden. Wenn die Variable *wert* des Attributs maxOccurs größer als die Variable *wert* von **INLINE-MAXOCCURS-LIMIT** ist, wird eine containerbasierte Zuordnung verwendet. Andernfalls wird eine Inline-Zuordnung verwendet.

Wenn Sie entscheiden, ob variabel wiederkehrende Listen inline zugeordnet werden sollen, ziehen Sie die Länge der einzelnen Elemente von wiederkehrenden Daten in Betracht. Wenn es wenige lange Instanzen gibt, ist die containerbasierte Zuordnung die bevorzugte Lösung. Wenn es viele kurze Instanzen gibt, ist die Inline-Zuordnung geeigneter.

JSON-SCHEMA-REQUEST = *wert*

Dies ist ein obligatorischer Parameter.

Der Wert gibt die UNIX System Services-Position an, an der das JSON-Anforderungsschema gespeichert ist.

JSON-SCHEMA-RESPONSE = *wert*

Dies ist ein obligatorischer Parameter.

Der Wert gibt die UNIX System Services-Position an, an der das JSON-Antwortschema gespeichert ist.

LANG = COBOL|PLI-ENTERPRISE|PLI-OTHER|C|CPP

Gibt die Datenstruktur der höheren Programmiersprache an:

COBOL

COBOL

PLI-ENTERPRISE

Enterprise PL/I

PLI-OTHER

Andere PL/I-Ebene als Enterprise PL/I

C C

CPP C++

LOGFILE = wert

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die DFHJS2LS das Aktivitätenprotokoll und die Traceinformationen schreibt. DFHJS2LS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur.

In der Regel verwenden Sie diese Datei nicht, aber dies kann von der IBM Serviceorganisation angefordert werden, wenn Sie Probleme mit DFHJS2LS feststellen.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 }

Gibt die Zuordnungsebene an, die DFHJS2LS bei der Generierung der Web-Service-Bindungsdatei und der Sprachstruktur verwendet. Sie haben folgende Optionen:

- 1.0** Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 1.1** Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 1.2** Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.0** Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.1** Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.2** Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 3.0** Verwenden Sie diese Zuordnungsebene, um ein JSON-Schema unter Verwendung aller verfügbaren Optionen zu generieren.
- 4.0** Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher, wenn Sie UTF-16 verwenden möchten.
- 4.1** Zur Unterstützung abschneidbarer Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher.

- 4.2 Für zusätzliche Eigenschaften verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.
- 4.3 Zur Unterstützung mehrdimensionaler Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.

Weitere Informationen zu Zuordnungsebenen finden Sie unter Mapping levels for the CICS assistants.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERScores | INTEGER-AS-PIC9 | LESS-DUP-NAMES | UNDERScores-AS-HYPHENS | NO-ARRAY-NAME-INDEXING }

Gibt an, ob das Standardverhalten für die angegebene Zuordnungsebene beim Generieren von Sprachstrukturen überschrieben wird.

SAME-AS-MAPPING-LEVEL

Dieser Parameter generiert Sprachstrukturen auf dieselbe Weise wie die Zuordnungsebene. Dies ist die Standardeinstellung.

HYPHENS-AS-UNDERScores

Nur für PL/I. Dieser Parameter konvertiert alle Bindestriche im JSON-Schema in Unterstriche und nicht in das Zeichen X, um die Lesbarkeit der generierten PL/I-Sprachstrukturen zu verbessern. Weitere Informationen finden Sie unter JSON schema to PL/I mapping. Diese Option ist auf Zuordnungsebene 4.2 automatisch aktiviert.

INTEGER-AS-PIC9

Nur für COBOL und DFHJS2LS. Dieser Parameter generiert Sprachstrukturen, die ganzzahlige Werte aus dem JSON-Schema in Form von Ziffern und nicht von alphanumerischen Zeichen enthalten. Diese Option ist auf Zuordnungsebene 4.0 automatisch aktiviert.

LESS-DUP-NAMES

Dieser Parameter generiert nicht strukturelle Strukturfeldnamen mit `_value` am Ende des Namens, um das direkte Referenzieren des Felds zu aktivieren. Beispiel: In der folgenden PL/I-Sprachstruktur wird, wenn `MAPPING-OVERRIDES=LESS-DUP-NAMES` angegeben ist, an das Ebene-12-Feld 'streetName' das Element `_value` angehängt:

```
09 streetName,
12 streetName CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

Die resultierende Struktur sieht wie folgt aus:

```
09 streetName,
12 streetName_value CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

Diese Option ist auf Zuordnungsebene 4.2 automatisch aktiviert.

UNDERScores-AS-HYPHENS

Nur für COBOL. Dieser Parameter konvertiert alle Unterstriche im JSON-Schema in Bindestriche und nicht in das Zeichen X, um die Lesbarkeit der generierten COBOL-Sprachstrukturen zu verbessern. Im Fall von Feldnamenskollisionen werden die Felder nummeriert, um sicherzustellen, dass sie eindeutig sind. Weitere Informationen finden Sie unter JSON schema to COBOL mapping.

Diese Option ist auf Zuordnungsebene 4.0 automatisch aktiviert.

NO-ARRAY-NAME-INDEXING

Nur für COBOL und Enterprise PL/I. Stellt sicher, dass die Feldnamen in einem Array nur im Rahmen der übergeordneten Struktur eindeutig sind.

MINIMUM-RUNTIME-LEVEL = { **MINIMUM** | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 | **CURRENT** }

Gibt die älteste CICS-Laufzeitumgebung an, in der die Web-Service-Bindungsdatei implementiert werden kann. Wenn Sie eine Version auswählen, die nicht mit den anderen Parametern übereinstimmt, die Sie angegeben haben, erhalten Sie eine Fehlermeldung. Sie haben folgende Optionen:

MINIMUM

Die niedrigste mögliche Laufzeitebene von CICS wird anhand der von Ihnen ausgewählten Parameter automatisch zugewiesen.

- 1.0 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 1.1 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 1.2 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.0 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.1 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.2 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 3.0 Die generierte Web-Service-Bindungsdatei wird in einer Region unter CICS TS 4.1 oder höher implementiert.

Anmerkung: JSON-Unterstützung ist erst ab CICS TS 4.2 verfügbar.

- 4.0 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.0 oder niedriger verwenden. Sie können alle optionalen Parameter auf dieser Ebene verwenden.
- 4.1 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.1 oder niedriger verwenden.
- 4.2 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.2 oder niedriger verwenden.

- 4.3 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.3 oder niedriger verwenden.

CURRENT

Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS-Region auf derselben Laufzeitebene implementiert, die Sie zum Generieren der Web-Service-Bindungsdatei verwenden.

NAME-TRUNCATION = { LEFT | RIGHT }

Gibt an, ob JSON-Namen von links oder von rechts abgeschnitten werden. Der CICS-Web-Service-Assistent schneidet JSON-Namen auf die passende Länge für die angegebene höhere Programmiersprache ab. Standardmäßig werden Namen von rechts abgeschnitten.

PDSCP = wert

Gibt die Codepage an, die in den Mitgliedern einer partitionierten Datei verwendet wird, die in den Parametern **REQMEM** und **RESPMEM** angegeben sind. Dabei ist *wert* eine CCSID-Nummer oder eine Java-Codepagenummer. Wenn dieser Parameter nicht angegeben wird, wird die Codepage von z/OS UNIX System Services verwendet. Beispielsweise können Sie **PDSCP = 037** angeben.

PDSLIB = wert

Gibt den Namen der partitionierten Datei an, die die generierte höhere Programmiersprache enthält. Die Dateimember, die für die Anforderung und Antwort verwendet werden, sind in den Parametern **REQMEM** und **RESPMEM** angegeben.

PGMINT = { CHANNEL | COMMAREA }

Gibt für einen Service-Provider an, wie CICS Daten an das Ziellanwendungsprogramm übergibt.

CHANNEL

CICS verwendet eine Kanalschnittstelle, um Daten an das Ziellanwendungsprogramm zu übergeben.

COMMAREA

CICS verwendet einen Kommunikationsbereich, um Daten an das Ziellanwendungsprogramm zu übergeben.

Wenn das Ziellanwendungsprogramm die Anforderung verarbeitet hat, muss es denselben Mechanismus verwenden, um die Antwort zurückzugeben. Wenn die Anforderung in einem Kommunikationsbereich empfangen wurde, muss die Antwort im Kommunikationsbereich zurückgegeben werden. Wenn die Anforderung in einem Container empfangen wurde, muss die Antwort in einem Container zurückgegeben werden. Die Länge des Kommunikationsbereichs oder des Containers, die CICS an das Ziellanwendungsprogramm zurückgibt, ist größer als die Länge des Anforderungskommunikationsbereichs oder -containers und des Antwortkommunikationsbereichs oder -containers.

PGMNAME = wert

Gibt den Namen einer CICS-PROGRAM-Ressource an.

Wenn DFHJS2LS verwendet wird, um eine Web-Service-Bindungsdatei zu generieren, die in einem Service-Provider verwendet wird, müssen Sie diesen Parameter angeben. Er gibt den Ressourcennamen des Anwendungsprogramms an, das als Web-Service zugänglich gemacht wird.

Wenn DFHJS2LS verwendet wird, um eine Web-Service-Bindungsdatei zu generieren, die in einem Service-Requester verwendet wird, dürfen Sie diesen Parameter nicht angeben.

REQMEM = *wert*

Gibt ein ein bis sechs Zeichen langes Präfix an, das DFHJS2LS verwendet, um die Namen der Member einer partitionierten Datei zu generieren, die die übergeordneten Sprachstrukturen für die Web-Service-Anforderung enthält, wobei es sich um die Eingabedaten für das Anwendungsprogramm handelt.

DFHJS2LS generiert den Namen des Members einer partitionierten Datei durch das Anhängen von '01' an das Präfix.

RESPMEM = *wert*

Gibt ein ein bis sechs Zeichen langes Präfix an, das DFHJS2LS verwendet, um die Namen der Member einer partitionierten Datei zu generieren, die die übergeordneten Sprachstrukturen für die Web-Service-Antwort enthält, wobei es sich um die Ausgabedaten des Anwendungsprogramms handelt.

DFHJS2LS generiert den Namen des Members einer partitionierten Datei durch das Anhängen von '01' an das Präfix.

STRUCTURE = (*anforderung* , *antwort*)

Nur für C und C++: Gibt an, wie die Namen der Anforderungs- und Antwortstrukturen generiert werden.

Die generierten Anforderungs- und Antwortstrukturen werden *anforderung01* und *antwort01* genannt.

Wenn ein Name oder beide Namen fehlen, entsprechen die Namen der Strukturen den Namen der Member der partitionierten Dateien, die aus den angegebenen Parametern **REQMEM** und **RESPMEM** generiert werden.

SYNCONRETURN = { **NO** | **YES** }

Gibt an, ob der ferne Web-Service einen Synchronisationspunkt absetzen kann.

NO Der ferne Web-Service kann keinen Synchronisationspunkt absetzen. Dies ist der Standardwert. Wenn der ferne Web-Service einen Synchronisationspunkt absetzt, schlägt er mit einem ADPL-Abbruch fehl.

YES Der ferne Web-Service kann einen Synchronisationspunkt absetzen. Wenn Sie YES auswählen, wird die ferne Task als separate Arbeitseinheit festgeschrieben, wenn die Steuerung vom fernen Web-Service zurückgegeben wird. Wenn der ferne Web-Service eine wiederherstellbare Ressource aktualisiert und ein Fehler auftritt, nachdem sie zurückgegeben wurde, kann die Aktualisierung dieser Ressource nicht zurückgesetzt werden.

TRANSACTION = *name*

In einem Service-Provider gibt dieser Parameter den ein bis vier Zeichen langen Namen einer Aliastransaktion an, die die Pipeline starten kann. Der Wert dieses Parameters wird verwendet, um das Attribut TRANSACTION der URIMAP-Ressource zu definieren, wenn es automatisch mithilfe des **PIPELINE**-Scanfehls erstellt wird.

Zulässige Zeichen:

A-Z a-z 0-9 \$ @ # _ < >

URI = *wert*

In einem Service-Provider gibt dieser Parameter den relativen URI an, über den ein Client auf den Web-Service zugreift. CICS verwendet den angegebenen

Wert, wenn eine URIMAP-Ressource aus der Web-Service-Bindungsdatei generiert wird, die von DFHJS2LS erstellt wird. Der Parameter gibt die Pfadkomponente des URI an, für den die URIMAP-Definition gilt.

USERID = *id*

In einem Service-Provider gibt dieser Parameter eine ein bis acht Zeichen lange Benutzer-ID an, die von jedem Web-Client verwendet werden kann. Für eine anwendungsgenerierte Antwort oder einen Web-Service wird die Aliastransaktion unter dieser Benutzer-ID angehängt. Der Wert dieses Parameters wird verwendet, um das Attribut USERID der URIMAP-Ressource zu definieren, wenn es automatisch mithilfe des **PIPELINE**-Scanbefehls erstellt wird.

Zulässige Zeichen:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3 = { **FULL** | **NO** | **YES** }

Steuert die maximale Länge der gepackten Dezimalvariablen in der generierten COBOL- oder PL/I-Sprachstruktur.

FULL Für COBOL und PL/I generiert DFHJS2LS ein gepacktes Dezimalfeld, das groß genug ist, um alle gültigen Werte aufzunehmen. Die maximale Länge sind 31 Ziffern. Dies ist die Standardeinstellung.

NO Nur für COBOL. DFHJS2LS begrenzt die Länge der gepackten Dezimalvariablen auf 18 Ziffern, wenn der COBOL-Sprachstrukturtyp COMP-3 generiert wird. Wenn die gepackte Dezimalzahl länger als 18 Ziffern ist, wird die Nachricht DFHPI9022W ausgegeben, um darauf hinzuweisen, dass der angegebene Typ auf höchstens 18 Ziffern begrenzt ist.

YES Nur für COBOL. DFHJS2LS unterstützt die maximale Länge von 31 Ziffern, wenn der COBOL-Sprachstrukturtyp COMP-3 generiert wird.

Anmerkung: Die Optionen NO und YES generieren Felder, die nicht alle gültigen Werte darstellen können. Mit der Option FULL wird dieses Problem behoben. Allerdings lässt die Option FULL auch die Darstellung einiger ungültiger Werte in dem gepackten Dezimalfeld zu. Wenn ein Schema beispielsweise angibt, dass es maximal fünf Ziffern und maximal zwei Nachkommastellen gibt, generiert die Option FULL ein gepacktes Dezimalfeld, das sieben Ziffern zulässt, wodurch Platz für gültige Werte wie 25000 und 999,99 verfügbar ist, aber auch genug Platz für ungültige Werte wie 9999,99. Wenn Sie die Option FULL verwenden, achten Sie darauf, keine ungültigen Werte in Anwendungsdaten zu generieren.

WSBIND = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Web-Service-Bindungsdatei. DFHJS2LS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur. Die Dateierweiterung ist standardmäßig .wsbind.

Weitere Informationen

- Die Benutzer-ID, unter der DFHJS2LS ausgeführt wird, muss für die Verwendung von UNIX System Services konfiguriert sein. Die Benutzer-ID muss über Leseberechtigungen für die z/OS UNIX-Dateistruktur und PDS-Bibliotheken unter CICS und über Schreibberechtigung für die in den Parametern **LOGFILE**, **WSBIND** und **WSDL** angegebenen Verzeichnisse verfügen.
- Der Benutzer-ID muss ausreichend Speicher zugeordnet sein, um Java auszuführen.

- Die JCL weist eine maximale Parameterlänge von 100 Zeichen auf. Dieser Wert kann mithilfe der Anweisung **STDPARM** erhöht werden. Weitere Informationen finden Sie unter z/OS UNIX System Services User's Guide.

Beispiel

```
//JS2LS JOB '
abrechnungsdaten
',
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: Konvertierung eines JSON-Schemas in eine höhere Programmiersprache für REST-konforme Services

Die Prozedur DFHJS2LS generiert aus einem JSON-Schema eine Datenstruktur einer höheren Programmiersprache und eine Web-Service-Bindungsdatei. Sie können DFHJS2LS verwenden, wenn Sie die Erstellung eines Providers REST-konformer JSON-Services vorbereiten. In diesem Thema werden die Jobsteueranweisungen, symbolischen Parameter, Eingabeparameter und ihre Beschreibungen für DFHJS2LS aufgelistet.

Die JCL-Prozedur DFHJS2LS wird in der Datei *HLQ* .XDFHINST installiert, wobei *HLQ* das übergeordnete Qualifikationsmerkmal der Installationsposition von CICS ist.

Jobsteueranweisungen für DFHJS2LS

JOB Startet den Job.

EXEC Gibt den Prozedurnamen an (DFHJS2LS).

INPUT.SYSUT1 DD

Gibt die Eingabe an. Die Eingabeparameter werden in der Regel im Eingabedatenstrom angegeben. Sie können jedoch in einer Datei oder in einem Member einer partitionierten Datei definiert werden.

Symbolische Parameter

Die folgenden symbolischen Parameter sind in DFHJS2LS definiert:

JAVADIR = *pfad*

Gibt den Namen des Java-Verzeichnisses an, das von DFHJS2LS verwendet wird. Der Wert dieses Parameters wird an */usr/lpp/* angehängt, um den vollständigen Pfadnamen */usr/lpp/pfad* zu erstellen.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **JAVADIR** angegeben wurde.

PATHPREF = *präfix*

Gibt ein Präfix an, das den z/OS UNIX-Verzeichnispfad erweitert, der in anderen Parametern verwendet wird, bzw. ' ' (leere Zeichenfolge), wenn kein Präfix verwendet wird.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **PATHPREF** angegeben wurde.

SERVICE = *wert*

Verwenden Sie diesen Parameter nur, wenn sie vom IBM Support dazu aufgefordert werden.

TMPDIR = *tmp-verzeichnis*

Gibt die Position eines Verzeichnisses in z/OS UNIX an, das DFHJS2LS als temporären Arbeitsbereich verwendet. Die Benutzer-ID, unter der der Job ausgeführt wird, muss über Lese- und Schreibberechtigungen für dieses Verzeichnis verfügen.

Der Standardwert ist /tmp.

TMPFILE = *tmp-präfix*

Gibt ein Präfix an, das DFHJS2LS verwendet, um die Namen der temporären Arbeitsbereichsdateien zu erstellen.

Der Standardwert ist JS2LS.

USSDIR = *pfad*

Gibt den Namen des CICS TS-Verzeichnisses im UNIX System Services-Dateisystem an. Der Wert dieses Parameters wird an /usr/lpp/cicsts/ angehängt, um den vollständigen Pfadnamen /usr/lpp/cicsts/*pfad* zu erstellen. Dieser muss als '.' (Punkt) angegeben werden, wenn der Standardwert verwendet wird.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **USSDIR** angegeben wurde.

Der temporäre Arbeitsbereich

DFHJS2LS erstellt die folgenden drei temporären Dateien zur Laufzeit:

```
tmp-verzeichnis / tmp-präfix .in  
tmp-verzeichnis / tmp-präfix .out  
tmp-verzeichnis / tmp-präfix .err
```

Dabei gilt Folgendes:

tmp-verzeichnis ist der Wert, der im Parameter **TMPDIR** angegeben ist.

tmp-präfix ist der Wert, der im Parameter **TMPFILE** angegeben ist.

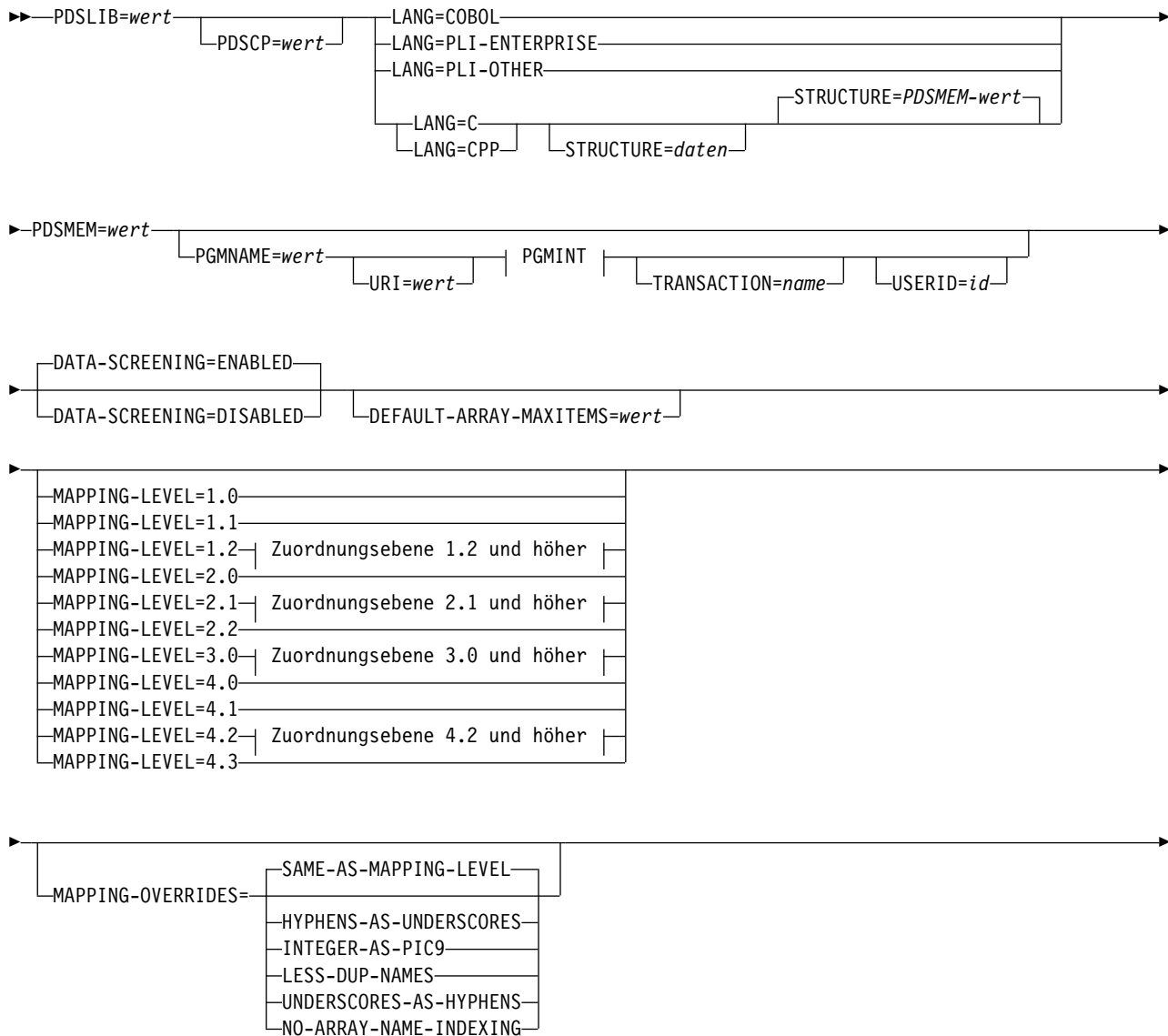
Die Standardnamen für die Dateien lauten wie folgt, wenn **TMPDIR** und **TMPFILE** nicht angegeben sind:

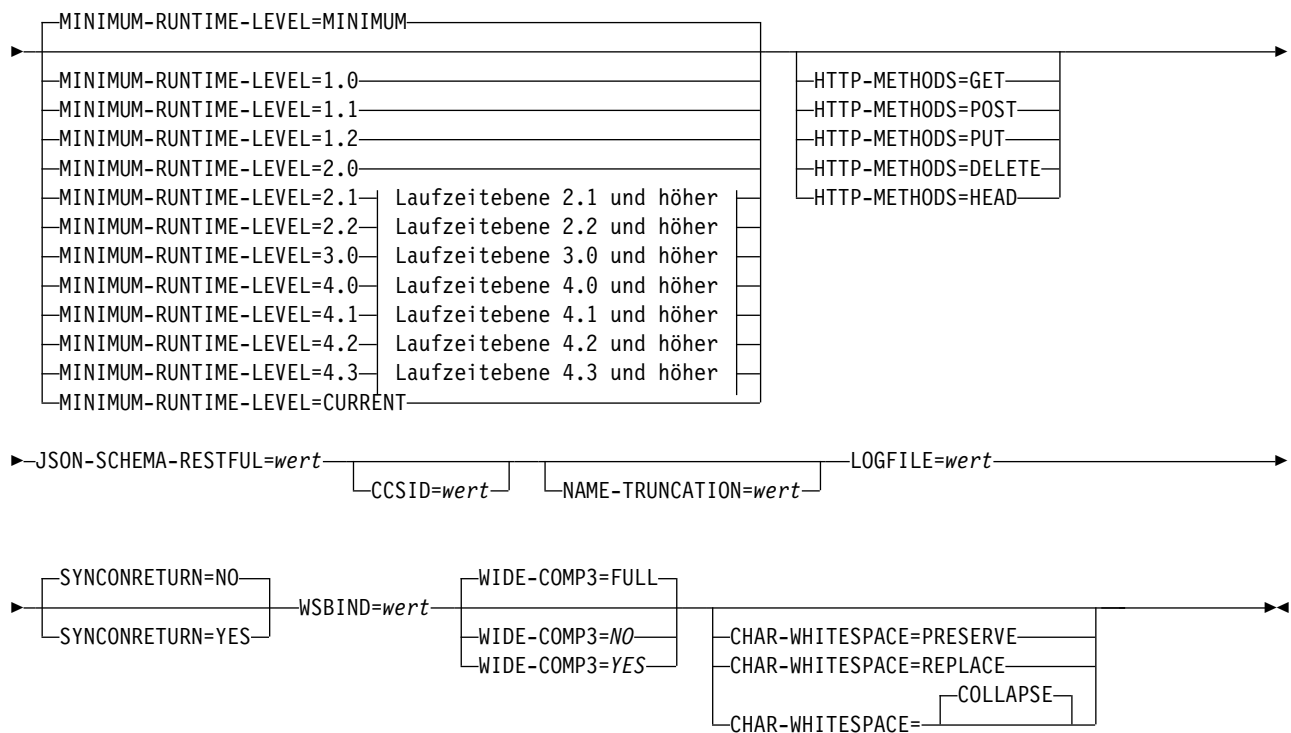
```
/tmp/JS2LS.in  
/tmp/JS2LS.out  
/tmp/JS2LS.err
```

Important: DFHJS2LS sperrt nicht den Zugriff auf die z/OS UNIX-Dateien oder -Dateimember. Wenn folglich zwei oder mehr Instanzen von DFHJS2LS gleichzeitig ausgeführt werden und dieselben temporären Arbeitsbereichsdateien verwenden, kann nicht verhindert werden, dass ein Job die Arbeitsbereichsdateien überschreibt, während ein anderer Job sie verwendet, was zu unvorhersehbaren Fehlern führt.

Deshalb wird empfohlen, eine Namenskonvention und Betriebsprozeduren auszu-
arbeiten, die diese Situation verhindern. Sie können beispielsweise den symboli-
schen Systemparameter **SYSUID** verwenden, um Arbeitsbereichsdateinamen zu ge-
nerieren, die für einen einzelnen Benutzer eindeutig sind. Diese temporären
Dateien werden gelöscht, bevor der Job beendet wird.

Eingabeparameter für DFHJS2LS

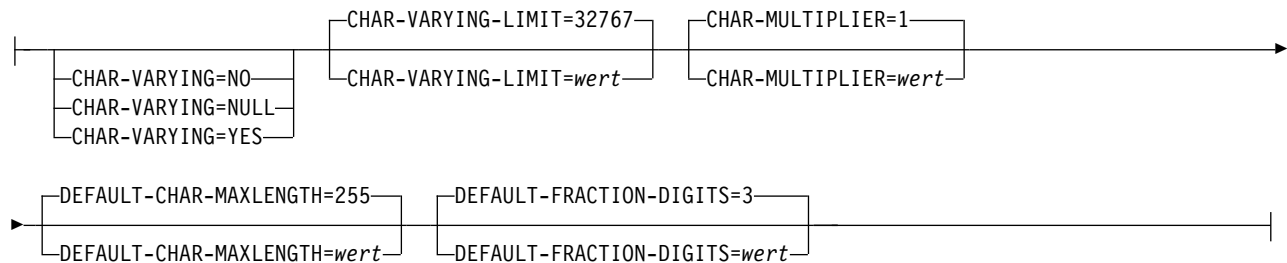




PGMINT:



Zuordnungsebene 1.2 und höher:



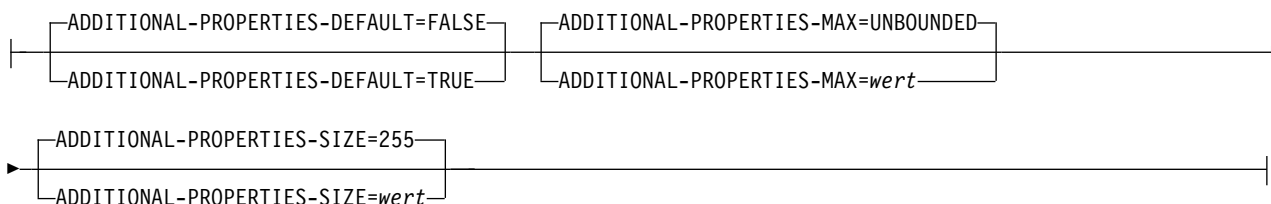
Zuordnungsebene 2.1 und höher:



Zuordnungsebene 3.0 und höher:



Zuordnungsebene 4.2 und höher:



Parameterverwendung

- Sie können die Eingabeparameter in beliebiger Reihenfolge angeben.
- Jeder Parameter muss in einer neuen Zeile eingegeben werden.
- Ein Parameter (und ggf. das Fortsetzungszeichen) darf nicht über Spalte 72 hinausgehen; die Spalten 73-80 müssen Leerzeichen enthalten.
- Wenn ein Parameter zu lang ist, um in eine einzelne Zeile zu passen, verwenden Sie am Ende der Zeile einen Stern (*), um anzugeben, dass der Parameter in der nächsten Zeile fortgesetzt wird. Alles, einschließlich der Leerzeichen, vor dem Stern wird als Teil des Parameters betrachtet. Beispiel:

```
WSBIND=wsbinddir*  
      /app1
```

ist äquivalent zu

```
WSBIND=wsbinddir/app1
```

- Ein #-Zeichen an der ersten Zeichenposition der Zeile ist ein Kommentarzeichen. Die Zeile wird ignoriert.
- Ein Komma an der letzten Zeichenposition der Zeile ist ein optionales Zeilen-trennzeichen und wird ignoriert.

Parameterbeschreibungen

ADDITIONAL-PROPERTIES-DEFAULT = { **true** | **false** }

Gibt an, ob JSON-Schemaobjekte, die zusätzliche Eigenschaften nicht explizit unterstützen, als unterstützende Elemente interpretiert werden oder nicht. Zusätzliche JSON-Eigenschaften sind alle Eigenschaften in einem JSON-Objekt, die im JSON-Schema nicht vordefiniert sind. Diese Eigenschaften werden in der Regel vom Datenumsetzungsmechanismus als unerwartete Zusatzdaten zurückgewiesen. Wenn **ADDITIONAL-PROPERTIES-DEFAULT** auf TRUE gesetzt ist, oder wenn das JSON-Schema explizit `additionalProperties:true` für ein Objekt festlegt, wird ein Bereich für solche Werte in den generierten Copybooks reserviert. Anwendungen können mit diesen Werten interagieren, indem sie die zugeordneten Felder in den Copybooks verwenden.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

ADDITIONAL-PROPERTIES-MAX = { **0-20** | **UNBOUNDED** }

Gibt an, wie viele zusätzliche Eigenschaften für ein entsprechendes JSON-Objekt unterstützt werden. Siehe **ADDITIONAL-PROPERTIES-DEFAULT**. Die generierten

Copybooks enthalten Strukturen, die sich für die Adressierung aller zusätzlichen Eigenschaften eignen. Standardmäßig gibt es keine maximale Einschränkung für die Anzahl der unterstützten Eigenschaften. Die Copybooks werden in ähnlicher Weise wie Arrays ohne Einschränkungen generiert und verwenden Container. Mit diesem Parameter kann eine maximale Einschränkung angewendet werden, die in Kombination mit dem Parameter **INLINE-MAXOCCURS-LIMIT** dafür sorgt, dass ein Array mit fester Länge für die maximale Anzahl von Eigenschaften zugeordnet werden kann, wodurch keine Container erforderlich sind.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

ADDITIONAL-PROPERTIES-SIZE = { **16-32767** | **255** }

Gibt die maximale Größe für jede der zusätzlichen JSON-Eigenschaften an. Wenn ein JSON-Objekt zusätzliche Eigenschaften unterstützt, wie in **ADDITIONAL-PROPERTIES-DEFAULT** definiert, dann haben die generierten Copybooks Bindungen, um Eigenschaften bis zu der von **ADDITIONAL-PROPERTIES-MAX** angegebenen Anzahl zu unterstützen. Standardmäßig beträgt der für jede zusätzliche Eigenschaft unterstützte Maximalwert 255 Zeichen. Ein Feld dieser Größe wird in den erstellten Copybooks generiert. Diese Größe kann durch Festlegen des Parameters **ADDITIONAL-PROPERTIES-SIZE** angepasst werden. So wird beispielsweise ein JSON-Objekt verarbeitet, das folgende Eigenschaft enthält:

```
"example": { "notes": "this extra property was not defined in the JSON Schema"
}
```

Wenn die Copybooks generiert wurden, um zusätzliche Eigenschaften zu unterstützen, wird der gesamte Wert zur Verarbeitung an die Anwendung übergeben. Der Wert beginnt mit dem führenden Anführungszeichen vor dem Eigenschaftsschlüssel und endet mit der abschließenden rechten geschweiften Klammer im Eigenschaftswert. In diesem Beispiel sind es etwa 100 Zeichen. Der Wert für **ADDITIONAL-PROPERTIES-SIZE** muss groß genug sein, um den größtmöglichen Wert enthalten zu können. Wenn der zugeordnete Puffer für den verarbeiteten Wert zu klein ist, wird eine Fehlerantwort generiert.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

CCSID = *wert*

Gibt die CCSID an, die zur Laufzeit verwendet wird, um Zeichendaten in der Anwendungsdatenstruktur zu codieren. Der Wert dieses Parameters überschreibt den Wert des Systeminitialisierungsparameters **LOCALCCSID**. *wert* muss eine EBCDIC-CCSID sein, die von Java und z/OS Unicode Services User's Guide and Reference unterstützt wird. Wenn Sie diesen Parameter nicht angeben, wird die Anwendungsdatenstruktur unter Verwendung der im Systeminitialisierungsparameter angegebenen CCSID codiert.

CHAR-MULTIPLIER = { **1** | *wert* }

Gibt die Anzahl von Bytes an, die für die einzelnen Zeichen auf Zuordnungsebene 1.2 oder höher zulässig sein sollen. Der Wert (*wert*) dieses Parameters kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein. Alle nicht numerischen, zeichenbasierten Zuordnungen unterliegen diesem Multiplikator. Binäre, numerische, in Zonen eingeteilte und gepackte Dezimalfelder unterliegen diesem Multiplikator nicht.

Dieser Parameter kann nützlich sein, wenn Sie beispielsweise planen, DBCS-Zeichen zu verwenden, bei denen Sie sich für einen Multiplikator von 3 entscheiden können, um zur Laufzeit Platz zu lassen für mögliche SO- und SI-Zeichen (shift-out = Umschalttaste nicht gedrückt, shift-in = Umschalttaste gedrückt) um jedes Doppelbytezeichen herum.

Wenn Sie **CCSID=1200** (d. h. UTF-16) angeben, sind 2 oder 4 die einzig gültigen Werte für **CHAR-MULTIPLIER**. Wenn Sie UTF-16 verwenden, ist der Standardwert 2. Verwenden Sie **CHAR-MULTIPLIER=2**, wenn Sie erwarten, dass Anwendungsdaten Zeichen enthalten, die eine UTF-16-Codierungseinheit erfordern. Verwenden Sie **CHAR-MULTIPLIER=4**, wenn Sie erwarten, dass Anwendungsdaten Zeichen enthalten, die zwei UTF-16-Codierungseinheiten erfordern.

Anmerkung: Wenn Sie **CHAR-MULTIPLIER** auf 1 festlegen, schließt dies nicht aus, dass DBCS-Zeichen verwendet werden können. Und wenn Sie den Parameter auf 2 festlegen, schließt dies nicht aus, dass UTF-16-Ersatzzeichenpaare verwendet werden können. Wenn jedoch regelmäßig Breitzichen verwendet werden, werden einige gültige Werte nicht in das zugeordnete Feld passen. Wenn ein größerer Wert für **CHAR-MULTIPLIER** verwendet wird, ist es möglich, mehr Zeichen in dem zugeordneten Feld zu speichern als in der XML gültig sind. Achten Sie darauf, die passenden Bereichseinschränkungen einzuhalten.

CHAR-VARYING = NO | NULL | YES

Gibt an, wie Zeichendaten mit variabler Länge auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Binärdatentypen mit variabler Länge werden immer entweder einem Container oder einer variierenden Struktur zugeordnet. Wenn Sie diesen Parameter nicht angeben, hängt die Standardzuordnung von der angegebenen Sprache ab. Sie haben folgende Optionen:

- NO** Zeichendaten mit variabler Länge werden als Zeichenfolgen mit fester Länge zugeordnet.
- NULL** Zeichendaten mit variabler Länge werden auf null endenden Zeichenfolgen zugeordnet.
- YES** Zeichendaten mit variabler Länge werden in PL/I einem CHAR VARYING-Datentyp zugeordnet. In den COBOL-, C- und C++-Sprachen werden Zeichendaten mit variabler Länge einer äquivalenten Darstellung zugeordnet, die zwei verwandte Elemente umfasst: die Datenlänge und die Daten.

CHAR-VARYING-LIMIT = 32767 | wert

Gibt die maximale Größe von Binärdaten und Zeichendaten mit variabler Länge an, die der Sprachstruktur auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Wenn die Zeichen oder Binärdaten größer sind als der in diesem Parameter angegebene Wert, werden sie einem Container zugeordnet, und der Containername wird in der generierten Sprachstruktur verwendet. Der Wert kann zwischen 0 und dem Standardwert 32.767 Bytes liegen.

CHAR-WHITESPACE = COLLAPSE | REPLACE | PRESERVE

Gibt an, wie Leerzeichen in Werten vom Typ Zeichenfolge von CICS verarbeitet werden.

COLLAPSE

Führende, abschließende und eingebettete Leerzeichen werden entfernt und alle Tabulatoren, neuen Zeilen und aufeinanderfolgenden Leerzeichen werden durch einzelne Leerzeichen ersetzt.

REPLACE

Alle Tabulatoren oder neuen Zeilen werden durch die entsprechende Anzahl von Leerzeichen ersetzt.

PRESERVE

Alle Leerzeichen in dem Datenwert werden beibehalten.

Wenn der Parameter **CHAR-WHITESPACE** nicht festgelegt ist, werden Leerzeichen komprimiert.

Anmerkung: Dieser Parameter gilt nicht für Felder mit dem Format `date-time`, `uri`, `base64Binary` oder `hexBinary`, bei denen Leerzeichen immer komprimiert werden.

CONTID = *wert*

Gibt in einem Service-Provider den Namen des Containers an, der die übergeordnete Datenstruktur enthält, die zur Darstellung einer JSON-Nachricht verwendet wird.

Die Länge des Containers, die CICS an das Ziellanwendungsprogramm übergibt, ist die größere der beiden Längen des Anforderungscontainers bzw. des Antwortcontainers.

DATA-SCREENING = { **ENABLED** | **DISABLED** }

Gibt an, ob die von Anwendungen bereitgestellten Daten auf Fehler überprüft werden.

ENABLED

Alle von Anwendungen bereitgestellten Laufzeitdaten, die nicht mit der Sprachstruktur konsistent sind, werden als Fehler behandelt, und es wird die Nachricht DFHPI1010 ausgegeben. Eine Fehlerantwort wird an die Anwendung zurückgegeben.

DISABLED

Werte in den von Anwendungen bereitgestellten Laufzeitdaten, die nicht konsistent mit der Sprachstruktur sind, werden durch Standardwerte ersetzt. Eine Null ersetzt beispielsweise einen fehlerhaften Wert in einem numerischen Feld. Die Nachricht DFHPI1010 wird nicht ausgegeben und es wird eine normale Antwort an die Anwendung zurückgegeben. Dieses Feature kann verwendet werden, um die Fehlerantworten `INVALID_PACKED_DEC` und `INVALID_ZONED_DEC` zu vermeiden, die von nicht initialisierten Ausgabefeldern generiert werden.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Gibt an, ob Daten mit variabler Länge in einer Feldstruktur mit fester Länge toleriert werden:

DISABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, weist CICS die abgeschnittenen Daten zurück und gibt eine Fehlerantwort aus.

ENABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, toleriert CICS die abgeschnittenen Daten und verarbeitet die fehlenden Daten als Nullwerte.

DATETIME = **PACKED15** | **STRING**

Gibt an, wie JSON-Datum/Zeit-Elemente der Sprachstruktur zugeordnet werden.

PACKED15

Standardmäßig wird jedes JSON-Datum/Zeit-Element als Zeitmarke verarbeitet und dem Format CICS `ABSTIME` zugeordnet.

STRING

Das JSON-Datum/Zeit-Element wird als Text verarbeitet.

DEFAULT-ARRAY-MAXITEMS = *wert*

Gibt die maximale Arraygrenze an, die angewendet werden soll, wenn im JSON-Schema keine Angabe zur maximalen Anzahl von Vorkommen (maxItems) eingeschlossen ist. Wenn dieser Parameter nicht festgelegt ist, wird kein Maximalwert angewendet. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein. Dieser Parameter kann mit dem Parameter **INLINE-MAXOCCURS-LIMIT** kombiniert werden, um die Zuordnung von JSON-Arrays in die Sprachstrukturen zu beeinflussen.

DEFAULT-CHAR-MAXLENGTH = 255 | *wert*

Gibt die Standardarraylänge von Zeichendaten in Zeichen für Zuordnungen an, in deren Web-Service-Beschreibungsdokument keine Länge eingeschlossen ist, wenn die Zuordnungsebene 1.2 oder höher ist. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein.

DEFAULT-FRACTION-DIGITS = { 3 | *wert* }

Gibt die Standardanzahl an Bruchteilstellen für einen JSON-Dezimalschematyp an. Der Standardwert ist 3. Für COBOL liegt der gültige Bereich zwischen 0 und 17, wenn der Parameter **WIDE-COMP3** verwendet wird. Für C oder PLI liegt der gültige Bereich zwischen 0 und 30.

HTTP-METHODS = { *GET* | *POST* | *PUT* | *DELETE* | *HEAD* }, { *GET* | *POST* | *PUT* | *DELETE* | *HEAD* }, *

Dies ist ein optionaler Parameter.

Standardmäßig sind GET, POST, PUT und DELETE festgelegt, wodurch DFHJS2LS weiß, dass die Anwendung diese vier REST-konformen Hauptoperationen unterstützt.

Wird ein Wert angegeben, erstellt DFHJS2LS eine WSBind-Datei, in der nur die explizit angegebenen HTTP-Methoden akzeptiert sind.

Wenn eine Anwendung die HEAD-Methode implementieren möchte, muss sie diese konkret angeben. Standardmäßig geht DFHJS2LS davon aus, dass die Anwendung eingehende HTTP-HEAD-Methoden nicht unterstützt.

Wenn ein JSON-Client versucht, eine nicht unterstützte HTTP-Methode zu verwenden, gibt CICS eine HTTP 405-Antwort zurück.

INLINE-MAXOCCURS-LIMIT = 1 | *wert*

Gibt auf Basis des JSON-Schemaschlüsselworts maxItems an, ob variabel wiederkehrende Inline-Inhalte verwendet werden. Variabel wiederkehrende Inhalte, die inline zugeordnet sind, werden zusammen mit der generierten Sprachstruktur im aktuellen Container platziert. Der variabel wiederkehrende Inhalt wird in zwei Teilen gespeichert: als Zähler, der die Anzahl der Datenvorkommen speichert, und als Array, in dem alle Datenvorkommen gespeichert werden. Die alternative Zuordnung für variabel wiederkehrenden Inhalt ist eine containerbasierte Zuordnung, die die Anzahl von Datenvorkommen und den Namen des Containers speichert, in dem die Daten platziert werden. Das Speichern der Daten in einem separaten Container wirkt sich auf die Leistung aus, weshalb eine Inline-Zuordnung die bevorzugte Lösung ist.

Der Parameter **INLINE-MAXOCCURS-LIMIT** ist nur ab Zuordnungsebene 2.1 verfügbar. Der Wert von **INLINE-MAXOCCURS-LIMIT** kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein. Der Wert 0 gibt an, dass die Inline-Zuordnung nicht verwendet wird. Der Wert 1 gibt an, dass optionale Elemente inline zugeordnet werden. Wenn die Variable *wert* des Attributs maxOccurs größer als die Variable *wert* von **INLINE-MAXOCCURS-LIMIT** ist, wird eine containerbasierte Zuordnung verwendet. Andernfalls wird eine Inline-Zuordnung verwendet.

Wenn Sie entscheiden, ob variabel wiederkehrende Listen inline zugeordnet werden sollen, ziehen Sie die Länge der einzelnen Elemente von wiederkehren-

den Daten in Betracht. Wenn es wenige lange Instanzen gibt, ist die containerbasierte Zuordnung die bevorzugte Lösung. Wenn es viele kurze Instanzen gibt, ist die Inline-Zuordnung geeigneter.

JSON-SCHEMA-RESTFUL = *wert*

Dies ist ein obligatorischer Parameter.

Der Wert gibt die UNIX System Services-Position an, an der das JSON-Antwortschema gespeichert ist.

LANG = **COBOL** | **PLI-ENTERPRISE** | **PLI-OTHER** | **C** | **CPP**

Gibt die Datenstruktur der höheren Programmiersprache an:

COBOL

COBOL

PLI-ENTERPRISE

Enterprise PL/I

PLI-OTHER

Andere PL/I-Ebene als Enterprise PL/I

C C

CPP C++

LOGFILE = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die DFHJS2LS das Aktivitätenprotokoll und die Traceinformationen schreibt. DFHJS2LS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur.

In der Regel verwenden Sie diese Datei nicht, aber dies kann von der IBM Serviceorganisation angefordert werden, wenn Sie Probleme mit DFHJS2LS feststellen.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 }

Gibt die Zuordnungsebene an, die DFHJS2LS bei der Generierung der Web-Service-Bindungsdatei und der Sprachstruktur verwendet. Sie haben folgende Optionen:

- 1.0 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 1.1 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 1.2 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.0 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.1 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.
- 2.2 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

- 3.0 Verwenden Sie diese Zuordnungsebene, um ein JSON-Schema unter Verwendung aller verfügbaren Optionen zu generieren.
- 4.0 Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher, wenn Sie UTF-16 verwenden möchten.
- 4.1 Zur Unterstützung abschneidbarer Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher.
- 4.2 Für zusätzliche Eigenschaften verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.
- 4.3 Zur Unterstützung mehrdimensionaler Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.

Weitere Informationen zu Zuordnungsebenen finden Sie unter Mapping levels for the CICS JSON assistants.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERSCORES | INTEGER-AS-PIC9 | LESS-DUP-NAMES | UNDERSCORES-AS-HYPHENS | NO-ARRAY-NAME-INDEXING }

Gibt an, ob das Standardverhalten für die angegebene Zuordnungsebene beim Generieren von Sprachstrukturen überschrieben wird.

SAME-AS-MAPPING-LEVEL

Dieser Parameter generiert Sprachstrukturen auf dieselbe Weise wie die Zuordnungsebene. Dies ist die Standardeinstellung.

HYPHENS-AS-UNDERSCORES

Nur für PL/I. Dieser Parameter konvertiert alle Bindestriche im JSON-Schema in Unterstriche und nicht in das Zeichen X, um die Lesbarkeit der generierten PL/I-Sprachstrukturen zu verbessern. Weitere Informationen finden Sie unter JSON schema to PL/I mapping. Diese Option ist auf Zuordnungsebene 4.2 automatisch aktiviert.

INTEGER-AS-PIC9

Nur für COBOL und DFHJS2LS. Dieser Parameter generiert Sprachstrukturen, die ganzzahlige Werte aus dem JSON-Schema in Form von Ziffern und nicht von alphanumerischen Zeichen enthalten. Diese Option ist auf Zuordnungsebene 4.0 automatisch aktiviert.

LESS-DUP-NAMES

Dieser Parameter generiert nicht strukturelle Strukturfeldnamen mit `_value` am Ende des Namens, um das direkte Referenzieren des Felds zu aktivieren. Beispiel: In der folgenden PL/I-Sprachstruktur wird, wenn `MAPPING-OVERRIDES=LESS-DUP-NAMES` angegeben ist, an das Ebene-12-Feld 'streetName' das Element `_value` angehängt:

```
09 streetName,
12 streetName CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

Die resultierende Struktur sieht wie folgt aus:

```
09 streetName,
12 streetName_value CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

Diese Option ist auf Zuordnungsebene 4.2 automatisch aktiviert.

UNDERScores-AS-HYPHENS

Nur für COBOL. Dieser Parameter konvertiert alle Unterstriche im JSON-Schema in Bindestriche und nicht in das Zeichen X, um die Lesbarkeit der generierten COBOL-Sprachstrukturen zu verbessern. Im Fall von Feldnamenskollisionen werden die Felder nummeriert, um sicherzustellen, dass sie eindeutig sind. Weitere Informationen finden Sie unter JSON schema to COBOL mapping.

Diese Option ist auf Zuordnungsebene 4.0 automatisch aktiviert.

NO-ARRAY-NAME-INDEXING

Nur für COBOL und Enterprise PL/I. Stellt sicher, dass die Feldnamen in einem Array nur im Rahmen der übergeordneten Struktur eindeutig sind.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 | CURRENT }

Gibt die älteste CICS-Laufzeitumgebung an, in der die Web-Service-Bindungsdatei implementiert werden kann. Wenn Sie eine Version auswählen, die nicht mit den anderen Parametern übereinstimmt, die Sie angegeben haben, erhalten Sie eine Fehlermeldung. Sie haben folgende Optionen:

MINIMUM

Die niedrigste mögliche Laufzeitebene von CICS wird anhand der von Ihnen ausgewählten Parameter automatisch zugewiesen.

1.0 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

1.1 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

1.2 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

2.0 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

2.1 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

2.2 Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

3.0 Die generierte Web-Service-Bindungsdatei wird in einer Region unter CICS TS 4.1 oder höher implementiert.

Anmerkung: JSON-Unterstützung ist erst ab CICS TS 4.2 verfügbar.

4.0 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.0 oder niedriger verwenden. Sie können alle optionalen Parameter auf dieser Ebene verwenden.

4.1 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Re-

gion unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.1 oder niedriger verwenden.

4.2 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.2 oder niedriger verwenden.

4.3 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.3 oder niedriger verwenden.

CURRENT

Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS-Region auf derselben Laufzeitebene implementiert, die Sie zum Generieren der Web-Service-Bindungsdatei verwenden.

NAME-TRUNCATION = { LEFT | RIGHT }

Gibt an, ob JSON-Namen von links oder von rechts abgeschnitten werden. Der CICS-Web-Service-Assistent schneidet JSON-Namen auf die passende Länge für die angegebene höhere Programmiersprache ab. Standardmäßig werden Namen von rechts abgeschnitten.

PDSCP = wert

Gibt die Codepage an, die in den Mitgliedern einer partitionierten Datei verwendet wird, die in dem Parameter **PDSMEM** angegeben sind. Dabei ist *wert* eine CC-SID-Nummer oder eine Java-Codepagenummer. Wenn dieser Parameter nicht angegeben ist, wird die z/OS UNIX System Services-Codepage verwendet. Beispielsweise können Sie **PDSCP = 037** angeben.

PDSLIB = wert

Gibt den Namen der partitionierten Datei an, die die generierte höhere Programmiersprache enthält. Die Dateimember, die für die Anforderung und Antwort verwendet werden, sind im Parameter **PDSMEM** angegeben.

PDSMEM = wert

Gibt das ein bis sechs Zeichen lange Präfix an, das DFHJS2LS verwendet, um den Namen des Members der partitionierten Datei zu generieren, die die Struktur der höheren Programmiersprache enthalten wird.

DFHJS2LS generiert ein Member einer partitionierten Datei, indem 01 an das Präfix angehängt wird.

PGMINT = CHANNEL | COMMAREA

Gibt für einen Service-Provider an, wie CICS Daten an das Ziellanwendungsprogramm übergibt.

CHANNEL

CICS verwendet eine Kanalschnittstelle, um Daten an das Ziellanwendungsprogramm zu übergeben.

COMMAREA

CICS verwendet einen Kommunikationsbereich, um Daten an das Ziellanwendungsprogramm zu übergeben.

Wenn das Ziellanwendungsprogramm die Anforderung verarbeitet hat, muss es denselben Mechanismus verwenden, um die Antwort zurückzugeben. Wenn die Anforderung in einem Kommunikationsbereich empfangen wurde, muss die Antwort im Kommunikationsbereich zurückgegeben werden. Wenn die Anforderung in einem Container empfangen wurde, muss die Antwort in einem

Container zurückgegeben werden. Die Länge des Kommunikationsbereichs oder des Containers, die CICS an das Zielanwendungsprogramm zurückgibt, ist größer als die Länge des Anforderungskommunikationsbereichs oder -containers und des Antwortkommunikationsbereichs oder -containers.

PGMNAME = *wert*

Gibt den Namen einer CICS-PROGRAM-Ressource an.

Wenn DFHJS2LS verwendet wird, um eine Web-Service-Bindungsdatei zu generieren, die in einem Service-Provider verwendet wird, müssen Sie diesen Parameter angeben. Er gibt den Ressourcennamen des Anwendungsprogramms an, das als Web-Service zugänglich gemacht wird.

Wenn DFHJS2LS verwendet wird, um eine Web-Service-Bindungsdatei zu generieren, die in einem Service-Requester verwendet wird, dürfen Sie diesen Parameter nicht angeben.

STRUCTURE = *name*

Nur für C und C++: Gibt an, wie der Name der Struktur generiert wird.

Die generierte Struktur wird *name 01* genannt.

Wird der Name weggelassen, entspricht der Name der Struktur dem Namen des Members der partitionierten Datei, der aus dem von Ihnen angegebenen Parameter **PDSMEM** generiert wurde.

SYNCONRETURN = { **NO** | **YES** }

Gibt an, ob der ferne Web-Service einen Synchronisationspunkt absetzen kann.

NO Der ferne Web-Service kann keinen Synchronisationspunkt absetzen. Dies ist der Standardwert. Wenn der ferne Web-Service einen Synchronisationspunkt absetzt, schlägt er mit einem ADPL-Abbruch fehl.

YES Der ferne Web-Service kann einen Synchronisationspunkt absetzen. Wenn Sie YES auswählen, wird die ferne Task als separate Arbeitseinheit festgeschrieben, wenn die Steuerung vom fernen Web-Service zurückgegeben wird. Wenn der ferne Web-Service eine wiederherstellbare Ressource aktualisiert und ein Fehler auftritt, nachdem sie zurückgegeben wurde, kann die Aktualisierung dieser Ressource nicht zurückgesetzt werden.

TRANSACTION = *name*

In einem Service-Provider gibt dieser Parameter den ein bis vier Zeichen langen Namen einer Aliastransaktion an, die die Pipeline starten kann. Der Wert dieses Parameters wird verwendet, um das Attribut **TRANSACTION** der URIMAP-Ressource zu definieren, wenn es automatisch mithilfe des **PIPELINE**-Scanfehls erstellt wird.

Zulässige Zeichen:

A-Z a-z 0-9 \$ @ # _ < >

URI = *wert*

In einem Service-Provider gibt dieser Parameter den relativen URI an, über den ein Client auf den Web-Service zugreift. CICS verwendet den angegebenen Wert, wenn eine URIMAP-Ressource aus der Web-Service-Bindungsdatei generiert wird, die von DFHJS2LS erstellt wird. Der Parameter gibt die Pfadkomponente des URI an, für den die URIMAP-Definition gilt. Wenn Sie das Platzhalterzeichen * am Ende eines URI verwenden, muss auf den URI-Wert ein Komma folgen.

USERID = *id*

In einem Service-Provider gibt dieser Parameter eine ein bis acht Zeichen lange Benutzer-ID an, die von jedem Web-Client verwendet werden kann. Für eine anwendungsgenerierte Antwort oder einen Web-Service wird die Aliastransaktion unter dieser Benutzer-ID angehängt. Der Wert dieses Parameters wird verwendet, um das Attribut USERID der URIMAP-Ressource zu definieren, wenn es automatisch mithilfe des **PIPELINE**-Scanbefehls erstellt wird.

Zulässige Zeichen:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3 = { **FULL** | **NO** | **YES** }

Steuert die maximale Länge der gepackten Dezimalvariablen in der generierten COBOL- oder PL/I-Sprachstruktur.

FULL Für COBOL und PL/I generiert DFHJS2LS ein gepacktes Dezimalfeld, das groß genug ist, um alle gültigen Werte aufzunehmen. Die maximale Länge sind 31 Ziffern. Dies ist die Standardeinstellung.

NO Nur für COBOL. DFHJS2LS begrenzt die Länge der gepackten Dezimalvariablen auf 18 Ziffern, wenn der COBOL-Sprachstrukturtyp COMP-3 generiert wird. Wenn die gepackte Dezimalzahl länger als 18 Ziffern ist, wird die Nachricht DFHPI9022W ausgegeben, um darauf hinzuweisen, dass der angegebene Typ auf höchstens 18 Ziffern begrenzt ist.

YES Nur für COBOL. DFHJS2LS unterstützt die maximale Länge von 31 Ziffern, wenn der COBOL-Sprachstrukturtyp COMP-3 generiert wird.

Anmerkung: Die Optionen NO und YES generieren Felder, die nicht alle gültigen Werte darstellen können. Mit der Option FULL wird dieses Problem behoben. Allerdings lässt die Option FULL auch die Darstellung einiger ungültiger Werte in dem gepackten Dezimalfeld zu. Wenn ein Schema beispielsweise angibt, dass es maximal fünf Ziffern und maximal zwei Nachkommastellen gibt, generiert die Option FULL ein gepacktes Dezimalfeld, das sieben Ziffern zulässt, wodurch Platz für gültige Werte wie 25000 und 999,99 verfügbar ist, aber auch genug Platz für ungültige Werte wie 9999,99. Wenn Sie die Option FULL verwenden, achten Sie darauf, keine ungültigen Werte in Anwendungsdaten zu generieren.

WSBIND = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Web-Service-Bindungsdatei. DFHJS2LS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur. Die Dateierweiterung ist standardmäßig **.wsbind**.

Weitere Informationen

- Die Benutzer-ID, unter der DFHJS2LS ausgeführt wird, muss für die Verwendung von UNIX System Services konfiguriert sein. Die Benutzer-ID muss über Leseberechtigungen für die z/OS UNIX-Dateistruktur und PDS-Bibliotheken unter CICS und über Schreibberechtigung für die in den Parametern **LOGFILE**, **WSBIND** und **WSDL** angegebenen Verzeichnisse verfügen.
- Der Benutzer-ID muss ausreichend Speicher zugeordnet sein, um Java auszuführen.
- Die JCL weist eine maximale Parameterlänge von 100 Zeichen auf. Dieser Wert kann mithilfe der Anweisung **STDPARM** erhöht werden. Weitere Informationen finden Sie unter z/OS UNIX System Services User's Guide.

Beispiel

```
//JS2LS JOB '  
abrechnungsdaten  
'  
,  
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  
/*
```

JSON-Service-Provider-Anwendung unter Verwendung des JSON-Assistenten erstellen

Sie können eine Service-Provider-Anwendung aus einem JSON-Schema erstellen, das mit dem JSON-Schema Version 4 (Entwurf) kompatibel ist, oder aus einer übergeordneten Sprachdatenstruktur. Der CICS-JSON-Assistent hilft Ihnen bei der Implementierung Ihrer CICS-Anwendungen in einer Service-Provider-Einstellung.

About this task

Wenn Sie den Assistenten verwenden, um eine CICS-Anwendung als Service-Provider zu implementieren, haben Sie zwei Optionen:

- Beginnen Sie mit einem JSON-Schema und verwenden Sie den Assistenten, um die Sprachdatenstrukturen zu generieren.
Verwenden Sie diese Option, wenn Sie einen Service-Provider implementieren möchten, der mit einer vorhandenen Web-Service-Beschreibung konform ist.
- Beginnen Sie mit den Sprachdatenstrukturen und verwenden Sie den Assistenten, um das JSON-Schema zu generieren.
Verwenden Sie diese Option, wenn Sie ein vorhandenes Programm als JSON-Service verfügbar machen möchten.

Service-Provider-Anwendung aus einem JSON-Schema erstellen

Mit dem CICS JSON-Assistenten können Sie eine Service-Provider-Anwendung aus einem JSON-Schema erstellen.

Before you begin

Bevor Sie eine Service-Provider-Anwendung erstellen können, müssen die folgenden Bedingungen erfüllt sein:

- Ihre Web-Service-Beschreibung muss sich in einer UNIX-Datei unter z/OS befinden und Sie müssen eine geeignete Pipeline im Providermodus in der CICS-Region erstellen.
- Sie müssen in OMVS die Benutzer-ID definieren, unter der DFHJS2LS ausgeführt wird.

- Die Benutzer-ID muss über Leseberechtigungen für z/OS UNIX- und PDS-Bibliotheken und über Schreibberechtigungen für die Verzeichnisse verfügen, die in den Parametern **LOGFILE**, **WSBIND**, **JSON-SCHEMA-REQUEST**, **JSON-SCHEMA-RESPONSE** und **JSON-SCHEMA-RESTFUL** angegeben sind.
- Sie müssen der Benutzer-ID ausreichend Speicherplatz zuordnen, damit Java ausgeführt werden kann. Sie können alle unterstützten Versionen von Java verwenden. DFHJS2LS verwendet standardmäßig die Java-Version, die im Parameter **JAVADIR** angegeben ist.

About this task

Mit dem JSON-Assistenten können Sie Sprachstrukturen aus Ihrem JSON-Schema für die Service-Provider-Anwendung erstellen.

Procedure

1. Verwenden Sie das DFHJS2LS-Stapelverarbeitungsprogramm, um eine Web-Service-Bindungsdatei und mindestens eine Sprachdatenstruktur zu generieren. DFHJS2LS enthält eine große Auswahl optionaler Parameter, die Ihnen die Flexibilität bieten, die Bindungsdatei und die Sprachstrukturen zu erstellen, die für Ihre Anwendung erforderlich sind. Ziehen Sie diese Optionen in Betracht, wenn Sie eine vorhandene Anwendung für Web-Services aktivieren:
 - Welchen Mechanismus verwendet CICS, um Daten an das Service-Provider-Anwendungsprogramm zu übergeben? Sie können Kanäle verwenden und die Daten in Containern übergeben oder Sie können einen Kommunikationsbereich verwenden. Kanäle und Container werden empfohlen. Geben Sie diese mit dem Parameter **PGMINT** an.
 - Welche Sprache möchten Sie generieren? DFHJS2LS kann COBOL-, C/C++- oder PL/I-Sprachdatenstrukturen generieren. Geben Sie die Sprache mithilfe des Parameters **LANG** an.
 - Welche Zuordnungsebene soll verwendet werden? Je höher die Zuordnungsebene, umso mehr Steuerungsmöglichkeiten und Unterstützung stehen Ihnen für die Verarbeitung von Zeichen und Binärdaten zur Laufzeit zur Verfügung. Manche optionalen Parameter sind nur für die höheren Zuordnungsebenen verfügbar. Es wird empfohlen, die höchste verfügbare Zuordnungsebene zu verwenden. Geben Sie die Zuordnungsebene mit dem Parameter **MAPPING-LEVEL** an.
 - Welcher URI soll vom Web-Service-Requester verwendet werden? Geben Sie einen relativen URI mithilfe des Parameters **URI** an, z. B. `URI=/my/test/webservice`. Der Wert wird von CICS verwendet, wenn die URIMAP-Ressource erstellt wird.
 - Unter welcher Transaktion und Benutzer-ID werden die Web-Service-Anforderung und -Antwort ausgeführt? Sie können eine Aliastransaktion verwenden, um die Anwendung auszuführen und eine Antwort an den Service-Requester zu erstellen. Die Aliastransaktion wird unter der Benutzer-ID angehängt. Geben Sie diese mit den Parametern **TRANSACTION** und **USERID** an. Diese Werte werden beim Erstellen der URIMAP-Ressource verwendet. Wenn Sie keine bestimmte Transaktion verwenden möchten, geben Sie diese Parameter nicht an.

Wenn Sie DFHJS2LS übergeben, generiert CICS die Web-Service-Bindungsdatei und platziert sie an der im Parameter **WSBIND** angegebenen Position. Die Sprachstrukturen werden in der partitionierten Datei hinzugefügt, die Sie mit dem Parameter **PDSLIB** angegeben haben.

2. Kopieren Sie die generierte Web-Service-Bindungsdatei in das Abholverzeichnis (pickup) der PIPELINE-Ressource im Providermodus, die Sie für Ihre Web-Service-Anwendung nutzen möchten. Sie müssen die Bindungsdatei im Binärmodus kopieren.
3. Schreiben Sie ein Service-Provider-Anwendungsprogramm, das die Schnittstelle zu den generierten Sprachstrukturen bildet, und implementieren Sie die erforderliche Geschäftslogik.
4. Verwenden Sie den Befehl **PIPELINE SCAN**, um die WEBSERVICE-Ressource und URIMAP-Ressource dynamisch zu erstellen.
 - Die WEBSERVICE-Ressource schließt die Web-Service-Bindungsdatei in CICS ein und wird zur Laufzeit verwendet.
 - Die URIMAP-Ressource stellt CICS die Informationen für die Zuordnung der WEBSERVICE-Ressource zu einem bestimmten URI bereit.

Alternativ können Sie die Ressourcen selbst definieren. Dies wird jedoch nicht empfohlen.

Results

Wenn es beim Übergeben von DFHJS2LS zu Fehlern kommt oder wenn die Ressourcen nicht korrekt installiert werden können, finden Sie weitere Informationen unter Troubleshooting the JSON assistant.

Service-Provider-Anwendung aus einer Datenstruktur erstellen

Mit dem CICS-Web-Service-Assistenten können Sie eine Service-Provider-Anwendung aus einer Datenstruktur einer höheren Programmiersprache erstellen.

Before you begin

Bevor Sie eine Service-Provider-Anwendung erstellen, muss Ihre Konfiguration diese Vorbedingungen erfüllen:

- Ihre Datenstrukturen einer höheren Programmiersprache müssen die folgenden Kriterien erfüllen:
 - Die Datenstrukturen müssen getrennt vom Quellenprogramm definiert werden, z. B. in einem COBOL-Copybook.
 - Wenn Ihr PL/I- oder COBOL-Anwendungsprogramm für die Eingabe und Ausgabe verschiedene Datenstrukturen verwendet, müssen diese in zwei unterschiedlichen Membern in einer partitionierten Datei definiert sein. Wenn dieselbe Struktur für Eingabe und Ausgabe verwendet wird, muss sie in einem einzigen Member definiert sein.

Für C und C++ können sich Ihre Datenstrukturen in demselben Member in einer partitionierten Datei befinden.
- Die Sprachstrukturen müssen in einer partitionierten Datei verfügbar sein und Sie müssen eine geeignete PIPELINE-Ressource in der CICS-Region erstellen.
- Sie müssen in OMVS die Benutzer-ID definieren, die von DFHLS2JS verwendet wird.
- Die Benutzer-ID muss über Leseberechtigungen für z/OS UNIX- und PDS-Bibliotheken und über Schreibberechtigungen für die Verzeichnisse verfügen, die in den Ausgabeparametern **LOGFILE** , **WSBIND** , **JSON-SCHEMA-REQUEST** und **JSON-SCHEMA-RESPONSE** angegeben sind.
- Der Benutzer-ID muss ausreichend Speicher zugeordnet sein, um Java auszuführen. Sie können alle unterstützten Versionen von Java verwenden. Standardmäßig verwendet DFHLS2JS die Java-Version, die im Parameter **JAVADIR** angegeben ist.

Procedure

Führen Sie die folgenden Schritte aus, um eine Service-Provider-Anwendung aus einer Datenstruktur einer höheren Programmiersprache zu erstellen:

1. Wenn die Schnittstelle der Service-Provider-Anwendung Kanäle und viele Container verwendet, erstellen Sie ein Kanalbeschreibungsdokument, das die Schnittstelle in JSON beschreibt. Sie müssen das Kanalbeschreibungsdokument in einem geeigneten Verzeichnis unter z/OS UNIX speichern. CICS verwendet dieses Dokument, um eine JSON-Nachricht aus den Containern in einem Kanal zu erstellen bzw. zu zerlegen. Alternativ können Sie einen (1) Container in einem Kanal verwenden und kein Kanalbeschreibungsdokument erstellen.

Weitere Informationen zum Erstellen eines Kanalbeschreibungsdokuments finden Sie unter „Kanalbeschreibungsdokument erstellen“ auf Seite 297.

2. Verwenden Sie das Stapelverarbeitungsprogramm DFHLS2JS, um eine Web-Service-Bindungsdatei und eine Web-Service-Beschreibung aus der Sprachstruktur zu generieren. Das Stapelverarbeitungsprogramm DFHLS2JS befindet sich im Verzeichnis *HLQ* .XDFHINST . Dabei ist *HLQ* die Position, an der Sie CICS installiert haben. DFHLS2JS enthält eine große Auswahl optionaler Parameter, die Ihnen die Flexibilität bieten, die Bindungsdatei und die Sprachstrukturen zu erstellen, die für Ihre Anwendung erforderlich sind. Ziehen Sie die folgenden Optionen in Betracht, wenn Sie Web-Services für eine vorhandene Anwendung aktivieren:

- Welchen Mechanismus soll CICS verwenden, um Daten an das Service-Provider-Anwendungsprogramm zu übergeben? Sie können Kanäle verwenden und die Daten in Containern übergeben oder Sie können einen Kommunikationsbereich verwenden. Geben Sie den Mechanismus mithilfe des Parameters **PGMINT** an. Wenn Ihre Anwendungsschnittstelle Kanäle und viele Container verwendet, geben Sie den Parameter **REQUEST-CHANNEL** und optional den Parameter **RESPONSE-CHANNEL** an. Sie können diese Parameter nur bei einer Zuordnungsebene 3.0 oder höher verwenden.
- Welche Zuordnungsebene soll verwendet werden? Je höher die Zuordnungsebene, umso mehr Steuerungsmöglichkeiten und Unterstützung stehen Ihnen für die Verarbeitung von Zeichen und Binärdaten zur Laufzeit zur Verfügung. Manche optionalen Parameter sind nur für die höheren Zuordnungsebenen verfügbar. Sie müssen die höchste verfügbare Zuordnungsebene im Parameter **MAPPING-LEVEL** angeben.
- Welcher URI soll vom Web-Service verwendet werden? Geben Sie einen absoluten URI mithilfe des Parameters **URI** an, z. B. **URI** = `http://www.example.org:80/my/test/webservice`. Der relative Teil dieser Adresse, `/my/test/webservice`, wird beim Erstellen der URIMAP-Ressource verwendet.

Wenn Sie DFHLS2JS übergeben, generiert CICS die Web-Service-Bindungsdatei und platziert sie an der im Parameter **WSBIND** angegebenen Position. Die generierten JSON-Schemas werden an der Position platziert, die Sie in den Parametern **JSON-SCHEMA-REQUEST** und **JSON-SCHEMA-RESPONSE** angegeben haben.

3. Überprüfen Sie das generierte JSON-Schema.

Diese Schemas werden verwendet, um die Eingabe- und Ausgabedatenformate für den JSON-Web-Service zu definieren. Der Anwendungsentwickler muss diese Schemas verwenden, wenn er eine Anwendung für die Interaktion mit dem JSON-Web-Service erstellt.

Anmerkung: Das Ändern des generierten Schemas führt dazu, dass die generierte Web-Service-Bindungsdatei, **WSBind**, ungültig wird.

Wenn Sie das Schema ändern möchten, z. B. um die Felder in dem Schema umzubenennen, müssen Sie DFHJS2LS verwenden, um eine neue Web-Service-Bindungsdatei und einen neuen Satz von Sprachstrukturen zu generieren. Das Anwendungsprogramm in CICS muss geändert werden, damit es die neue Sprachstruktur nutzt.

4. Kopieren Sie die Web-Service-Bindungsdatei in das Abholverzeichnis der Pipeline im Providermodus, die Sie für Ihre Web-Service-Anwendung nutzen möchten. Sie müssen die Web-Service-Bindungsdatei im Binärmodus kopieren.
5. Verwenden Sie den Befehl **PIPELINE SCAN**, um die WEBSERVICE-Ressource und eine URIMAP-Ressource dynamisch zu erstellen.
 - Die WEBSERVICE-Ressource enthält die Web-Service-Bindungsdatei in CICS und wird zur Laufzeit verwendet.
 - Die URIMAP-Ressource stellt CICS die Informationen für die Zuordnung der WEBSERVICE-Ressource zu einem bestimmten URI bereit.

Alternativ können Sie die Ressourcen selbst definieren.

Results

Ihre Service-Provider-Anwendung wurde erstellt.

Wenn es beim Übergeben von DFHLS2JS zu Fehlern kommt oder wenn die Ressourcen nicht korrekt installiert werden können, finden Sie weitere Informationen unter *Troubleshooting the JSON assistant*.

What to do next

Machen Sie die Web-Service-Beschreibung für jeden verfügbar, der einen Web-Service entwickelt, der auf Ihren Service zugreifen wird.

Kanalbeschreibungsdokument erstellen

Erstellen Sie ein Kanalbeschreibungsdokument, wenn Ihre Service-Provider-Anwendung eine Kanalschnittstelle mit vielen Containern verwendet.

About this task

Verwenden Sie einen XML-Editor, um das Kanalbeschreibungsdokument zu erstellen. Das Schema für die Kanalbeschreibung heißt `channel.xsd` und befindet sich im Verzeichnis `/usr/lpp/cicsts/cicsts55/schemas/channel` (wobei `/usr/lpp/cicsts/cicsts55` das Standardinstallationsverzeichnis für CICS-Dateien unter z/OS UNIX ist).

Procedure

1. Erstellen Sie ein XML-Dokument mit einem `<channel>`-Element und dem Namensbereich des CICS-Kanals.

```
<channel name="myChannel"
        xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```
2. Fügen Sie ein `<container>`-Element für jeden Container hinzu, den die Anwendungsprogrammierschnittstelle in dem Kanal verwendet. Sie müssen die einzelnen Container mithilfe von Namens-, Typ- und Verwendungsattributen beschreiben. Im folgenden Beispiel sind sechs Container mit verschiedenen Attributwerten dargestellt:

```

<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>

```

Das Strukturelement gibt an, dass der Inhalt in einer Sprachstruktur definiert ist, die sich in einem Member einer partitionierten Datei befindet.

3. Speichern Sie das XML-Dokument unter z/OS UNIX.

Kanalschema

Das Kanalbeschreibungsdokument muss dem folgenden Schema entsprechen:

```

<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. Dieses Element stellt einen CICS-Kanal dar.

2. Dieses Element stellt einen CICS-Container innerhalb des Kanals dar.
3. Eine Struktur kann nur mit Bitmodus-Containern verwendet werden. Das Attribut 'location' gibt die Position einer Datei an, die den Inhalt des Containers abbildet. Das Attribut 'structure' kann in C und C++ verwendet werden, um den Namen der Struktur anzugeben.

What to do next

Führen Sie DFHLS2JS aus, um die Zuordnungen und das JSON-Schema für die Web-Service-Provider-Anwendung zu erstellen. DFHLS2JS fügt die Zuordnungen für den Kanal in der Reihenfolge in das JSON-Schema ein, in der die Container im Kanalbeschreibungsdocument angegeben sind.

Generierte JSON-Schemas anpassen

Die JSON-Schemas, die von DFHLS2JS generiert werden, enthalten einige automatisch generierte Inhalte, die Sie möglicherweise ändern sollten, bevor Sie sie veröffentlichen. Die Anpassung von JSON-Schemas kann dazu führen, dass die Web-Service-Bindungsdatei neu generiert wird und in einigen Fällen ein Wrapperprogramm geschrieben wird.

About this task

Führen Sie diese Schritte aus, um generierte JSON-Schemas anzupassen:

Procedure

1. Um die Unterstützung für HTTPS zugänglich zu machen, verwenden Sie den Parameter **URI** in DFHLS2JS zum Festlegen eines absoluten URI.
2. Um die Netzadresse Ihres Web-Service bereitzustellen, verwenden Sie den Parameter **URI** in DFHLS2JS zum Festlegen eines absoluten URI.
3. Überlegen Sie, ob die automatisch generierten Namen in dem JSON-Schema für Ihre Zwecke geeignet sind. Sie können die Eigenschaften umbenennen.

Diese Werte bilden einen Teil der programmgesteuerten Schnittstelle, für die Sie ein Clientprogramm codieren. Wenn die generierten Namen nicht ausreichend aussagekräftig sind, kann dies die Wartung Ihres Anwendungscodes im Laufe der Zeit schwierig gestalten.

Wenn Sie einen dieser Werte ändern, müssen Sie DFHJS2LS verwenden, um die Web-Service-Bindungsdatei neu zu generieren. Die Sprachstrukturen, die erstellt werden, sind wahrscheinlich nicht kompatibel mit Ihrer vorhandenen Anwendung, deshalb können Anwendungsänderungen erforderlich sein. Überprüfen Sie die generierten Sprachstrukturen und ziehen Sie in Betracht, ein Wrapperprogramm zu schreiben, wie in Schritt 4 beschrieben.

4. Überlegen Sie, ob die Kommunikationsbereichsfelder, die in den JSON-Schemas zugänglich gemacht werden, geeignet sind. Möglicherweise erscheint es Ihnen sinnvoll, alle Felder, die für einen JSON-Cliententwickler nicht hilfreich sind, zu entfernen:
 - Felder, die nur für Ausgabewerte verwendet werden, können aus dem Schema entfernt werden, das die Eingabedatenstrukturen zuordnet.
 - Füllfelder.
 - Automatisch generierte Annotationen.

Wenn Sie eine dieser Änderungen vornehmen, müssen Sie die Web-Service-Bindungsdatei mithilfe von DFHJS2LS neu generieren. Die neuen Sprachstrukturen, die generiert werden, sind nicht mit den ursprünglichen Sprachstrukturen kompatibel, deshalb müssen Sie ein Wrapperprogramm schreiben, um Daten

aus der neuen Darstellung der alten Darstellung zuzuordnen. Dieses Wrapperprogramm muss einen Befehl **EXEC CICS LINK** an das Zielanwendungsprogramm absetzen und die zurückgegebenen Daten dann zuordnen.

Diese Anpassungen sind aufwändig, resultieren aber in den aussagekräftigsten programmgesteuerten Schnittstellen für Ihre JSON-Cliententwickler.

Results

Sie verfügen über ein angepasstes JSON-Schema, das Ihren Geschäftsanforderungen entspricht, und über ein Programm in CICS, das sie implementiert.

REST-konforme Web-Service-Provider-Anwendung erstellen

Die CICS-Implementierung von REST-konformen JSON-Web-Services entspricht der Implementierung von SOAP-Web-Services. Die meisten Konzepte und die Architektur werden gemeinsam verwendet, aber CICS setzt die Verwendung eines JSON-Schemas voraus.

About this task

Die Implementierung eines REST-konformen JSON-Web-Service umfasst die folgenden Tasks:

Procedure

1. Generieren Sie die Anwendungsschnittstelle.

Eingabe:

- Das JSON-Schema, das das Datenmodell für den REST-konformen Web-Service definiert.

Ausgabe:

- Die Sprachstrukturen (z. B. COBOL-Copybook), die das JSON-Schema zuordnen.
- Eine WSBind-Datei.

Führen Sie das Dienstprogramm DFHJS2LS aus und geben Sie die entsprechenden Eingabeparameter an. Zu diesen Parametern gehören:

- Die Position des JSON-Schemas.
- Die Liste der unterstützten Methoden (GET, PUT, POST und DELETE sind standardmäßig aktiviert).
- Der URI, unter dem der Service implementiert ist.
- Der Name des Anwendungsprogramms, das den Service implementiert.
- Alle erforderlichen Datenzuordnungsparameter.

2. Erstellen Sie eine Anwendung, die diese Schnittstelle verwendet.

Eingabe:

- Die Sprachstrukturen aus Schritt 1.
- Das Wissen, welche REST-konformen Operationen implementiert werden.

Ausgabe:

- Ein Programm, das sich für die Implementierung in CICS eignet.

Schreiben Sie ein Programm für Folgendes:

Anmerkung: Wenn Sie den Namen Ihres eigenen Containers im Parameter **CONTID** für DFHJS2LS angeben, müssen Sie statt DFHWS-DATA diesen Container verwenden, wann immer die Zeichenfolge in den folgenden Schritten erwähnt wird.

- a. Untersuchen Sie den URI, um mehr über die Identität der Ressource zu erfahren. CICS stellt verschiedene Container bereit, um Sie beim Ermitteln der interessanten Komponenten des URI zu unterstützen. Die Container sind unter Tabelle 15 beschrieben. Die Beispiele zeigen die Inhalte der einzelnen Container für den URI:

`http://www.example.org:10000/JSONServices/CustomerDetails/13388?action=query`

Wenn die passende URI-Maske (URIMAP) den folgenden Pfad hat:
`/JSONServices/CustomerDetails/*`

Tabelle 15. DFHWS-URI-Container. DFHWS-URI-Container

Container	Inhalte	Beispiel
DFHWS-URI	Vollständiger URI	<code>http://www.example.org:10000/JSONServices/CustomerDetails/13388?action=query</code>
DFHWS-URIMAPPATH	Der Teil des URI-Pfads, der mit der URI-Maske (URIMAP) übereinstimmt.	<code>/JSONServices/CustomerDetails/*</code>
DFHWS-URI-RESID	Der URI-Pfad, aus dem der Teil, der mit der URI-Maske (URIMAP) übereingestimmt hat, entfernt wurde (<i>die Ressourcen-ID</i>).	<code>13388</code>
DFHWS-URI-QUERY	Die Abfragezeichenfolge.	<code>action=query</code>

- b. Überprüfen Sie den URI. Wenn ein Problem vorliegt, melden Sie es an CICS und beenden Sie das Programm.
- c. Fragen Sie den Container DFHHTTPMETHOD ab, um zu bestimmen, welche Methode angewendet wird. Weitere Informationen finden Sie unter DFHHTTPMETHOD.
- d. Für eine POST (create)-Methode (falls erforderlich):
- Lesen Sie die Eingabedaten aus dem Container DFHWS-DATA.
 - Interpretieren Sie die Daten mithilfe der in Schritt 1 auf Seite 300 generierten Sprachstruktur(en).
 - Überprüfen Sie die Daten. Wenn ein Problem vorliegt, melden Sie es an CICS und beenden Sie das Programm.
 - Führen Sie die anwendungsspezifische Verarbeitung aus, um die *Ressource* zu erstellen.
 - Schreiben Sie optional in den Container DFHRESPONSE, um dem Client die ID der neuen Ressource mitzuteilen. Die Inhalte dieses Containers werden nicht von CICS umgesetzt, sondern direkt in der HTTP-Antwort gesendet. Der Container DFHWS-DATA wird ignoriert.
- e. Wenn eine GET (inquire)- oder HEAD (inquire)-Methode erforderlich ist, schreiben Sie die Daten, die die *Ressource* darstellen, in den Container DFHWS-DATA.
- f. Wenn eine PUT (set)-Methode erforderlich ist:
- Lesen Sie die Eingabedaten aus dem Container DFHWS-DATA.
 - Interpretieren Sie die Daten mithilfe der in Schritt 1 auf Seite 300 generierten Sprachstruktur(en).

- Überprüfen Sie die Daten. Wenn ein Problem vorliegt, melden Sie es an CICS und beenden Sie das Programm.
 - Führen Sie die anwendungsspezifische Verarbeitung aus, um die *Ressource* zu aktualisieren.
- g. Wenn eine DELETE-Methode erforderlich ist, führen Sie die anwendungsspezifische Verarbeitung aus, um die *Ressource* zu löschen.

Anmerkung: Das REST-konforme Datenmodell, das von CICS implementiert wird, sendet standardmäßig keinen Antworthauptteil für die Methoden PUT, POST oder DELETE. REST-konforme Anwendungen verwenden typischerweise den HTTP-Statuscode, um Erfolg oder Misserfolg zu melden. Wenn die Anwendung regulär beendet wird, sendet CICS die HTTP-Antwort 200 (OK). Weitere Informationen zum Senden von Fehlernachrichten finden Sie unter Application error reporting. Wenn Sie einen Antworthauptteil für eine PUT-, POST- oder DELETE-Methode senden möchten, müssen Sie in den Container DFHRESPONSE schreiben. Wenn dieser Container vorhanden ist, sendet CICS dessen Inhalte ohne weitere Verarbeitung im HTTP-Hauptteil. CICS ignoriert den Container DFHWS-DATA in der Antwortverarbeitung für diese Methoden.

3. Implementieren Sie die Artefakte.

Eingabe:

- Die WSBIND-Datei aus Schritt 1 auf Seite 300.
- Das CICS-Programm aus Schritt 2 auf Seite 300.
- Eine CICS-Pipelineressource im Providermodus, die für JSON konfiguriert ist.

Ausgabe:

- Ein implementierter REST-konformer JSON-Web-Service.

Implementieren Sie das Programm in CICS auf üblichem Wege.

Zwei Möglichkeiten:

- Implementieren Sie die WSBIND-Datei im Abholverzeichnis der Pipeline. Geben Sie dann einen Befehl **PIPELINE SCAN** aus, um die WEBSERVICE- und URIMAP-Ressourcen zu erstellen.
- Definieren und installieren Sie manuell eine WEBSERVICE-Ressource und die zugehörige URIMAP-Ressource. Die URIMAP muss den URI sowohl der PIPELINE als auch dem WEBSERVICE zuordnen.

4. Testen Sie den Service.

Eingabe:

- Das JSON-Schema.
- Der URI des REST-konformen Web-Service.
- Der implementierte Service aus Schritt 3.
- Ein JSON-Testclient Ihrer Wahl.

Ausgabe:

- Eine erfolgreich verarbeitete Anforderung.

Verwenden Sie den Testclient Ihrer Wahl, um eine JSON-Anforderung an CICS zu senden.

Wenn Sie eine unerwartete Antwort erhalten, versuchen Sie, den Fehler zu bestimmen. Weitere Informationen finden Sie unter Troubleshooting problems with JSON requests.

Designüberlegungen für REST-konforme Web-Service-Provider-Anwendungen

In diesem Thema werden einige Probleme beschrieben, die Sie bei der Planung und Gestaltung einer REST-konformen Web-Service-Provider-Anwendung für JSON beachten sollten.

Sammlungen von Ressourcen

Ein gängiges Design für REST-konforme APIs dient der Unterstützung des Abrufs von Sammlungen von Ressourcen. Beispiel: Es gibt einen Service, der wie folgt einen Satz von Objekten zurückgibt:

```
GET /Services/CustomerDetails?Surname=Cooper
```

Diese Anforderung soll Informationen zu allen CustomerDetails-Objekten zurückgeben, bei denen der Nachname "Cooper" lautet. Einzelne CustomerDetails-Objekte können mithilfe eines spezifischeren URI wie dem folgenden zurückgegeben werden:

```
GET /Services/CustomerDetails/Customer27
```

In diesem Beispiel ist Customer27 der Primärschlüssel für einen bestimmten Kunden. Die Ausgabe dieser zweiten Abfrage ist eine Instanz des CustomerDetails-Objekts. Die Ausgabe der ersten Abfrage ist weniger eindeutig: Sie könnte eine Liste von CustomerDetails-Objekten zurückgeben oder eine Liste von URIs für CustomerDetails-Objekte (die der Client einzeln abrufen kann). Beide Konventionen sind gängig.

Um eine Sammlung in CICS zu implementieren, erstellen Sie ein JSON-Schema, das entweder eine Liste von Dateninstanzen oder eine Liste von URIs beschreibt. Anschließend können Sie den Service erstellen und wie gewohnt implementieren. In diesem Beispiel wählen Sie aus, dass nur die GET-Methode implementiert werden soll. Sie können in Betracht ziehen, einen Paginierungsservice zu implementieren, damit ein Client vorwärts und rückwärts durch umfangreiche Datenmengen blättern kann:

```
GET  
/Services/CustomerDetails?startRecord=200&endRecord=225
```

Sie benötigen wahrscheinlich zwei URIMAP-Ressourcen in CICS (und zwei WEBSERVICE-Ressourcen). Eine, die die URI-Stammstruktur für die Sammlung zuordnet, und eine URIMAP mit einem Platzhalter für die Instanzen. Beispiel:

```
URIMAP1: Path=/Services/CustomerDetails  
WEBSERVICE=CollectionService  
URIMAP2: Path=/Services/CustomerDetails/* WEBSERVICE=InstanceService
```

Cacheverwaltung

Die REST-konforme Architektur fördert die Integration mit standardmäßigen HTTP-Cacheverwaltungsverfahren. So können die Ergebnisse von GET-Anforderungen im Netz zwischengespeichert werden, wodurch die Auslastung des Servers reduziert wird. Das entsprechende Verfahren beinhaltet das Festlegen eines Ablaufdatums bzw. einer Ablaufzeit für die Rückgabe von Daten für GET-Anforderungen.

Es gibt keinen allgemeinen zweckbestimmten Mechanismus zur Unterstützung eines anwendungsgesteuerten Ablaufs des Cache in CICS, aber es könnte ein Pipelinehandlerprogramm geschrieben werden, das den passenden HTTP-Header mithilfe der EXEC CICS WEB WRITE HTTPHEADER-API hinzufügt.

Anwendungsprogramme können ähnliche Tasks ausführen, aber nur, wenn sie in derselben CICS-Region gehostet sind, die die HTTP-Anforderung empfängt.

Variable Arrays von Elementen in DFHJS2LS

JSON kann Gruppen mit einer unterschiedlichen Anzahl an Elementen enthalten. Im Allgemeinen werden JSON-Schemas, die eine unterschiedliche Anzahl an Elementen enthalten, einer einzelnen Datenstruktur einer höheren Programmiersprache nicht effizient zugeordnet. CICS verarbeitet eine unterschiedliche Anzahl an Elementen in JSON-Daten mithilfe von containerbasierten Zuordnungen oder Inline-Zuordnungen.

Ein Array mit einer variierenden Anzahl von Elementen wird in dem JSON-Schema mithilfe der Schlüsselwörter `minItems` und `maxItems` in dem Schema mit einem "type"-Wert von "array" dargestellt:

- Das Schlüsselwort `minItems` gibt die Mindestanzahl von Vorkommen des Elements an. Es hat den Wert 0 oder eine beliebige positive ganze Zahl. Der Standardwert ist 0.
- Das Schlüsselwort `maxItems` gibt die Höchstanzahl von Vorkommen des Elements an. Es kann als Wert jede positive ganze Zahl größer-gleich dem Wert des Schlüsselworts `minItems` haben.
- Wenn das Schlüsselwort `maxItems` nicht angegeben ist, bedeutet das, dass das Array unbegrenzt ist.

Ein optionales Feld kann durch ein variables Array von `"maxItems":1` bezeichnet werden. Beispiel: Eine optionale 8-Byte-Zeichenfolge namens "component" :

```
"properties":{
  "component": {
    "type":"array",
    "maxItems":1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

Derselbe Effekt kann erzeugt werden, indem der Feldname nicht in den Schlüsselwortwert "required" eingeschlossen wird:

```
"properties":{
  "component": {
    "type": "string",
    "maxLength": 8
  }
}
```

Im Allgemeinen werden JSON-Schemas, die eine unterschiedliche Anzahl an Elementen enthalten, einer einzelnen Datenstruktur einer höheren Programmiersprache nicht effizient zugeordnet. Für diese Fälle verwendet CICS eine Reihe von verbundenen Datenstrukturen, die in einer Reihe von Containern an das Anwendungsprogramm übergeben werden. Diese Strukturen werden als Eingabe und Ausgabe der Anwendung verwendet:

- Wenn CICS JSON-Daten in Anwendungsdaten umsetzt, füllt es diese Strukturen mit den Anwendungsdaten, und die Anwendung liest sie.
- Wenn CICS die Anwendungsdaten in JSON-Daten umsetzt, liest es die Anwendungsdaten in den Strukturen, die von der Anwendung gefüllt wurden.

Das folgende Beispiel veranschaulicht das Format dieser Datenstrukturen. Diese Beispiele verwenden ein Array von einfachen 8-Byte-Feldern. Das Modell unterstützt jedoch Arrays komplexer Datentypen ebenso wie Arrays von Datentypen, die andere Arrays enthalten.

Beispiel 1. Feste Anzahl von Elementen

Dieses Beispiel stellt ein Element dar, das genau dreimal vorkommt.

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 3,
    "minItems": 3,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  }
},
"required": ["component"]
```

In diesem Beispiel ist die Anzahl der Vorkommen des Elements vorab bekannt, deshalb kann es als Array fester Länge in einer einfachen COBOL-Deklaration oder als das Äquivalent in anderen Sprachen dargestellt werden:

```
05 component PIC X(8) OCCURS 3 TIMES
```

Beispiel 2. Variierende Anzahl von Elementen auf Zuordnungsebene 2 und niedriger

Dieses Beispiel stellt ein obligatorisches Element dar, das ein- bis fünfmal vorkommen kann:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  }
},
"required": ["component"]
```

Die Hauptdatenstruktur enthält eine Deklaration mit zwei Feldern. Wenn CICS die JSON-Daten in binäre Daten umsetzt, enthält das erste Feld, `component-num`, die Anzahl von Vorkommen des Elements in den JSON-Daten und das zweite Feld, `component-cont`, den Namen eines Containers:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Eine zweite Datenstruktur enthält die Deklaration des Elements selbst:

```
01 DFHJS-component
02 component PIC X(8)
```

Sie müssen den Wert von `component-num` untersuchen, wobei es sich um einen Wert im Bereich zwischen 1 und 5 handelt, um herauszufinden, wie oft das Element vorkommt. Der Elementinhalt befindet sich im Container namens `component-cont`.

Der Container enthält ein Array von Elementen, wobei jedes Element von der Datenstruktur DFHJS-component zugeordnet wird.

Wenn `minItems="0"` oder nicht angegeben ist und wenn `maxItems="1"`, ist das Element optional. Um die Datenstruktur in Ihrem Anwendungsprogramm zu verarbeiten, müssen Sie den Wert von `component-num` untersuchen:

- Ist er 0, hat die Nachricht kein Komponentenelement und der Inhalt von `component-cont` ist nicht definiert.
- Ist er 1, befindet sich das Komponentenelement im Container namens `component-cont`.

Der Inhalt des Containers wird von der Datenstruktur DFHJS-component zugeordnet.

Anmerkung: Wenn die JSON-Daten aus einem einzelnen wiederkehrenden Element bestehen, generiert DFHJS2LS zwei Sprachstrukturen. Die Hauptsprachstruktur enthält die Anzahl von Elementen in dem Array und den Namen eines Containers, der das Array von Elementen enthält. Die zweite Sprachstruktur ordnet eine einzelne Instanz des wiederkehrenden Elements zu.

Beispiel 3. Variierende Anzahl von Elementen auf Zuordnungsebene 2.1 und höher

Auf Zuordnungsebene 2.1 und höher können Sie den Parameter **INLINE-MAXOCCURS-LIMIT** in den CICS-Assistenten verwenden. Der Parameter **INLINE-MAXOCCURS-LIMIT** gibt an, wie die variierende Anzahl von Elementen verarbeitet wird. Bei den Zuordnungsoptionen für eine variierende Anzahl von Elementen handelt es sich containerbasierte Zuordnung, wie unter „Beispiel 2. Variierende Anzahl von Elementen auf Zuordnungsebene 2 und niedriger“ auf Seite 305 beschrieben, oder um eine Inline-Zuordnung. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein:

- Der Standardwert von **INLINE-MAXOCCURS-LIMIT** ist 1, was sicherstellt, dass optionale Elemente inline zugeordnet werden.
- Der Wert 0 für den Parameter **INLINE-MAXOCCURS-LIMIT** verhindert eine Inline-Zuordnung.
- Wenn `maxItems` kleiner-gleich dem Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die Inline-Zuordnung verwendet.
- Wenn `maxItems` größer als der Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die containerbasierte Zuordnung verwendet.

Die Inline-Zuordnung einer variierenden Anzahl von Elementen führt dazu, dass sowohl ein Array (wie in dem obigen Beispiel für die feste Anzahl von Vorkommen) als auch ein Zähler generiert wird. Das Feld `component-num` gibt an, wie viele Instanzen des Elements vorhanden sind, und auf diese wird vom Array verwiesen. Für das Beispiel in „Beispiel 2. Variierende Anzahl von Elementen auf Zuordnungsebene 2 und niedriger“ auf Seite 305, bei dem **INLINE-MAXOCCURS-LIMIT** kleiner-gleich 5 ist, sieht die generierte Datenstruktur wie folgt aus:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

Das erste Feld, `component-num`, ist identisch mit der Ausgabe für das Beispiel für die containerbasierte Zuordnung im vorherigen Abschnitt. Das zweite Feld enthält ein Array mit der Länge 5, das groß genug ist, um die maximale Anzahl von Elementen, die generiert werden können, zu enthalten.

Die Inline-Zuordnung unterscheidet sich von der containerbasierten Zuordnung, die die Anzahl von Vorkommen des Elements und den Namen des Containers, in dem die Daten enthalten sind, speichert, insofern als dass sie alle Daten in dem aktuellen Container speichert. Durch das Speichern der Daten im aktuellen Container wird die Leistung im Allgemeinen verbessert, was die Inline-Zuordnung zur präferierten Lösung macht.

Beispiel 4. Verschachtelte variierende Arrays

Komplexe JSON-Schemas können variierende wiederkehrende Elemente enthalten, die ihrerseits wiederum variierende wiederkehrende Elemente enthalten können. In diesem Fall erstreckt sich die beschriebene Struktur über die beiden in den Beispielen beschriebenen Ebenen hinaus.

Dieses Beispiel stellt ein optionales Element namens "component2" dar, das in einem obligatorischen Element namens "component1" verschachtelt ist, wobei das obligatorische Element ein- bis fünfmal vorkommen kann:

```
"properties":{
  "component1": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "object",
      "properties":{
        "component2":{
          "type": "string",
          "maxLength": 8
        }
      },
      "required": ["component2"]
    }
  },
  "required": ["component1"]
}
```

Die übergeordnete Datenstruktur ist identisch mit früheren Beispielen:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

Die zweite Datenstruktur enthält jedoch die folgenden Elemente:

```
01 DFHJS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Eine Struktur auf dritter Ebene enthält die folgenden Elemente:

```
01 DFHJS-component2
02 component2 PIC X(8)
```

Die Anzahl von Vorkommen des äußersten Elements "component1" ist in component1-num angegeben.

Der in component1-cont benannte Container enthält ein Array mit dieser Anzahl von Instanzen der zweiten Datenstruktur DFHJS-component1.

Jede Instanz von component2-cont benennt einen anderen Container, der jeweils die Datenstruktur enthält, die von der Struktur auf dritter Ebene, DFHJS-component2, zugeordnet wird.

Um diese Struktur zu veranschaulichen, ziehen Sie das Fragment von JSON-Daten in Betracht, das dem folgenden Beispiel entspricht:

```
{ "component1":  
  [  
    {  
      "component2": "string1"  
    },  
    {  
      "component2": "string2"  
    },  
  ]  
}
```

"component1" kommt dreimal vor. Die ersten beiden Vorkommen enthalten eine Instanz von "component2" , das dritte Vorkommen nicht.

In der übergeordneten Datenstruktur enthält component-num den Wert 3. Der in component1-cont benannte Container enthält drei Instanzen von DFHJS-component1 :

1. In der ersten Instanz hat component2-num den Wert 1 und der in component2-cont benannte Container enthält *string1*.
2. In der zweiten Instanz hat component2-num den Wert 1 und der in component2-cont benannte Container enthält *string2*.
3. In der dritten Instanz hat component2-num den Wert 0 und der Inhalt von component2-cont ist nicht definiert.

In dieser Instanz wird die gesamte Datenstruktur von vier Containern dargestellt:

- Stammdatenstruktur in Container DFHJS-DATA
- In component1-cont benannter Container
- In den ersten beiden Instanzen von component2-cont benannte zwei Container

Optionale Strukturen und das Schlüsselwort **required**

Datenstrukturen werden durch das JSON-Schema "type" von "object" definiert. Die Schemas setzen Feldnamen zu einzelnen Typen in Bezug mithilfe des durch das Schlüsselwort "properties" bereitgestellten Objekts. Die Anforderung, dass diese Felder Bestandteil der JSON-Daten sein müssen, die durch das JSON-Schema beschrieben werden, wird von dem im Schlüsselwort "required" angegebenen Array gesteuert. In diesem Array werden alle Feldnamen aufgelistet, die in den JSON-Daten vorhanden sein müssen. Optionale Felder werden daher entweder durch ihr Fehlen in diesem Array dargestellt oder durch das Fehlen des Schlüsselworts "required", da das Array nicht leer sein darf. In diesem Fall sind alle Felder optional.

Optionale Felder werden als variierendes Array von null oder einem Element behandelt. Dadurch wird ein zusätzliches Feld mit dem Suffix "-num" an den Elementnamen angehängt. Wenn die Gesamtlänge mehr als 28 Zeichen beträgt, wird der Elementname abgeschnitten. Zur Laufzeit wird dies ungleich null sein, um anzugeben, dass der Wert in den JSON-Daten vorhanden war, und gleich null, wenn er nicht vorhanden war.

In diesem Beispiel werden zwei Felder dargestellt, ein obligatorisches namens "required-structure" und ein optionales namens "optional-structure" :

```
{  
  "type": "object",  
  "properties": {  
    "required-structure": {
```

```

"type":"string",
"maxLength": 8
},
"optional-structure": {
"type":"string",
"maxLength": 8
}
},
"required": [
"required-structure"
]
}

```

Die generierte COBOL-Struktur zeigt beide Felder an, aber dem zweiten wird "optional-structure-num" vorangestellt. Dies ist eine ganzzahlige Angabe der Elemente, wobei 0 angibt, dass keine Elemente vorhanden sind, und 1 angibt, dass eines vorhanden ist. Der Wert wird festgelegt, um anzugeben, ob "optional-structure" gültige Daten enthält oder nicht.

```

03 OutputData.
06 required-structure PIC X(8).
06 optional-structure-num PIC S9(9) COMP-5 SYNC.
06 optional-structure PIC X(8).

```

Berichte über Anwendungsfehler

In diesem Thema finden Sie Informationen dazu, wie Anwendungen von Providern REST-konformer JSON-Web-Services Fehler an Clients melden können.

In Top-down-Szenarios ist es wahrscheinlich, dass die Anwendung Fehlerbedingungen melden muss. Ein Fehler könnte beispielsweise lauten: „Accountnummer wurde nicht erkannt“. Diese Anforderung gilt nur für Top-down-Szenarios. Bei der Bottom-up-Entwicklung verfügt die Anwendung entweder über einen Fehlerberichtsmechanismus, der in den Datenfeldern codiert ist, oder sie wird abnormal beendet. Im Top-down-Szenario ist es unwahrscheinlich, dass das JSON-Schema ein Feld definiert, in dem Fehler gemeldet werden können. Daher ist eine Alternative erforderlich.

Für SOAP-basierte Web-Services wird dieses Problem mithilfe der API **EXEC CICS SOAPFAULT** adressiert. SOAP-Fehlernachrichten sind in JSON nicht vorhanden. Stattdessen können Sie den Container DFHHTTPSTATUS verwenden, um von Anwendungen erkannte Fehler für JSON-Anwendungen zu berichten. Weitere Informationen finden Sie unter DFHHTTPSTATUS.

Anmerkung: Anwendungen können auch den Container DFHRESPONSE und andere Steuercontainer verwenden, um ggf. eine detailliertere Fehlerantwort bereitzustellen.

Einschränkungen von JSON-Web-Services

In diesem Referenzmaterial finden Sie Informationen zu den Funktionen, die nicht von JSON-Web-Services unterstützt werden.

Die folgenden Funktionen werden nicht unterstützt:

- Pipelines im Requestermodus mit JSON werden nicht unterstützt.
- Laufzeitvalidierung von JSON-Daten gegen ein Schema wird nicht unterstützt. Der Wert des Attributs VALIDATION einer WEBSERVICE-Ressource, die mit JSON-Nutzdaten verwendet wird, wird ignoriert.
- Die Verwendung von Namensbereichen in JSON-Daten (Badgerfish- oder Mapped-Konventionen) wird nicht unterstützt.

- An CICS gesendete JSON-Nutzdaten müssen in UTF-8 codiert sein. Es wird keine andere Codierung unterstützt. Entsprechend müssen von CICS gesendete JSON-Nutzdaten immer in UTF-8 codiert sein.
- WebSphere MQ-Transporte mit JSON-Pipelines werden nicht unterstützt.
- Umsetzungsprogramme anderer Anbieter werden nicht für die Verwendung mit dem JSON-Umsetzungsprogramm unterstützt.
- Die Verwendung von WSBIND-Dateien, die für SOAP-Web-Service-Anwendungen in einer JSON-Pipeline erstellt werden, wird nicht unterstützt. WSBIND-Dateien für die Verwendung mit JSON-Service-Provider-Anwendungen müssen von dem JSON-Assistenten generiert werden.
- Wenn in JSON-Nutzdaten bei der Umsetzung durch CICS obligatorische Inhalte fehlen, werden die äquivalenten Felder in den Datenstrukturen bei der Weitergabe an das Anwendungsprogramm nicht initialisiert.
- CICS kann keine ganzzahligen Werte größer als den maximalen Wert für 'signed long' ($2^{63} - 1$) umsetzen, es sei denn, sie sind in Anführungszeichen eingeschlossen.
- Die Verwendung einfacher Datentypen wird im Stammverzeichnis eines JSON-Schemas nicht unterstützt. Das JSON-Schema beschreibt ein JSON-Objekt oder ein JSON-Array, auch wenn das JSON-Objekt wiederum einfache Datentypen, Arrays und andere Objekte enthalten kann.
- Wenn ein Array in einem JSON-Schema mit einem Wert für **maxItems** von 1 deklariert wird, serialisiert CICS das Array bei der Generierung von JSON zur Laufzeit als einfache Zeichenfolge oder ganze Zahl.

Important: Die einzigen unterstützten Zeichen für JSON-Eigenschaftsnamen sind A-Z a-z _ : für das erste Zeichen und A-Z a-z 0-9 _ : . - für alle nachfolgenden Zeichen.

Die Axis2-Web-Service-Unterstützung verfügt heute über eine Reihe von Optionen für die Entwicklung und Implementierung von Anwendungen und Anpassungen. Die folgenden Optionen werden nicht unterstützt:

- Vom Benutzer bereitgestellte Anwendungshandler - Sie müssen die von CICS bereitgestellte Anwendungshandlerklasse `com.ibm.cicsts.axis2.CICSAXIS2ApplicationHandler` verwenden.
- Vom Benutzer geschriebene Axis2-Java-Anwendungen.
- Die SOAPFAULT- und WS-Addressing-APIs können nicht mit der JSON-Pipeline verwendet werden.

Containereinschränkungen

Anmerkung: Manche Pipelinecontainer werden nicht gefüllt, wenn eine JSON-Anforderung verarbeitet wird. Weitere Informationen finden Sie unter Containers used in the pipeline.

Unterschiede bei REST-konformen Web-Services

Bei INQUIRE PIPELINE:

- SOAPLEVEL gibt NOTSOAP zurück.
- Die Attribute MTOMNOXOPST, MTOMST, SENDMTOMST, SOAPRNUM, SOAPVNUM, XOPDIRECTST und XOPSUPPORTST werden nicht verwendet.

Bei INQUIRE WEBSERVICE:

- Die Attribute ARCHIVEFILE, BINDING, VALIDATIONST, XOPDIRECTST und XOP SUPPORTST werden nicht verwendet.
- WSDLFILE gibt den Namen der JSON-Schemadatei zurück, die dem WEBSERVICE zugeordnet ist.

Bei der WEBSERVICE-Ressource:

- Die Parameter ARCHIVEFILE und VALIDATION werden nicht verwendet und ihre Werte werden ignoriert.
- WSDLFILE ist der Name der JSON-Schemadatei, die dem WEBSERVICE zugeordnet ist.

Anwendungsdaten und XML zuordnen und umsetzen

Sie können Anwendungsprogramme schreiben, um Anwendungsbinärdaten in XML umzusetzen und umgekehrt. CICS unterstützt eine Reihe höherer Programmiersprachen und stellt einen XML-Assistenten bereit, um zuzuordnen, wie die Daten während der Laufzeitverarbeitung umgesetzt werden. CICS verwendet im Rahmen der Web-Service-Unterstützung dieselbe Technologie zum Zuordnen von Anwendungsdaten zu XML in SOAP-Nachrichten.

Before you begin

Java muss installiert sein, um den XML-Assistenten und optional eine Validierung auszuführen, wenn CICS die Daten oder XML umsetzt.

About this task

Der Vorteil dieser Vorgehensweise zur Umsetzung von Anwendungsdaten in und aus XML besteht darin, dass CICS mehr kann als ein XML-Parser. CICS kann die XML interpretieren und datensatzbasierte Konvertierungen der Anwendungsdaten durchführen. Deshalb können Sie mit dieser Methode einfacher und schneller Anwendungen erstellen, die mit XML arbeiten.

Der CICS-XML-Assistent ist ein bereitgestelltes Dienstprogramm, das Sie beim Erstellen der erforderlichen Artefakte zum Umsetzen von Anwendungsbinärdaten in XML oder von XML in Anwendungsbinärdaten unterstützt. Der XML-Assistent kann die Artefakte in einem Bundleverzeichnis oder an einer anderen Position unter z/OS UNIX erstellen.

Procedure

1. Erstellen Sie die Zuordnungen mithilfe des XML-Assistenten.
2. Erstellen Sie die XMLTRANSFORM-Ressourcen in CICS, um die Zuordnungen verfügbar zu machen.
3. Erstellen oder aktualisieren Sie ein Anwendungsprogramm, um den API-Befehl **TRANSFORM** zu verwenden. Die Anwendung muss eine kanalbasierte Schnittstelle verwenden.
4. Führen Sie die Anwendung aus, um zu testen, ob die Umsetzung wie gewünscht funktioniert. Sie können die Validierung aktivieren, um zu prüfen, ob CICS die Daten ordnungsgemäß konvertiert.

What to do next

Diese Schritte werden in den folgenden Themen ausführlicher erläutert.

CICS-XML-Assistent

Der CICS-XML-Assistent besteht aus einer Gruppe von Stapeldienstprogrammen, die Sie dabei unterstützen, XML in Strukturen einer höheren Programmiersprache umzusetzen und umgekehrt. Der Assistent unterstützt die schnelle Implementierung von Anwendungen, die XML mit minimalem Programmieraufwand verarbeiten.

Das Verwenden des XML-Assistenten für CICS reduziert die Menge des Codes, den Sie schreiben müssen, um XML zu parsen oder zu erstellen; CICS setzt Daten zwischen XML-Fragmenten und der Datenstruktur eines Anwendungsprogramms um.

Der XML-Assistent kann ein XML-Schema aus einer einfachen Sprachstruktur bzw. eine Sprachstruktur aus einem vorhandenen XML-Schema erstellen und unterstützt COBOL, C/C++ und PL/I. Er generiert außerdem Metadaten, die CICS zur Laufzeit verwendet, um XML-Daten automatisch in binäre Anwendungsdaten zu konvertieren (oder umgekehrt). Die Metadaten sind in einer XML-Bindung definiert und unter z/OS UNIX gespeichert. Das Schema für die XML-Bindung befindet sich im Verzeichnis `/usr/lpp/cicsts/cicsts52/schemas/xmltransform/` unter z/OS UNIX.

Der CICS-XML-Assistent setzt sich aus zwei Dienstprogrammen zusammen:

DFHLS2SC

Dieses Dienstprogramm generiert ein XML-Schema und eine Bindung aus einer Sprachstruktur.

DFHSC2LS

Dieses Dienstprogramm generiert eine XML-Bindung und eine Sprachstruktur, die Sie in Ihren Anwendungsprogrammen verwenden können. Sie können entweder ein WSDL-Dokument oder ein XML-Schema als Eingabe verwenden.

Die JCL-Prozeduren zum Ausführen beider Programme befinden sich in der Bibliothek `hlq.XDFHINST`, wobei *hlq* das übergeordnete Qualifikationsmerkmal Ihrer CICS-Installation ist.

DFHLS2SC: Konvertierung einer höheren Programmiersprache in ein XML-Schema

Die katalogisierte Prozedur DFHLS2SC generiert aus einer Struktur einer höheren Programmiersprache ein XML-Schema und eine XML-Bindungsdatei. Verwenden Sie DFHLS2SC, wenn Sie ein CICS-Programm erstellen wollen, das XML parsen oder erstellen kann.

Jobsteueranweisungen für DFHLS2SC

JOB Startet den Job.

EXEC Gibt den Prozedurnamen an (DFHLS2SC).

INPUT.SYSUT1 DD

Gibt die Eingabe an. Die Eingabeparameter werden in der Regel im Eingabedatenstrom angegeben. Sie können jedoch auch in einer Datei oder in einem Member einer partitionierten Datei definiert werden.

Symbolische Parameter

Die folgenden symbolischen Parameter sind in DFHLS2SC definiert:

JAVADIR = *pfad*

Gibt den Namen des Java-Verzeichnisses an, das von DFHLS2SC verwendet wird. Der Wert dieses Parameters wird an /usr/lpp/ angehängt, wodurch sich der vollständige Pfadname /usr/lpp/*pfad* ergibt.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **JAVADIR** angegeben wurde.

PATHPREF = *präfix*

Gibt ein optionales Präfix an, das den z/OS UNIX-Verzeichnispfad erweitert, der in anderen Parametern verwendet wird. Der Standardwert ist eine leere Zeichenfolge.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **PATHPREF** angegeben wurde.

SERVICE = *wert*

Verwenden Sie diesen Parameter nur, wenn sie vom IBM Support dazu aufgefordert werden.

TMPDIR = *tmp-verzeichnis*

Gibt die Position eines Verzeichnisses in z/OS UNIX an, das DFHLS2SC als temporären Arbeitsbereich verwendet. Die Benutzer-ID, unter der der Job ausgeführt wird, muss über Lese- und Schreibberechtigungen für dieses Verzeichnis verfügen.

Der Standardwert ist /tmp.

TMPFILE = *tmp-präfix*

Gibt ein Präfix an, das DFHLS2SC verwendet, um die Namen der temporären Arbeitsbereichsdateien zu erstellen.

Der Standardwert ist SC2WS.

USSDIR = *pfad*

Gibt den Namen des CICS TS-Verzeichnisses im z/OS UNIX-Dateisystem an. Der Wert dieses Parameters wird an /usr/lpp/cicsts/ angehängt, wodurch sich der vollständige Pfadname /usr/lpp/cicsts/*pfad* ergibt.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **USSDIR** angegeben wurde.

Der temporäre Arbeitsbereich

DFHLS2SC erstellt die folgenden drei temporären Dateien zur Laufzeit:

```
tmp-verzeichnis/tmp-präfix.in
tmp-verzeichnis/tmp-präfix.out
tmp-verzeichnis/tmp-präfix.err
```

Dabei gilt Folgendes:

tmp-verzeichnis ist der Wert, der im Parameter **TMPDIR** angegeben ist.

tmp-präfix ist der Wert, der im Parameter **TMPFILE** angegeben ist.

Die Standardnamen für die Dateien lauten wie folgt (wenn **TMPDIR** und **TMPFILE** nicht angegeben sind):

```
/tmp/LS2SC.in
/tmp/LS2SC.out
```

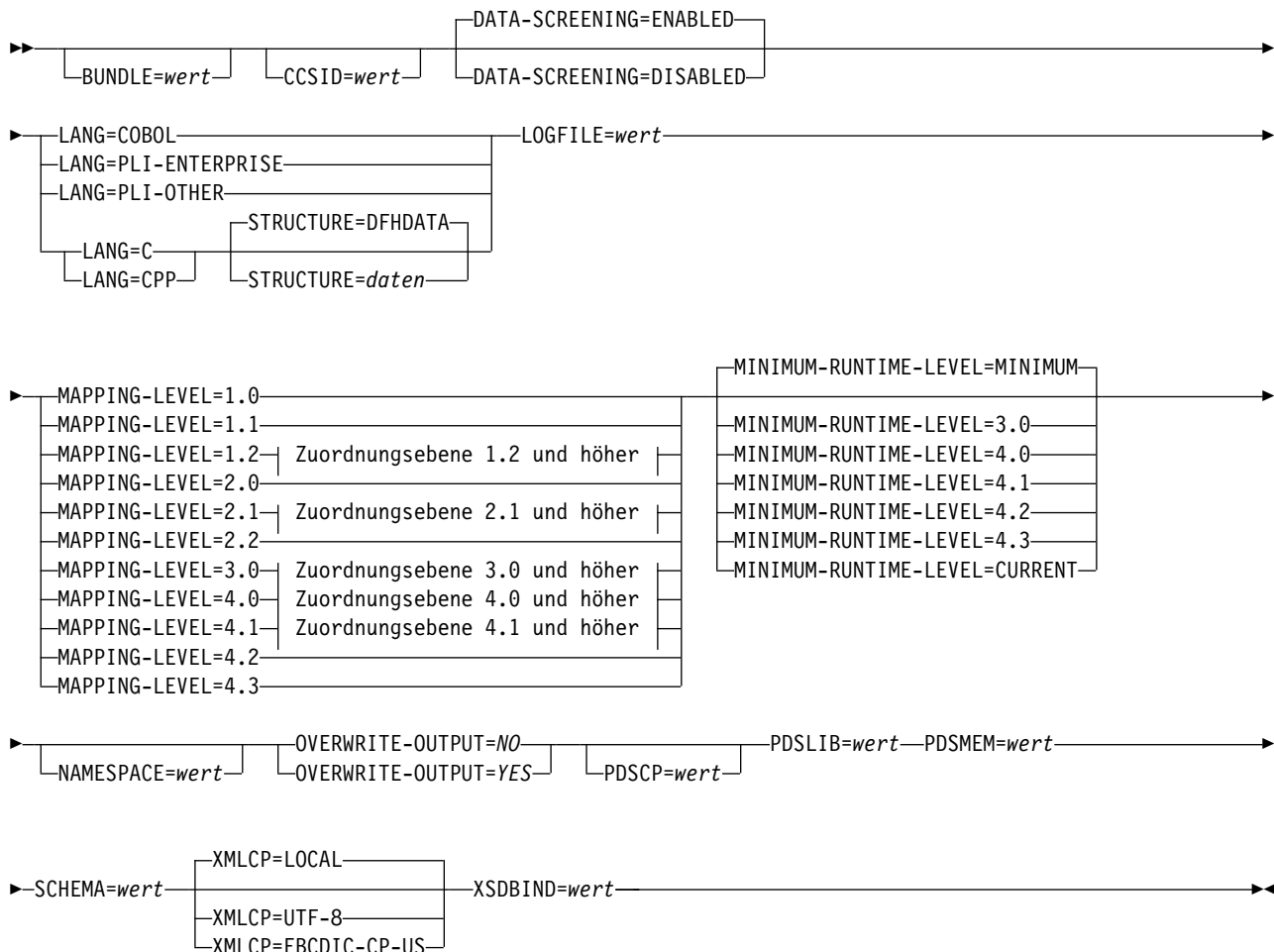
/tmp/LS2SC.err

Important: DFHLS2SC sperrt nicht den Zugriff auf die z/OS UNIX-Dateien oder -Dateimember. Wenn folglich zwei oder mehr Instanzen von DFHLS2SC gleichzeitig ausgeführt werden und dieselben temporären Arbeitsbereichsdateien verwenden, kann nicht verhindert werden, dass ein Job die Arbeitsbereichsdateien überschreibt, während ein anderer Job sie verwendet, was zu unvorhersehbaren Fehlern führt.

Deshalb wird empfohlen, eine Namenskonvention und Betriebsprozeduren auszuarbeiten, die diese Situation verhindern. Sie können beispielsweise den symbolischen Systemparameter **SYSUID** verwenden, um Arbeitsbereichsdateinamen zu generieren, die für einen einzelnen Benutzer eindeutig sind.

Diese temporären Dateien werden gelöscht, bevor der Job beendet wird.

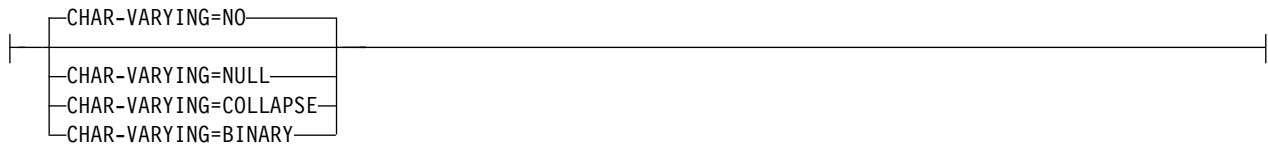
Eingabeparameter für DFHLS2SC



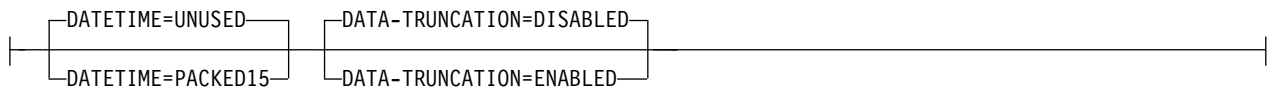
Zuordnungsebene 1.2 und höher:



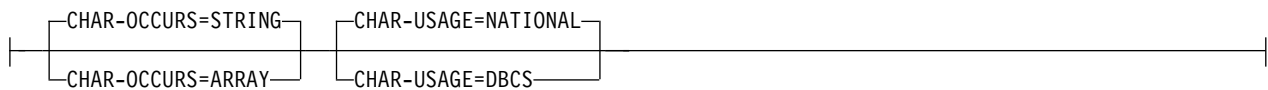
Zuordnungsebene 2.1 und höher:



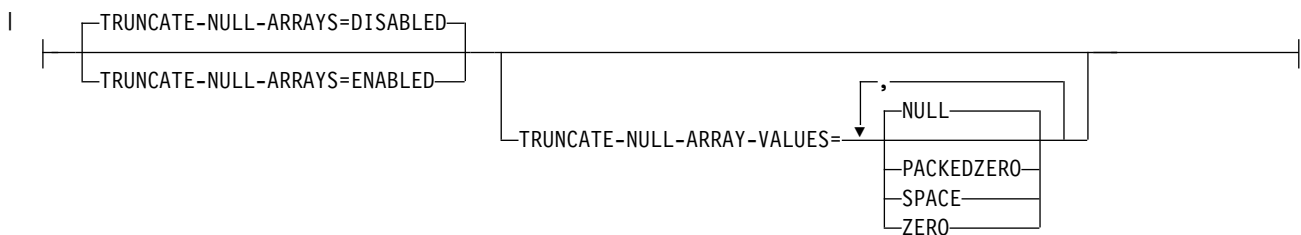
Zuordnungsebene 3.0 und höher:



Zuordnungsebene 4.0 und höher:



Zuordnungsebene 4.1 und höher:



Parameterverwendung

- Sie können die Eingabeparameter in beliebiger Reihenfolge angeben.
- Jeder Parameter muss in einer neuen Zeile eingegeben werden.
- Ein Parameter (und ggf. das Fortsetzungszeichen) darf nicht über Spalte 72 hinausgehen; die Spalten 73-80 müssen Leerzeichen enthalten.
- Wenn ein Parameter zu lang ist, um in eine einzelne Zeile zu passen, verwenden Sie am Ende der Zeile einen Stern (*), um anzugeben, dass der Parameter in der nächsten Zeile fortgesetzt wird. Alles (einschließlich der Leerzeichen) vor dem Stern wird als Teil des Parameters betrachtet. Beispiel:

```
XSDBIND=/path/xsdbinddir*
/app1
```

ist äquivalent zu

```
XSDBIND=/path/xsdbinddir/app1
```

- Ein #-Zeichen an der ersten Zeichenposition der Zeile ist ein Kommentarzeichen. Die Zeile wird ignoriert.
- Ein Komma an der letzten Zeichenposition der Zeile ist ein optionales Zeilen-trennzeichen und wird ignoriert.

Parameterbeschreibungen

BUNDLE = *wert*

Gibt den Pfad und den Namen des Bundleverzeichnisses unter z/OS UNIX an. Wenn Sie diesen Wert angeben, generiert der XML-Assistent ein Bundle, das die XSD-Bindung enthält. Die Pfadinformationen für diesen Parameter überschreiben alle Pfadinformationen für den Parameter **XSDBIND**.

Sie können optional eine Archivdatei statt eines Verzeichnisnamens angeben. Der XML-Assistent unterstützt .zip- und .jar-Archive. Sie müssen jedoch das Archiv dekomprimieren, bevor Sie versuchen, die BUNDLE-Ressource zu installieren.

Wenn Sie diesen Parameter nicht angeben, platziert CICS das XML-Schema und die Bindung an der im Parameter **XSDBIND** angegebenen Position.

CCSID = *wert*

Gibt die CCSID an, die zur Laufzeit verwendet wird, um Zeichendaten in der Anwendungsdatenstruktur zu codieren. Der Wert dieses Parameters überschreibt den Wert des Systeminitialisierungsparameters **LOCALCCSID**. *wert* muss eine EBCDIC-CCSID sein, die von Java- und z/OS-Konvertierungsservices unterstützt wird. Wenn Sie diesen Parameter nicht angeben, wird die Anwendungsdatenstruktur unter Verwendung der im Systeminitialisierungsparameter angegebenen CCSID codiert.

Sie können diesen Parameter mit jeder Zuordnungsebene verwenden.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Gibt an, wie Zeichenfelder in der Sprachstruktur auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Ein Zeichenfeld in COBOL ist eine Picture-Klausel vom Typ X, z. B. PIC(X) 10. Ein Zeichenfeld in C/C++ ist ein Zeichenarray. Dieser Parameter gilt nicht für Enterprise PL/I und andere PL/I-Sprachstrukturen. Sie haben folgende Optionen:

NO Zeichenfelder werden einem xsd:string-Element zugeordnet und als Felder mit fester Länge verarbeitet. Die maximale Länge der Daten ist gleich der Länge des Felds. NO ist der Standardwert für den Parameter **CHAR-VARYING** für COBOL und PL/I auf den Zuordnungsebenen 2.0 und niedriger.

NULL Zeichenfelder werden einem xsd:string-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet. CICS fügt bei der Umsetzung aus einem XML-Schema ein abschließendes Nullzeichen (NULL) hinzu. Die maximale Länge der Zeichenfolge wird als ein Zeichen weniger als die in der Sprachstruktur angegebene Länge berechnet. NULL ist der Standardwert für den Parameter **CHAR-VARYING** für C/C++.

COLLAPSE

Zeichenfelder werden einem xsd:string-Element zugeordnet. Abschließende und eingebettete Leerzeichen in dem Feld werden nicht in das XML-Schema eingeschlossen, z. B. wird
`<leerzeichen>AB<leerzeichen><leerzeichen><leerzeichen>C<leerzeichen>`
zu `AB<leerzeichen>C`. COLLAPSE ist der Standardwert für den Parameter **CHAR-VARYING** für COBOL und PL/I ab Zuordnungsebene 2.1.

BINARY

Zeichenfelder werden als xsd:base64binary-Datentyp zugeordnet und als Felder mit fester Länge verarbeitet. Der BINARY-Wert im Parameter **CHAR-VARYING** ist nur ab Zuordnungsebene 2.1 verfügbar.

CHAR-OCCURS = { STRING | **ARRAY** }

Gibt an, wie Zeichenarrays in der Sprachstruktur auf Zuordnungsebene 4.0 oder höher zugeordnet werden. Beispiel: PIC X OCCURS 20. Dieser Parameter kann nur in der COBOL-Sprache verwendet werden.

ARRAY

Zeichenarrays werden einem XML-Array zugeordnet. Das heißt, dass jedes Zeichen als einzelnes XML-Element zugeordnet wird. Dies ist auch das Verhalten auf Zuordnungsebene 3.0 und niedriger.

STRING

Zeichenarrays werden einer XML-Zeichenfolge zugeordnet. Das heißt, dass das gesamte COBOL-Array als einzelnes XML-Element zugeordnet wird.

CHAR-USAGE = { NATIONAL | **DBCS** }

In COBOL kann der nationale Datentyp, PIC N, für UTF-16- oder DBCS-Daten verwendet werden. Diese Einstellung wird durch die Compileroption NSYMBOL gesteuert. Sie müssen den Parameter **CHAR-USAGE** im Assistenten auf denselben Wert setzen wie die Compileroption NSYMBOL, um sicherzustellen, dass die Daten richtig verarbeitet werden. Dieser Wert wird in der Regel auf CHAR-USAGE=NATIONAL gesetzt, wenn Sie UTF-16 verwenden.

DBCS Daten aus PIC(*n*)-Feldern werden als DBCS-codierte Daten behandelt.

NATIONAL

Daten aus PIC(*n*)-Feldern werden als UTF-16-codierte Daten behandelt.

DATA-SCREENING = { ENABLED | **DISABLED** }

Gibt an, ob die von Anwendungen bereitgestellten Daten auf Fehler überprüft werden.

ENABLED

Alle von Anwendungen bereitgestellten Laufzeitdaten, die nicht mit der Sprachstruktur konsistent sind, werden als Fehler behandelt, und es wird die Nachricht DFHPI1010 ausgegeben. Eine Fehlerantwort wird an die Anwendung zurückgegeben.

DISABLED

Werte in den von Anwendungen bereitgestellten Laufzeitdaten, die nicht konsistent mit der Sprachstruktur sind, werden durch Standardwerte ersetzt. Eine Null ersetzt beispielsweise einen fehlerhaften Wert in einem numerischen Feld. Die Nachricht DFHPI1010 wird nicht ausgegeben und es wird eine normale Antwort an die Anwendung zurückgegeben. Dieses Feature kann verwendet werden, um die Fehlerantworten INVALID_PACKED_DEC und INVALID_ZONED_DEC zu vermeiden, die von nicht initialisierten Ausgabefeldern generiert werden.

DATA-TRUNCATION = { DISABLED | **ENABLED** }

Gibt an, ob Daten mit variabler Länge in einer Feldstruktur mit fester Länge toleriert werden:

DISABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, weist CICS die abgeschnittenen Daten zurück und gibt eine Fehlermeldung aus.

ENABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, toleriert CICS die abgeschnittenen Daten und verarbeitet die fehlenden Daten als Nullwerte.

DATETIME = { UNUSED | **PACKED15** }

Gibt an, ob potenzielle ABSTIME-Felder in der Struktur einer höheren Programmiersprache als Zeitmarken zugeordnet werden:

PACKED15

Gepackte Dezimalfelder mit einer Länge von 15 (8 Byte) werden als CICS ABSTIME-Felder behandelt und als Zeitmarken zugeordnet.

UNUSED

Gepackte Dezimalfelder mit einer Länge von 15 (8 Byte) werden nicht als Zeitmarken behandelt.

Sie können diesen Parameter auf der Zuordnungsebene 3.0 festlegen.

LANG = **COBOL** | **PLI-ENTERPRISE** | **PLI-OTHER** | **C** | **CPP**

Gibt die Datenstruktur der höheren Programmiersprache an:

COBOL

COBOL

PLI-ENTERPRISE

Enterprise PL/I

PLI-OTHER

Andere PL/I-Ebene als Enterprise PL/I

C

C

CPP

C++

LOGFILE = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die DFHLS2SC das Aktivitätenprotokoll und die Traceinformationen schreibt. DFHLS2SC erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur.

In der Regel verwenden Sie diese Datei nicht, aber dies kann von der IBM Serviceorganisation angefordert werden, wenn Sie Probleme mit DFHLS2SC feststellen.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 }

Gibt die Zuordnungsebene für den Assistenten an, der beim Generieren der XML-Bindung und der Sprachstrukturen verwendet werden soll. Es wird empfohlen, die aktuellste Zuordnungsebene zu verwenden, die verfügbar ist. Wenn Sie eine XML-Bindung für einen Atom-Feed erstellen, müssen Sie die Zuordnungsebene 3.0 verwenden.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 | **CURRENT** }

Gibt die älteste CICS-Laufzeitumgebung an, in der die Web-Service-Bindungsdatei implementiert werden kann. Wenn Sie eine Version auswählen, die nicht mit den anderen Parametern übereinstimmt, die Sie angegeben haben, erhalten Sie eine Fehlermeldung. Sie haben folgende Optionen:

MINIMUM

Die niedrigste mögliche Laufzeitebene von CICS wird anhand der von Ihnen ausgewählten Parameter automatisch zugewiesen.

- 3.0 Geben Sie die Laufzeitebene 3.0 oder höher an, wenn Sie den CICS-XML-Assistenten verwenden und von fortschrittlichen Datenzuordnungen profitieren möchten.
- 4.0 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.0 oder niedriger verwenden. Sie können alle optionalen Parameter auf dieser Ebene verwenden.
- 4.1 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.1 oder niedriger verwenden.
- 4.2 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS V5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.2 oder niedriger verwenden.
- 4.3 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.3 oder niedriger verwenden.

CURRENT

Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS-Region auf derselben Laufzeitebene implementiert, die Sie zum Generieren der Web-Service-Bindungsdatei verwenden.

NAMESPACE = *wert*

Gibt den Namensbereich für CICS an, der in dem generierten XML-Schema verwendet werden soll. Für Atom-Feeds stellt CICS diesen Namensbereich im Atom-Feed zusammen mit dem Atom-Namensbereich bereit.

Wenn Sie diesen Parameter nicht angeben, generiert CICS automatisch einen Namensbereich.

OVERWRITE-OUTPUT = NO | YES

Steuert, ob vorhandene CICS-BUNDLES im Dateisystem überschrieben werden können.

NO Vorhandene BUNDLES werden nicht ersetzt. Wenn ein vorhandenes BUNDLE gefunden wird, gibt DFHLS2SC die Fehlermeldung DFHPI9689E aus und wird beendet.

YES Vorhandene BUNDLES werden ersetzt. Wenn ein vorhandenes BUNDLE gefunden wird, wird die Nachricht DFHPI9683W ausgegeben, um Sie darüber zu informieren, dass die Datei ersetzt wurde.

PDSCP = *wert*

Gibt die Codepage an, die in den Mitgliedern der partitionierten Datei verwendet wird, wobei *wert* eine CCSID-Nummer oder eine Java-Codepagenummer ist. Wenn dieser Parameter nicht angegeben wird, wird die z/OS UNIX System Services-Codepage verwendet. Beispielsweise können Sie PDSCP=037 angeben.

PDSLIB = *wert*

Gibt den Namen der partitionierten Datei an, die die zu verarbeitenden Datenstrukturen der höheren Programmiersprachen enthält.

Restriction: Die Datensätze in der partitionierten Datei müssen eine feste Länge von 80 Bytes haben.

PDSMEM = *wert*

Gibt den Namen des Members der partitionierten Datei an, das die zu verarbeitenden Strukturen der höheren Programmiersprachen enthält.

SCHEMA = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die das XML-Schema geschrieben wird. Das XML-Schema entspricht der WSDL 2.0-Spezifikation. DFHLS2SC erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur.

STRUCTURE = { **DFHDATA** | *daten* }

Der Name der übergeordneten Datenstruktur in C und C++. Der Standardwert ist DFHDATA.

TRUNCATE-NULL-ARRAYS = { **DISABLED** | **ENABLED** }

Gibt an, wie strukturierte Arrays auf Zuordnungsebene 4.1 oder höher verarbeitet werden. Wenn diese Option aktiviert ist, versucht CICS, leere Datensätze innerhalb eines Arrays zu erkennen (weitere Informationen zur Identifizierung leerer Datensätze finden Sie unter TRUNCATE-NULL-ARRAY-VALUES). Wenn fünf aufeinanderfolgende leere Arraydatensätze erkannt werden, wird das Array bei der Generierung von XML/JSON beim ersten solchen Datensatz abgeschnitten. Diese Abschneidefunktion ist nur für Arrays mit strukturiertem Inhalt aktiviert, Arrays aus einfachen primitiven Feldern werden nicht abgeschnitten. Das Abschneiden von Arrays kann zu einer kürzeren Darstellung der Daten in JSON/XML führen, dies ist jedoch nicht ohne Risiko. Wenn fünf aufeinanderfolgende Datensätze fehlerhaft als nicht initialisierter Speicher identifiziert werden (beispielsweise weil sie legitim niedrige Werte enthalten), kann es zu Datenverlusten kommen. Wenn TRUNCATE-NULL-ARRAYS aktiviert und TRUNCATE-NULL-ARRAY-VALUES nicht festgelegt ist, wird der Standardwert für TRUNCATE-NULL-ARRAY-VALUES verwendet.

TRUNCATE-NULL-ARRAY-VALUES = { **NULL** | **PACKEDZERO** | **SPACE** | **ZERO** }

Gibt an, welche Werte für die Verarbeitung von TRUNCATE-NULL-ARRAYS auf Zuordnungsebene 4.1 oder höher als leer behandelt werden. Standardmäßig wird der Nullwert ('0x00' oder 'low-values') als leer behandelt. Wenn alle Bytes des Speichers in einem Datensatz eines strukturierten Arrays Nullen enthalten, wird der gesamte Datensatz als leer betrachtet. Ein Wert oder mehrere Werte vom Typ NULL, PACKEDZERO, SPACE und ZERO können in einer durch Kommas getrennten Liste angegeben werden.

NULL Impliziert ein Nullzeichen (0x00).

PACKEDZERO

Impliziert eine gepackte Dezimalnull mit positivem Vorzeichen (0x0C), eine gepackte Dezimalnull mit negativem Vorzeichen (0x0D) oder eine gepackte Dezimalnull ohne Vorzeichen (0x0F).

SPACE

Impliziert einen SBCS EBCDIC-Bereich (0x40).

ZERO Impliziert eine gezonte Dezimalnull ohne Vorzeichen (0xF0).

Jede passende Kombination der ausgewählten Bytes innerhalb eines strukturierten Arraydatensatzes führt dazu, dass der gesamte Datensatz als leer erkannt wird.

Wenn für TRUNCATE-NULL-ARRAY-VALUES ein Wert definiert ist, muss TRUNCATE-NULL-ARRAYS aktiviert sein.

XSDBIND = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der XSD-Bindung. DFHLS2SC

erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur. Ist der Parameter BUNDLE angegeben, schließen Sie den Pfad aus. Die Dateierweiterung ist .xsdbind.

DFHSC2LS: Konvertierung eines XML-Schemas in eine höhere Programmiersprache

Die katalogisierte Prozedur DFHSC2LS generiert eine Datenstruktur für eine höhere Programmiersprache sowie eine XML-Bindung aus einem XML-Schema oder WSDL-Dokument. Verwenden Sie DFHSC2LS, wenn Sie ein CICS-Programm erstellen wollen, das XML parsen oder erstellen kann. In diesem Thema werden die Jobsteueranweisungen, symbolischen Parameter, Eingabeparameter und ihre Beschreibungen für DFHSC2LS aufgelistet.

Jobsteueranweisungen für DFHSC2LSJ

JOB Startet den Job.

EXEC Gibt den Prozedurnamen an (DFHSC2LS).

INPUT.SYSUT1 DD

Gibt die Eingabe an. Die Eingabeparameter werden in der Regel im Eingabedatenstrom angegeben. Sie können jedoch auch in einer Datei oder in einem Member einer partitionierten Datei definiert werden.

Symbolische Parameter

Die folgenden symbolischen Parameter sind in DFHSC2LS definiert:

JAVADIR = *pfad*

Gibt den Namen des Java-Verzeichnisses an, das von DFHSC2LS verwendet wird. Der Wert dieses Parameters wird an /usr/lpp/ angehängt, wodurch sich der vollständige Pfadname /usr/lpp/*pfad* ergibt.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **JAVADIR** angegeben wurde.

PATHPREF = *präfix*

Gibt ein optionales Präfix an, das den z/OS UNIX-Verzeichnispfad erweitert, der in anderen Parametern verwendet wird. Der Standardwert ist eine leere Zeichenfolge.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **PATHPREF** angegeben wurde.

TMPDIR = *tmp-verzeichnis*

Gibt die Position eines Verzeichnisses in z/OS UNIX an, das DFHSC2LS als temporären Arbeitsbereich verwendet. Die Benutzer-ID, unter der der Job ausgeführt wird, muss über Lese- und Schreibberechtigungen für dieses Verzeichnis verfügen.

Der Standardwert ist /tmp.

TMPFILE = *tmp-präfix*

Gibt ein Präfix an, das DFHSC2LS verwendet, um die Namen der temporären Arbeitsbereichsdateien zu erstellen.

Der Standardwert ist SC2LS.

USSDIR = *pfad*

Gibt den Namen des CICS TS-Verzeichnisses im z/OS UNIX-Dateisystem an.

Der Wert dieses Parameters wird an `/usr/lpp/cicsts/` angehängt, wodurch sich der vollständige Pfadname `/usr/lpp/cicsts/pfad` ergibt.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **USSDIR** angegeben wurde.

SERVICE = *wert*

Verwenden Sie diesen Parameter nur, wenn sie vom IBM Support dazu aufgefordert werden.

Der temporäre Arbeitsbereich

DFHSC2LS erstellt die folgenden drei temporären Dateien zur Laufzeit:

tmp-verzeichnis/tmp-präfix.in
tmp-verzeichnis/tmp-präfix.out
tmp-verzeichnis/tmp-präfix.err

Dabei gilt Folgendes:

tmp-verzeichnis ist der Wert, der im Parameter **TMPDIR** angegeben ist.
tmp-präfix ist der Wert, der im Parameter **TMPFILE** angegeben ist.

Die Standardnamen für die Dateien lauten wie folgt (wenn **TMPDIR** und **TMPFILE** nicht angegeben sind):

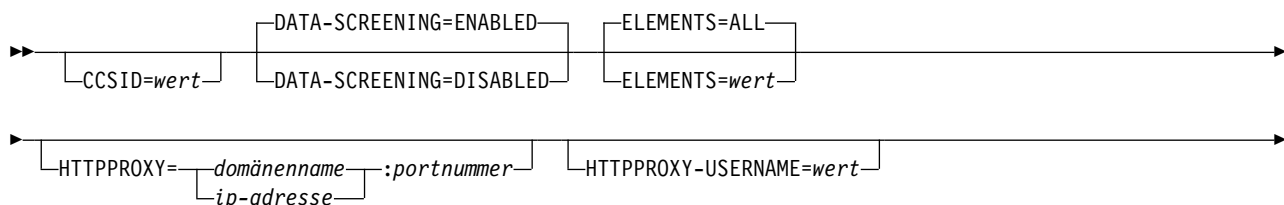
`/tmp/SC2LS.in`
`/tmp/SC2LS.out`
`/tmp/SC2LS.err`

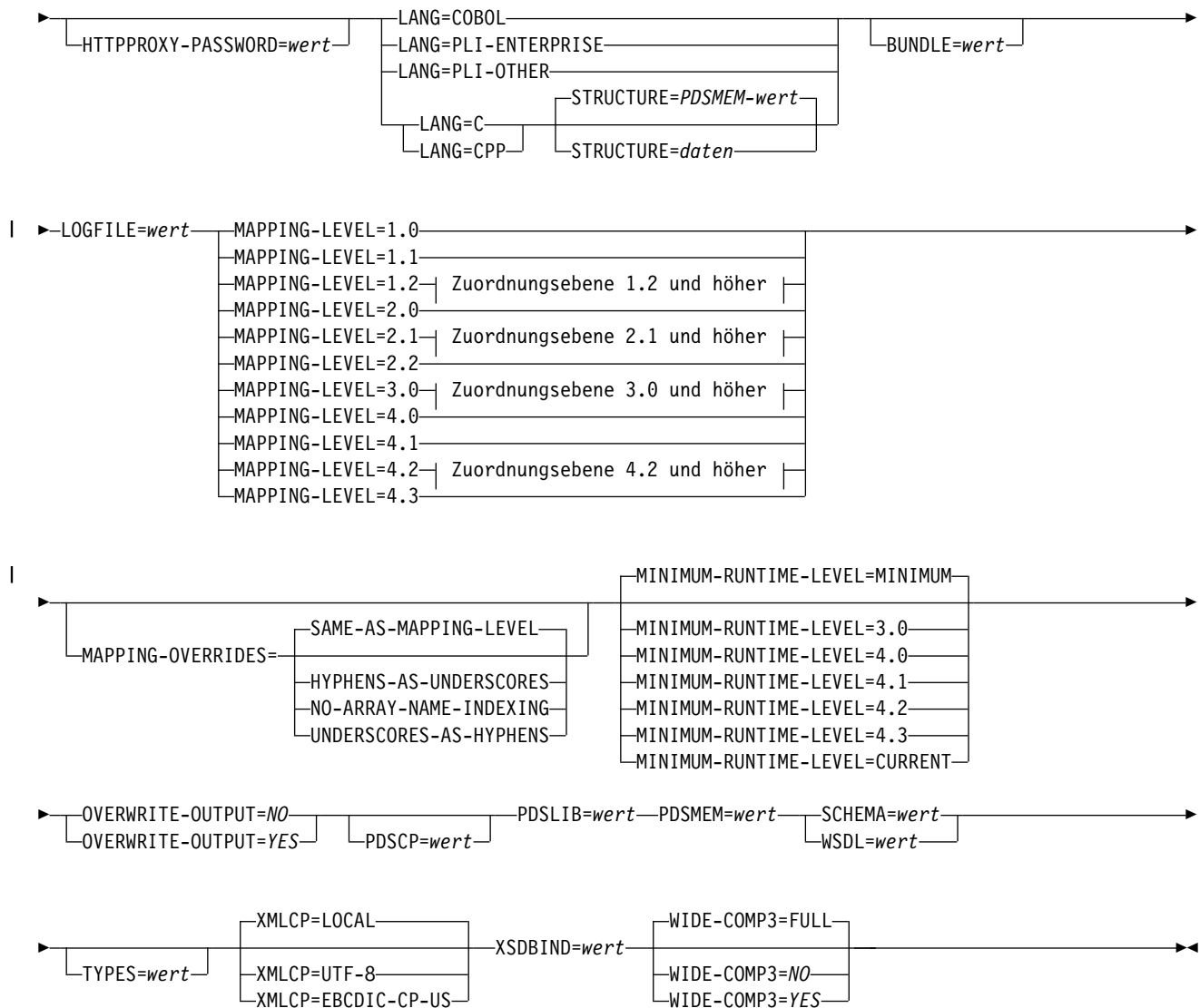
Important: DFHSC2LS sperrt nicht den Zugriff auf die z/OS UNIX-Dateien oder -Dateimember. Wenn folglich zwei oder mehr Instanzen von DFHSC2LS gleichzeitig ausgeführt werden und dieselben temporären Arbeitsbereichsdateien verwenden, kann nicht verhindert werden, dass ein Job die Arbeitsbereichsdateien überschreibt, während ein anderer Job sie verwendet, was zu unvorhersehbaren Fehlern führt.

Deshalb wird empfohlen, eine Namenskonvention und Betriebsprozeduren auszuarbeiten, die diese Situation verhindern. Sie können beispielsweise den symbolischen Systemparameter **SYSUID** verwenden, um Arbeitsbereichsdateinamen zu generieren, die für einen einzelnen Benutzer eindeutig sind.

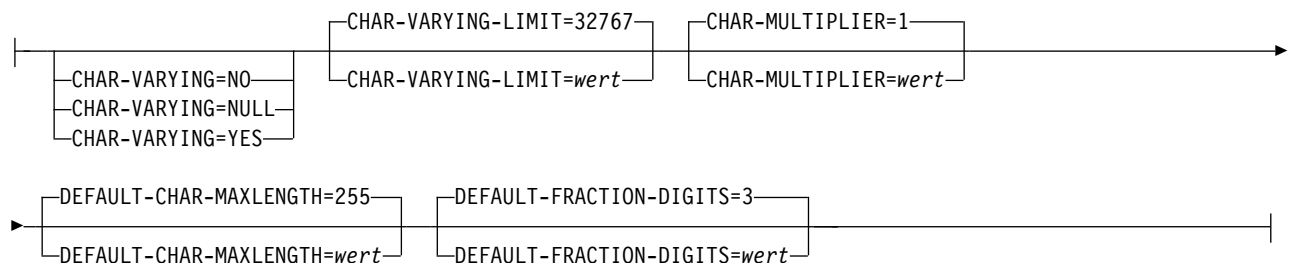
Diese temporären Dateien werden gelöscht, bevor der Job beendet wird.

Eingabeparameter für DFHSC2LS





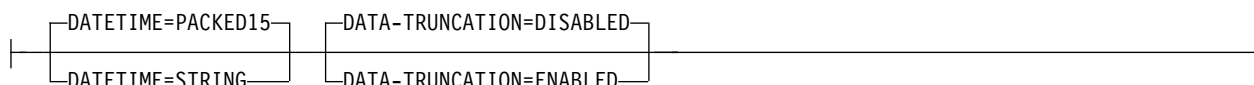
Zuordnungsebene 1.2 und höher:



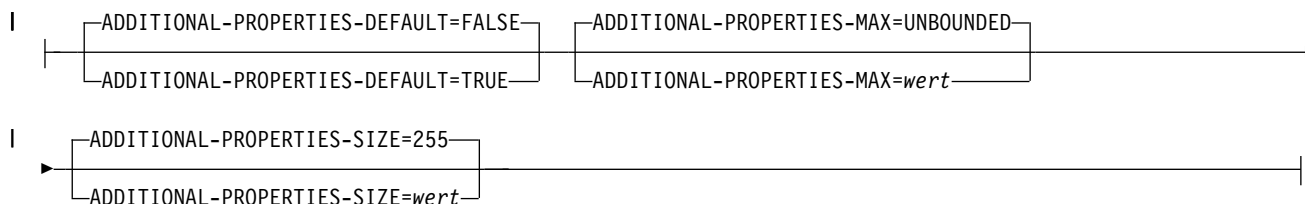
Zuordnungsebene 2.1 und höher:



Zuordnungsebene 3.0 und höher:



Zuordnungsebene 4.2 und höher:



Parameterverwendung

- Sie können die Eingabeparameter in beliebiger Reihenfolge angeben.
- Jeder Parameter muss in einer neuen Zeile eingegeben werden.
- Ein Parameter (und ggf. das Fortsetzungszeichen) darf nicht über Spalte 72 hinausgehen; die Spalten 73-80 müssen Leerzeichen enthalten.
- Wenn ein Parameter zu lang ist, um in eine einzelne Zeile zu passen, verwenden Sie am Ende der Zeile einen Stern (*), um anzugeben, dass der Parameter in der nächsten Zeile fortgesetzt wird. Alles (einschließlich der Leerzeichen) vor dem Stern wird als Teil des Parameters betrachtet. Beispiel:

```
XSDBIND=xsdbinddir*  
      /app1
```

ist äquivalent zu

```
XSDBIND=xsdbinddir/app1
```

- Ein #-Zeichen an der ersten Zeichenposition der Zeile ist ein Kommentarzeichen. Die Zeile wird ignoriert.

Parameterbeschreibungen

ADDITIONAL-PROPERTIES-DEFAULT = { **true** | **false** }

Gibt an, ob JSON-Schemaobjekte, die zusätzliche Eigenschaften nicht explizit unterstützen, als unterstützende Elemente interpretiert werden oder nicht. Zusätzliche JSON-Eigenschaften sind alle Eigenschaften in einem JSON-Objekt, die im JSON-Schema nicht vordefiniert sind. Diese Eigenschaften werden in der Regel vom Datenumsetzungsmechanismus als unerwartete Zusatzdaten zurückgewiesen. Wenn **ADDITIONAL-PROPERTIES-DEFAULT** auf TRUE gesetzt ist, oder wenn das JSON-Schema explizit `additionalProperties:true` für ein Objekt festlegt, wird ein Bereich für solche Werte in den generierten Copybooks reserviert. Anwendungen können mit diesen Werten interagieren, indem sie die zugeordneten Felder in den Copybooks verwenden.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

ADDITIONAL-PROPERTIES-MAX = { **0-20** | **UNBOUNDED** }

Gibt an, wie viele zusätzliche Eigenschaften für ein entsprechendes JSON-Objekt unterstützt werden. Siehe **ADDITIONAL-PROPERTIES-DEFAULT**. Die generierten Copybooks enthalten Strukturen, die sich für die Adressierung aller zusätzlichen Eigenschaften eignen. Standardmäßig gibt es keine maximale Einschränkung für die Anzahl der unterstützten Eigenschaften. Die Copybooks werden

in ähnlicher Weise wie Arrays ohne Einschränkungen generiert und verwenden Container. Mit diesem Parameter kann eine maximale Einschränkung angewendet werden, die in Kombination mit dem Parameter **INLINE-MAXOCCURS-LIMIT** dafür sorgt, dass ein Array mit fester Länge für die maximale Anzahl von Eigenschaften zugeordnet werden kann, wodurch keine Container erforderlich sind.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

ADDITIONAL-PROPERTIES-SIZE = { **16-32767** | **255** }

Gibt die maximale Größe für jede der zusätzlichen JSON-Eigenschaften an. Wenn ein JSON-Objekt zusätzliche Eigenschaften unterstützt, wie in **ADDITIONAL-PROPERTIES-DEFAULT** definiert, dann haben die generierten Copybooks Bindungen, um Eigenschaften bis zu der von **ADDITIONAL-PROPERTIES-MAX** angegebenen Anzahl zu unterstützen. Standardmäßig beträgt der für jede zusätzliche Eigenschaft unterstützte Maximalwert 255 Zeichen. Ein Feld dieser Größe wird in den erstellten Copybooks generiert. Diese Größe kann durch Festlegen des Parameters **ADDITIONAL-PROPERTIES-SIZE** angepasst werden. So wird beispielsweise ein JSON-Objekt verarbeitet, das folgende Eigenschaft enthält:

```
"example": { "notes": "this extra property was not defined in the JSON Schema"
}
```

Wenn die Copybooks generiert wurden, um zusätzliche Eigenschaften zu unterstützen, wird der gesamte Wert zur Verarbeitung an die Anwendung übergeben. Der Wert beginnt mit dem führenden Anführungszeichen vor dem Eigenschaftsschlüssel und endet mit der abschließenden rechten geschweiften Klammer im Eigenschaftswert. In diesem Beispiel sind es etwa 100 Zeichen. Der Wert für **ADDITIONAL-PROPERTIES-SIZE** muss groß genug sein, um den größtmöglichen Wert enthalten zu können. Wenn der zugeordnete Puffer für den verarbeiteten Wert zu klein ist, wird eine Fehlerantwort generiert.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

CCSID = *wert*

Gibt die CCSID an, die zur Laufzeit verwendet wird, um Zeichendaten in der Anwendungsdatenstruktur zu codieren. Der Wert dieses Parameters überschreibt den Wert des Systeminitialisierungsparameters **LOCALCCSID**. *wert* muss eine EBCDIC-CCSID sein, die von Java- und z/OS-Konvertierungsservices unterstützt wird (siehe z/OS Unicode Services User's Guide and Reference). Wenn Sie diesen Parameter nicht angeben, wird die Anwendungsdatenstruktur unter Verwendung der im Systeminitialisierungsparameter angegebenen CCSID codiert.

Sie können diesen Parameter mit jeder Zuordnungsebene verwenden.

CHAR-MULTIPLIER = { **1** | *wert* }

Gibt die Anzahl von Bytes an, die für die einzelnen Zeichen auf Zuordnungsebene 1.2 oder höher zulässig sein sollen. Der Wert (*wert*) dieses Parameters kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein. Alle nicht numerischen, zeichenbasierten Zuordnungen unterliegen diesem Multiplikator. Binäre, numerische, in Zonen eingeteilte und gepackte Dezimalfelder unterliegen diesem Multiplikator nicht.

Dieser Parameter kann nützlich sein, wenn Sie beispielsweise planen, DBCS-Zeichen zu verwenden, bei denen Sie sich für einen Multiplikator von 3 entscheiden können, um zur Laufzeit Platz zu lassen für mögliche SO- und SI-Zeichen (shift-out = Umschalttaste nicht gedrückt, shift-in = Umschalttaste gedrückt) um jedes Doppelbytezeichen herum.

Wenn Sie **CCSID=1200** (d. h. UTF-16) angeben, sind 2 oder 4 die einzig gültigen Werte für **CHAR-MULTIPLIER**. Wenn Sie UTF-16 verwenden, ist der Standardwert 2. Verwenden Sie **CHAR-MULTIPLIER=2**, wenn Sie erwarten, dass Anwendungsdaten Zeichen enthalten, die eine UTF-16-Codierungseinheit erfordern. Verwenden Sie **CHAR-MULTIPLIER=4**, wenn Sie erwarten, dass Anwendungsdaten Zeichen enthalten, die zwei UTF-16-Codierungseinheiten erfordern.

Anmerkung: Wenn Sie **CHAR-MULTIPLIER** auf 1 festlegen, schließt dies nicht aus, dass DBCS-Zeichen verwendet werden können. Und wenn Sie den Parameter auf 2 festlegen, schließt dies nicht aus, dass UTF-16-Ersatzzeichenpaare verwendet werden können. Wenn jedoch regelmäßig Breitzichen verwendet werden, werden einige gültige Werte nicht in das zugeordnete Feld passen. Wenn ein größerer Wert für **CHAR-MULTIPLIER** verwendet wird, ist es möglich, mehr Zeichen in dem zugeordneten Feld zu speichern als in der XML gültig sind. Achten Sie darauf, die passenden Bereichseinschränkungen einzuhalten.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Gibt an, wie Zeichendaten mit variabler Länge auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Binärdatentypen mit variabler Länge werden immer entweder einem Container oder einer variierenden Struktur zugeordnet. Wenn Sie diesen Parameter nicht angeben, hängt die Standardzuordnung von der angegebenen Sprache ab. Sie haben folgende Optionen:

NO Zeichendaten mit variabler Länge werden als Zeichenfolgen mit fester Länge zugeordnet.

NULL Zeichendaten mit variabler Länge werden auf null endenden Zeichenfolgen zugeordnet.

YES Zeichendaten mit variabler Länge werden in PL/I einem CHAR VARYING-Datentyp zugeordnet. In den COBOL-, C- und C++-Sprachen werden Zeichendaten mit variabler Länge einer äquivalenten Darstellung zugeordnet, die aus zwei verwandten Elementen besteht: der Datenlänge und den Daten.

CHAR-VARYING-LIMIT = { **32767** | *wert* }

Gibt die maximale Größe von Binärdaten und Zeichendaten mit variabler Länge an, die der Sprachstruktur auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Wenn die Zeichen oder Binärdaten größer sind als der in diesem Parameter angegebene Wert, werden sie einem Container zugeordnet, und der Containername wird in der generierten Sprachstruktur verwendet. Der Wert (*wert*) kann zwischen 0 und dem Standardwert 32.767 Bytes liegen.

DATA-SCREENING = { **ENABLED** | **DISABLED** }

Gibt an, ob die von Anwendungen bereitgestellten Daten auf Fehler überprüft werden.

ENABLED

Alle von Anwendungen bereitgestellten Laufzeitdaten, die nicht mit der Sprachstruktur konsistent sind, werden als Fehler behandelt, und es wird die Nachricht DFHPI1010 ausgegeben. Eine Fehlerantwort wird an die Anwendung zurückgegeben.

DISABLED

Werte in den von Anwendungen bereitgestellten Laufzeitdaten, die nicht konsistent mit der Sprachstruktur sind, werden durch Standardwerte ersetzt. Eine Null ersetzt beispielsweise einen fehlerhaften Wert in einem numerischen Feld. Die Nachricht DFHPI1010 wird nicht ausgegeben und es wird eine normale Antwort an die Anwendung zurückgegeben. Dieses Feature kann verwendet werden, um die Feh-

lerantworten INVALID_PACKED_DEC und INVALID_ZONED_DEC zu vermeiden, die von nicht initialisierten Ausgabefeldern generiert werden.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Gibt an, ob Daten mit variabler Länge in einer Feldstruktur mit fester Länge toleriert werden:

DISABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, weist CICS die abgeschnittenen Daten zurück und gibt eine Fehlermeldung aus.

ENABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, toleriert CICS die abgeschnittenen Daten und verarbeitet die fehlenden Daten als Nullwerte.

DATETIME = { **PACKED15** | **STRING** }

Gibt an, dass "xsd:dateTime"-Felder dem Datenformat CICS ABSTIME oder Text zugeordnet werden:

PACKED15

"xsd:dateTime"-Felder werden dem Format CICS ABSTIME zugeordnet.

STRING

"xsd:dateTime"-Felder werden Text zugeordnet. Die Zuordnung ist identisch mit allen vorherigen Zuordnungsebenen.

Sie können diesen Parameter auf der Zuordnungsebene 3.0 verwenden.

DEFAULT-CHAR-MAXLENGTH = { **255** | *wert* }

Gibt die Standardarraylänge von Zeichendaten in Zeichen für Zuordnungen an, in deren XML-Schemadokument oder WSDL-Dokument keine Länge eingeschlossen ist, wenn die Zuordnungsebene 1.2 oder höher ist. Der Wert (*wert*) dieses Parameters kann eine positive ganze Zahl im Bereich von 1 bis 2.147.483.647 sein.

DEFAULT-FRACTION-DIGITS = **3** | *wert*

Gibt die Standardanzahl an Bruchteilstellen für einen XML-Dezimalschematyp an. Der Standardwert ist 3. Für COBOL liegt der gültige Bereich zwischen 0 und 17, wenn der Parameter **WIDE-COMP3** verwendet wird. Für C oder PL/I liegt der gültige Bereich zwischen 0 und 30.

ELEMENTS = { **ALL** | *wert* }

Definiert eine Liste lokaler Namen von globalen Elementen, die aktiviert werden sollen. Der Standardwert ALL gibt an, dass alle globalen Elemente aktiviert werden.

HTTPPROXY = { *domänenname* | *ip-adresse* } : *portnummer*

Wenn Ihr XML-Schema oder WSDL-Dokument Referenzen auf andere XML-Schema- oder WSDL-Dateien enthält, die sich im Internet befinden, und das System, auf dem Sie DFHSC2LS ausführen, einen Proxy-Server für den Zugriff auf das Internet verwendet, geben Sie entweder den Domännennamen oder die IP-Adresse und die Portnummer des Proxy-Servers an. Beispiel:

HTTPPROXY=proxy.example.com:8080

In anderen Fällen ist dieser Parameter nicht erforderlich.

HTTPPROXY-USERNAME = *wert*

Gibt den HTTP-Proxy-Benutzernamen an, der mit **HTTPPROXY-PASSWORD** verwen-

det werden muss, wenn das System, auf dem DFHSC2LS ausgeführt wird, einen HTTP-Proxy-Server für den Zugriff auf das Internet verwendet und wenn der HTTP-Proxy-Server Basisauthentifizierung verwendet. Sie können diesen Parameter nur verwenden, wenn Sie auch **HTTPPROXY** angeben.

HTTPPROXY-PASSWORD = *wert*

Gibt das HTTP-Proxy-Kennwort an, das mit **HTTPPROXY-USERNAME** verwendet werden muss, wenn das System, auf dem DFHSC2LS ausgeführt wird, einen HTTP-Proxy-Server für den Zugriff auf das Internet verwendet und wenn der HTTP-Proxy-Server Basisauthentifizierung verwendet. Sie können diesen Parameter nur verwenden, wenn Sie auch **HTTPPROXY** angeben.

INLINE-MAXOCCURS-LIMIT = { 1 | *wert* }

Gibt auf Basis des Attributs `maxOccurs` des XML-Attributs an, ob variabel wiederkehrende Inline-Inhalte verwendet werden.

Der Parameter **INLINE-MAXOCCURS-LIMIT** ist nur ab Zuordnungsebene 2.1 verfügbar. Der Wert (*wert*) von **INLINE-MAXOCCURS-LIMIT** kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein. Der Wert 0 gibt an, dass die Inline-Zuordnung nicht verwendet wird. Der Wert 1 gibt an, dass optionale Elemente inline zugeordnet werden. Wenn die Variable *wert* des Attributs `maxOccurs` größer als die Variable *wert* von **INLINE-MAXOCCURS-LIMIT** ist, wird eine containerbasierte Zuordnung verwendet. Andernfalls wird eine Inline-Zuordnung verwendet.

Wenn Sie entscheiden, ob variabel wiederkehrende Listen inline zugeordnet werden sollen, ziehen Sie die Länge der einzelnen Elemente von wiederkehrenden Daten in Betracht. Wenn es wenige lange Instanzen gibt, ist die containerbasierte Zuordnung die bevorzugte Lösung. Wenn es viele kurze Instanzen gibt, ist die Inline-Zuordnung geeigneter.

LANG = **COBOL** | **PLI-ENTERPRISE** | **PLI-OTHER** | **C** | **CPP**

Gibt die Datenstruktur der höheren Programmiersprache an:

COBOL

COBOL

PLI-ENTERPRISE

Enterprise PL/I

PLI-OTHER

Andere PL/I-Ebene als Enterprise PL/I

C **C**

CPP **C++**

LOGFILE = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die DFHSC2LS das Aktivitätenprotokoll und die Traceinformationen schreibt. DFHSC2LS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur.

In der Regel verwenden Sie diese Datei nicht, aber dies kann von der IBM Serviceorganisation angefordert werden, wenn Sie Probleme mit DFHSC2LS feststellen.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 }

Gibt die Zuordnungsebene für den Assistenten an, der beim Generieren der XML-Bindung und der Sprachstrukturen verwendet werden soll. Sie sollten die aktuellste Zuordnungsebene verwenden, die verfügbar ist. Für DFHSC2LS sollten Sie die Zuordnungsebene 3.0 oder höher verwenden.

- 3.0 Der xsd:dateTime-Datentyp wird dem CICS ASKTIME-Format zugeordnet.
- 4.0 Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher, wenn Sie UTF-16 verwenden möchten.
- 4.1 Zur Unterstützung abschneidbarer Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher.
- 4.2 Für zusätzliche Eigenschaften verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.
- 4.3 Zur Unterstützung mehrdimensionaler Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERScores | NO-ARRAY-NAME-INDEXING | UNDERScores-AS-HYPHENS }

Gibt an, ob das Standardverhalten für die angegebene Zuordnungsebene beim Generieren von Sprachstrukturen überschrieben wird.

SAME-AS-MAPPING-LEVEL

Dieser Parameter generiert Sprachstrukturen auf dieselbe Weise wie die Zuordnungsebene. Dies ist die Standardeinstellung.

HYPHENS-AS-UNDERScores

Nur für PL/I. Dieser Parameter konvertiert alle Bindestriche im XML-Dokument in Unterstriche und nicht in das Zeichen X, um die Lesbarkeit der generierten PL/I-Sprachstrukturen zu verbessern. Weitere Informationen finden Sie unter XML schema to PL/I mapping. Diese Option ist auf Zuordnungsebene 4.2 automatisch aktiviert.

NO-ARRAY-NAME-INDEXING

Nur für COBOL und Enterprise PL/I. Stellt sicher, dass die Feldnamen in einem Array nur im Rahmen der übergeordneten Struktur eindeutig sind.

UNDERScores-AS-HYPHENS

Nur für COBOL. Konvertiert alle Unterstriche im XML-Dokument in Bindestriche und nicht in das Zeichen X. Diese Option verbessert die Lesbarkeit der generierten COBOL-Sprachstrukturen. Im Fall von Feldnamenskollisionen werden die Felder nummeriert, um sicherzustellen, dass sie eindeutig sind. Weitere Informationen finden Sie unter „Zuordnung von XML-Schema zu COBOL“ auf Seite 344.

Diese Option ist auf Zuordnungsebene 4.0 automatisch aktiviert.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 3.0 | 4.1 | 4.2 | 4.3 | CURRENT }

Gibt die älteste CICS-Laufzeitumgebung an, in der die XML-Bindung implementiert werden kann. Wenn Sie eine Version auswählen, die nicht mit den anderen Parametern übereinstimmt, die Sie angegeben haben, erhalten Sie eine Fehlermeldung. Sie können die folgenden Optionen auswählen:

MINIMUM

Die niedrigste mögliche Laufzeitebene von CICS wird anhand der von Ihnen ausgewählten Parameter automatisch zugewiesen.

- 3.0 Geben Sie die Laufzeitebene 3.0 oder höher an, wenn Sie den CICS-XML-Assistenten verwenden und von fortschrittlichen Datenzuordnungen profitieren möchten.
- 4.0 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungs-

ebene von 4.0 oder niedriger verwenden. Sie können alle optionalen Parameter auf dieser Ebene verwenden.

- 4.1 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.1 oder niedriger verwenden.
- 4.2 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS V5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.2 oder niedriger verwenden.
- 4.3 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.3 oder niedriger verwenden.

CURRENT

Verwenden Sie diese Laufzeitebene, um die generierte XML-Bindung in einer CICS-Region zu implementieren, die dieselbe Laufzeitumgebung wie die Region hat, die zum Generieren der Bindungsdatei verwendet wurde.

OVERWRITE-OUTPUT = NO | YES

Steuert, ob vorhandene CICS-BUNDLES im Dateisystem überschrieben werden können.

NO Vorhandene BUNDLES werden nicht ersetzt. Wenn ein vorhandenes BUNDLE gefunden wird, gibt DFHLS2SC die Fehlermeldung DFHPI9689E aus und wird beendet.

YES Vorhandene BUNDLES werden ersetzt. Wenn ein vorhandenes BUNDLE gefunden wird, gibt DFHLS2SC die Nachricht DFHPI9683W aus, um Sie darüber zu informieren, dass die Datei ersetzt wurde.

PDSCP = *wert*

Gibt die Codepage an, die in den Members der partitionierten Datei verwendet wird, wobei *wert* eine CCSID-Nummer oder eine Java-Codepagenummer ist. Wenn dieser Parameter nicht angegeben wird, wird die z/OS UNIX System Services-Codepage verwendet. Beispielsweise können Sie PDSCP=037 angeben.

PDSLIB = *wert*

Gibt den Namen der partitionierten Datei an, die die generierte höhere Programmiersprache enthält.

PDSMEM = *wert*

Gibt das ein bis sechs Zeichen lange Präfix an, das DFHSC2LS verwendet, um den Namen des Members der partitionierten Datei zu generieren, die die Strukturen der höheren Programmiersprachen enthalten wird.

DFHSC2LS generiert ein Member einer partitionierten Datei pro Operation. Der Membername wird durch das Anhängen einer zweistelligen Zahl an das Präfix generiert.

SCHEMA = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, aus der das XML-Schema gelesen wird. DFHSC2LS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur.

Sie können entweder ein XML-Schema oder ein WSDL-Dokument als Eingabe für DFHSC2LS verwenden. Sie müssen entweder diesen Parameter oder den Parameter **WSDL** angeben, aber nicht beide, um anzuzeigen, woher die Eingabe stammt.

STRUCTURE = { PDSMEM-wert | *daten* }

Der Name der übergeordneten Datenstruktur in C und C++. Der Standardwert ist der Wert des **PDSMEM**-Parameters.

TYPES = *wert*

Definiert eine Liste lokaler Namen von globalen Typen, die aktiviert werden sollen. Der *wert* ALL gibt an, dass alle globalen Typen aktiviert werden. Standardmäßig sind globale Typen nicht aktiviert.

WIDE-COMP3 = { FULL | NO | YES }

Steuert die maximale Länge der gepackten Dezimalvariablen in der generierten COBOL- oder PL/I-Sprachstruktur.

FULL Für COBOL und PL/I generiert DFHJS2LS ein gepacktes Dezimalfeld, das groß genug ist, um alle gültigen Werte aufzunehmen. Die maximale Länge sind 31 Ziffern. Dies ist die Standardeinstellung.

NO Nur für COBOL. DFHJS2LS begrenzt die Länge der gepackten Dezimalvariablen auf 18 Ziffern, wenn der COBOL-Sprachstrukturtyp COMP-3 generiert wird. Wenn die gepackte Dezimalzahl länger als 18 Ziffern ist, wird die Nachricht DFHPI9022W ausgegeben, um darauf hinzuweisen, dass der angegebene Typ auf höchstens 18 Ziffern begrenzt ist.

YES Nur für COBOL. DFHJS2LS unterstützt die maximale Länge von 31 Ziffern, wenn der COBOL-Sprachstrukturtyp COMP-3 generiert wird.

Anmerkung: Die Optionen NO und YES generieren Felder, die nicht alle gültigen Werte darstellen können. Mit der Option FULL wird dieses Problem behoben. Allerdings lässt die Option FULL auch die Darstellung einiger ungültiger Werte in dem gepackten Dezimalfeld zu. Wenn ein Schema beispielsweise angibt, dass es maximal fünf Ziffern und maximal zwei Nachkommastellen gibt, generiert die Option FULL ein gepacktes Dezimalfeld, das sieben Ziffern zulässt, wodurch Platz für gültige Werte wie 25000 und 999,99 verfügbar ist, aber auch genug Platz für ungültige Werte wie 9999,99. Wenn Sie die Option FULL verwenden, achten Sie darauf, keine ungültigen Werte in Anwendungsdaten zu generieren.

WSDL = *wert*

Der vollständig qualifizierte z/OS UNIX-Name des WSDL-Dokuments.

Sie können entweder ein XML-Schema oder ein WSDL-Dokument als Eingabe für DFHSC2LS verwenden. Sie müssen entweder diesen Parameter oder den Parameter **SCHEMA** angeben, aber nicht beide, um anzuzeigen, woher die Eingabe stammt.

XMLCP = { LOCAL | UTF-8 | EBCDIC-CP-US }

Gibt die Codepage an, die zum Generieren der XML-Bindung verwendet wird.

LOCAL

Dies ist der Standardwert. Er gibt an, dass die XML mithilfe der lokalen Codepage generiert wird und dass kein Codierungstag im XML-Schema generiert wird.

UTF-8 Gibt an, dass die XML unter Verwendung der UTF-8-Codepage generiert wird. Ein Codierungstag wird im XML-Schema generiert. Wenn

Sie diese Option angeben, müssen Sie sicherstellen, dass die Codierung weiterhin korrekt ist, wenn Sie das XML-Schema zwischen verschiedenen Plattformen kopieren.

EBCDIC-CP-US

Gibt an, dass die XML unter Verwendung der US EBCDIC-Codepage generiert wird. Ein Codierungstag wird im XML-Schema generiert.

XSDBIND = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der XML-Bindung. DFHSC2LS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur. Die Dateierweiterung ist `.xsdbind`.

Zuordnung einer höheren Programmiersprache zu einem XML-Schema

Mit den CICS-Assistenten können Sie Zuordnungen zwischen Strukturen einer höheren Programmiersprache und XML-Schemas oder WSDL-Dokumenten generieren. Die CICS-Assistenten generieren auch XML-Schemas oder WSDL-Dokumente aus Datenstrukturen einer höheren Programmiersprache oder umgekehrt.

Die Dienstprogramme DFHSC2LS und DFHLS2SC werden gesammelt als CICS-XML-Assistent bezeichnet. Die Dienstprogramme DFHWS2LS und DFHLS2WS werden gesammelt als CICS-Web-Service-Assistent bezeichnet.

- DFHLS2SC und DFHLS2WS ordnen Strukturen höherer Programmiersprachen XML-Schemas und WSDL-Dokumenten zu.
- DFHSC2LS und DFHWS2LS ordnen XML-Schemas und WSDL-Dokumente Strukturen höherer Programmiersprachen zu.

Die beiden Zuordnungen sind nicht symmetrisch:

- Wenn Sie eine Sprachdatenstruktur mit DFHLS2SC oder DFHLS2WS verarbeiten und dann das resultierende XML-Schema bzw. WSDL-Dokument mit dem ergänzenden Dienstprogramm (DFHSC2LS bzw. DFHWS2LS) verarbeiten, stimmt die finale Datenstruktur wahrscheinlich nicht mit dem Original überein. Die finale Datenstruktur ist aber logisch äquivalent zum Original.
- Wenn Sie ein XML-Schema oder WSDL-Dokument mit DFHSC2LS oder DFHWS2LS verarbeiten und dann die resultierende Sprachstruktur mit dem ergänzenden Dienstprogramm (DFHLS2SC bzw. DFHLS2WS) verarbeiten, stimmt das finale XML-Schema oder WSDL-Dokument wahrscheinlich nicht mit dem Original überein.
- In manchen Fällen generieren DFHSC2LS und DFHWS2LS Sprachstrukturen, die nicht von DFHLS2SC und DFHLS2WS unterstützt werden.

Sie müssen Sprachstrukturen, die von DFHLS2SC und DFHLS2WS verarbeitet werden, entsprechend den Regeln der Sprache codieren, die in den von CICS unterstützten Sprachcompilern implementiert sind.

Einschränkungen der Datenzuordnung bei Verwendung der CICS-Assistenten:

CICS unterstützt bidirektionale Datenzuordnungen zwischen Strukturen höherer Programmiersprachen und XML-Schemas bzw. WSDL-Dokumenten, die mit WSDL Version 1.1 oder 2.0 konform sind. Es gibt jedoch Einschränkungen. Diese Einschränkungen gelten nur für die DFHWS2LS- und DFHSC2LS-Tools und variieren je nach Zuordnungsebene.

Einschränkungen auf allen Zuordnungsebenen

- Es werden nur SOAP-Bindungen unterstützt, die Literalcodierung verwenden. Daher müssen Sie das Attribut `use` auf den Wert `literal` setzen. `use="encoded"` wird nicht unterstützt.
- Datentypdefinitionen müssen mit XSD (XML Schema Definition) verschlüsselt werden. Im Schema müssen Datentypen, die in der SOAP-Nachricht verwendet werden, explizit deklariert werden.
- Die Länge mancher Schlüsselwörter in der Web-Service-Beschreibung ist begrenzt. Operations-, Bindungs- und Teilnamen dürfen beispielsweise nicht länger als 255 Zeichen sein. In einigen Fällen kann die maximale Länge des Operationsnamens geringfügig kürzer sein.
- Alle SOAP-Fehler, die in der Web-Service-Beschreibung definiert sind, werden ignoriert. Wenn Sie möchten, dass eine Service-Provider-Anwendung eine SOAP-Fehlernachricht sendet, verwenden Sie den Befehl **EXEC CICS SOAPFAULT**.
- DFHWS2LS und DFHSC2LS unterstützen nur ein einziges `<xsd:any>`-Element in einem bestimmten Geltungsbereich. Das folgende Schemafragment wird beispielsweise nicht unterstützt:

```
<xsd:sequence>
<xsd:any/>
<xsd:any/>
</xsd:sequence>
```

Hier kann `<xsd:any>` bei Bedarf `minOccurs` und `maxOccurs` angeben. Das folgende Schemafragment wird beispielsweise unterstützt:

```
<xsd:sequence>
<xsd:any minOccurs="2" maxOccurs="2"/>
</xsd:sequence>
```

- Zyklische Referenzen werden nicht unterstützt. Zum Beispiel wenn Typ A den Typ B enthält, der wiederum einen Typ A enthält.
- Eine Wiederholung wird in Gruppenelementen, z. B. `<xsd:choice>`, `<xsd:sequence>`, `<xsd:group>` oder `<xsd:all>`, nicht unterstützt. Das folgende Schemafragment wird beispielsweise nicht unterstützt:

```
<xsd:choice maxOccurs="2">
<xsd:element name="name1" type="string"/>
</xsd:choice>
```

Die Ausnahme besteht auf Zuordnungsebene 2.1 und höher, wo `maxOccurs="1"` und `minOccurs="0"` für diese Elemente unterstützt werden.

- DFHSC2LS und DFHWS2LS unterstützen keine Datentypen und Elemente in der SOAP-Nachricht, die von den deklarierten Datentypen und Elementen im XML-Schema abgeleitet werden, entweder aus dem Attribut `xsi:type` oder aus einer Substitutionsgruppe, außer auf Zuordnungsebene 2.2 und höher, wenn das übergeordnete Element oder der übergeordnete Typ als abstrakt definiert ist.
- Eingebettete `<xsd:sequence>`- und `<xsd:group>`-Elemente in einem `<xsd:choice>`-Element werden unterhalb der Zuordnungsebene 2.2 nicht unterstützt. Eingebettete `<xsd:choice>`- und `<xsd:all>`-Elemente in einem `<xsd:choice>`-Element werden nie unterstützt.

Verbesserte Unterstützung auf Zuordnungsebene 1.1 und höher

Auf Zuordnungsebene 1.1 oder höher bietet DFHWS2LS Unterstützung für die folgenden XML-Elemente und -Elementtypen:

- Das `<xsd:list>`-Element
- Das `<xsd:union>`-Element

- Den `xsd:anySimpleType`-Typ.
- Das `<xsd:attribute>`-Element. Auf Zuordnungsebene 1.0 wird dieses Element ignoriert.

Verbesserte Unterstützung auf Zuordnungsebene 2.1 und höher

Auf Zuordnungsebene 2.1 oder höher unterstützt DFHWS2LS die folgenden XML-Elemente und -Elementattribute:

- Das `<xsd:any>`-Element.
- Den `xsd:anyType`-Typ.
- Abstrakte Elemente. Auf niedrigeren Zuordnungsebenen werden abstrakte Elemente nur als Nichtterminaltypen in einer Vererbungshierarchie unterstützt.
- Die Attribute `maxOccurs` und `minOccurs` für die `<xsd:all>`-, `<xsd:choice>`- und `<xsd:sequence>`-Elemente, nur wenn `maxOccurs="1"` und `minOccurs="0"`.
- "FILLER"-Felder in COBOL und "*" -Felder in PL/I werden unterdrückt. Die Felder werden nicht in der generierten WSDL angezeigt und es wird eine entsprechende Lücke in den Datenstrukturen zur Laufzeit beibehalten.

Verbesserte Unterstützung auf Zuordnungsebene 2.2 und höher

Auf Zuordnungsebene 2.2 oder höher bieten DFHSC2LS und DFHWS2LS eine verbesserte Unterstützung für das `<xsd:choice>`-Element, indem maximal 255 Optionen im `<xsd:choice>`-Element unterstützt werden. Weitere Informationen zur Unterstützung für `<xsd:choice>` finden Sie unter „Unterstützung für `<xsd:choice>`“ auf Seite 381.

Auf Zuordnungsebene 2.2 und höher unterstützen die CICS-Assistenten die folgenden XML-Zuordnungen:

- Substitutionsgruppen
- Feste Werte für Elemente
- Abstrakte Datentypen

Eingebettete `<xsd:sequence>`- und `<xsd:group>`-Elemente in einem `<xsd:choice>`-Element werden auf Zuordnungsebene 2.2 und höher unterstützt. Das folgende Schemafragment wird beispielsweise unterstützt:

```
<xsd:choice>
<xsd:element name="name1" type="string"/>
<xsd:sequence/>
</xsd:choice>
```

Wenn das übergeordnete Element oder der übergeordnete Typ in der SOAP-Nachricht als abstrakt definiert ist, unterstützen DFHSC2LS und DFHWS2LS Datentypen und Elemente, die aus den deklarierten Datentypen und Elementen im XML-Schema abgeleitet werden.

Verbesserte Unterstützung auf Zuordnungsebene 3.0 und höher

Auf Zuordnungsebene 3.0 oder höher unterstützen die CICS-Assistenten die folgenden Zuordnungsverbesserungen:

- DFHSC2LS und DFHWS2LS ordnen `xsd:dateTime`-Datentypen dem CICS ASK-TIME-Format zu.
- DFHLS2WS kann ein WSDL-Dokument und eine Web-Service-Bindung aus einer Anwendung erstellen, die nicht nur einen, sondern viele Container verwendet.

- Das Tolerieren von abgeschnittenen Daten, die durch eine Datenstruktur fester Länge beschrieben werden. Sie können dieses Verhalten festlegen, indem Sie den Parameter **DATA-TRUNCATION** in den CICS-Assistenten verwenden.

Verbesserte Unterstützung auf Zuordnungsebene 4.0 und höher

Auf Zuordnungsebene 4.0 oder höher unterstützen die CICS-Assistenten die folgenden Zuordnungsverbesserungen:

Auf Zuordnungsebene 4.0 und höher unterstützen DFHLS2SC und DFHLS2WS die COBOL OCCURS DEPENDING ON-Klausel und die Zuordnung von COBOL-Zeichenarrays in XML-Zeichenfolgen. Sie können dieses Verhalten mithilfe des Parameters **CHAR-OCCURS** in den CICS-Assistenten festlegen.

- Sie müssen den Parameter **DATA-TRUNCATION=ENABLED** angeben.
- Eine komplexe Klausel OCCURS DEPENDING ON wird nicht unterstützt. Diese Einschränkung bedeutet, dass OCCURS DEPENDING ON nur für das letzte Feld einer Struktur verwendet wird.
- CICS unterstützt keine qualifizierten Namen (unter Verwendung des Schlüsselworts 'OF') als Ziel einer OCCURS DEPENDING ON-Klausel, z. B. FIELD1 OF STRUCTURE1.
- CICS unterstützt nicht das Schlüsselwort UNBOUNDED. Sie müssen die maximale Größe der Tabelle angeben, die von der Anwendung erwartet wird.

Auf Zuordnungsebene 4.0 und höher unterstützen CICS-Web-Services die Konvertierung von Anwendungsdaten, die mit UTF-16 Unicode codiert sind.

- Wenn Sie DFHLS2WS oder DFHLS2SC verwenden, können Sie dieses Verhalten aktivieren, indem Sie sprachspezifische Datentypen für UTF-16 verwenden.
- Wenn Sie DFHWS2LS oder DFHSC2LS verwenden, können Sie dieses Verhalten aktivieren, indem Sie CCSID=1200 festlegen.
- CICS unterstützt nur eine einzelne Unicode-Codepage, „UTF-16BE mit IBM Private Use Area“ (CCSID 1200).
- Die Konvertierung von Anwendungsdaten, die mit UTF-8 codiert sind, wird nicht unterstützt.

Anmerkung: DFHLS2WS und DFHLS2SC bieten keine Unterstützung für die COBOL-Klausel GROUP USAGE NATIONAL.

Zuordnung von COBOL zu XML-Schema:

Die Dienstprogramme DFHLS2SC und DFHLS2WS unterstützen die Zuordnungen zwischen COBOL-Datenstrukturen und XML-Schemadefinitionen.

COBOL-Namen werden entsprechend den folgenden Regeln in XML-Namen konvertiert:

1. Doppelte Namen werden durch Hinzufügen einer oder mehrerer Ziffern eindeutig gemacht.
Zwei Instanzen von year werden zu year und year1.
2. Bindestriche werden durch Unterstriche ersetzt. Aufeinanderfolgende Bindestriche werden durch aufeinanderfolgende Unterstriche ersetzt.
Beispiel: current-user--id wird zu current_user__id.
3. Segmente von Namen, die durch Bindestriche begrenzt sind und nur Großbuchstaben enthalten, werden in Kleinbuchstaben konvertiert.
Beispiel: CA-REQUEST-ID wird zu ca_request_id.

4. Ein Unterstrich wird vor Namen hinzugefügt, die mit einer Ziffer beginnen.
Beispiel: 9A-REQUEST-ID wird zu `_9a_request_id`.

CICS ordnet COBOL-Datenbeschreibungselemente entsprechend der folgenden Tabelle Schemaelementen zu. COBOL-Datenbeschreibungselemente, die in der Tabelle nicht enthalten sind, werden von DFHLS2SC oder DFHLS2WS nicht unterstützt. Es gelten außerdem die folgenden Einschränkungen:

- Datenbeschreibungselemente mit den Ebenennummern 66 und 77 werden nicht unterstützt. Datenbeschreibungselemente mit der Ebenennummer 88 werden ignoriert.
- Die folgenden Klauseln in Datenbeschreibungseinträgen werden nicht unterstützt:
 - REDEFINES
 - RENAMES; d. h. Ebene 66
 - DATE FORMAT
- Die folgenden Klauseln in Datenbeschreibungselementen werden ignoriert:
 - BLANK WHEN ZERO
 - JUSTIFIED
 - VALUE
- Die SIGN-Klausel SIGN TRAILING wird unterstützt. Die SIGN-Klausel SIGN LEADING wird nur unterstützt, wenn in DFHLS2SC oder DFHLS2WS eine Zuordnungsebene 1.2 oder höher angegeben ist.
- SEPARATE CHARACTER wird sowohl für SIGN TRAILING- als auch für SIGN LEADING-Klauseln auf Zuordnungsebene 1.2 oder höher unterstützt.
- Die folgenden Phrasen in der USAGE-Klausel werden nicht unterstützt:
 - OBJECT REFERENCE
 - POINTER
 - FUNCTION-POINTER
 - PROCEDURE-POINTER
- Die folgenden Phrasen in der USAGE-Klausel werden auf Zuordnungsebene 1.2 oder höher unterstützt:
 - COMPUTATIONAL-1
 - COMPUTATIONAL-2
- Die einzigen PICTURE-Zeichen, die für DISPLAY- und COMPUTATIONAL-5-Datenbeschreibungselemente unterstützt werden, sind 9, S und Z.
- Die PICTURE-Zeichen, die für PACKED-DECIMAL-Datenbeschreibungselemente unterstützt werden, sind 9, S, V und Z.
- Die einzigen PICTURE-Zeichen, die für bearbeitete numerische Datenbeschreibungselemente unterstützt werden, sind 9 und Z.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL festgelegt ist, werden Zeichenarrays einem `xsd:string`-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf BINARY festgelegt ist, werden Zeichenarrays einem `xsd:base64Binary`-Element zugeordnet und als binäre Daten verarbeitet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf COLLAPSE festgelegt ist, werden Leerzeichen am Ende von Zeichenfolgen ignoriert.

- Die Klausel OCCURS DEPENDING ON wird auf Zuordnungsebene 4.0 oder höher unterstützt. Eine komplexe Klausel OCCURS DEPENDING ON wird nicht unterstützt. Dies bedeutet, dass OCCURS DEPENDING ON nur für das letzte Feld einer Struktur unterstützt wird.
- Die Klausel OCCURS INDEXED BY wird auf jeder Zuordnungsebene unterstützt.
- Die Klausel OCCURS wird für bis zu 65535 Vorkommen (TIMES) unterstützt. Das bedeutet: OCCURS *n* TIMES . Wenn *n* größer als 65535 ist, wird die Klausel nicht unterstützt.

COBOL-Datenbeschreibung	Schema 'simpleType'
<div> <div><i>n</i></div> <div>PIC X(</div> <div>)</div> </div> <div> <div><i>n</i></div> <div>PIC A(</div> <div>)</div> </div> <div> <div><i>n</i></div> <div>PIC G(</div> <div>) DISPLAY-1</div> </div> <div> <div><i>n</i></div> <div>PIC N(</div> <div>)</div> </div>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value=" <i>n</i>"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType> </pre>
<div> <div>PIC S9 DISPLAY</div> <div>PIC S99 DISPLAY</div> <div>PIC S999 DISPLAY</div> <div>PIC S9999 DISPLAY</div> </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>Dabei ist <i>n</i> der Höchstwert, der durch ein Muster von '9'-Zeichen dargestellt werden kann.</p>
<div> <div><i>z</i></div> <div>PIC S9(</div> <div>) DISPLAY</div> </div> <p>where $5 \leq z \leq 9$</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:int"> <xsd:minInclusive value="- <i>n</i>"/> <xsd:maxInclusive value=" <i>n</i>"/> </xsd:restriction> </xsd:simpleType> </pre> <p>Dabei ist <i>n</i> der Höchstwert, der durch ein Muster von '9'-Zeichen dargestellt werden kann.</p>

COBOL-Datenbeschreibung	Schema 'simpleType'
PIC S9(z) DISPLAY Dabei ist $9 < z$.	<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> <xsd:minInclusive value="- n"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>Dabei ist n der Höchstwert, der durch ein Muster von '9'-Zeichen dargestellt werden kann.</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>Dabei ist n der Höchstwert, der durch ein Muster von '9'-Zeichen dargestellt werden kann.</p>
PIC 9(z) DISPLAY Dabei ist $5 \leq z \leq 9$.	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>Dabei ist n der Höchstwert, der durch ein Muster von '9'-Zeichen dargestellt werden kann.</p>
PIC 9(z) DISPLAY Dabei ist $9 < z$.	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>Dabei ist n der Höchstwert, der durch ein Muster von '9'-Zeichen dargestellt werden kann.</p>
PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY Dabei ist $n \leq 4$.	<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>

COBOL-Datenbeschreibung	Schema 'simpleType'
<pre> PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY </pre> <p>Dabei ist $5 \leq n \leq 9$.</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType> </pre>
<pre> PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY </pre> <p>Dabei ist $9 < n$.</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType> </pre>
<pre> PIC 9(n) COMP PIC 9(n) COMP-4 PIC 9(n) COMP-5 PIC 9(n) BINARY </pre> <p>Dabei ist $n \leq 4$.</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType> </pre>

COBOL-Datenbeschreibung	Schema 'simpleType'
PIC 9(n) COMP PIC 9(n) COMP-4 PIC 9(n) COMP-5 PIC 9(n) BINARY Dabei ist $5 \leq n \leq 9$.	<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 Dabei ist $9 < n$.	<pre> <xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType> </pre>
PIC S9(m)V9(n) COMP-3 Dabei ist $p = m + n$.	<pre> <xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" p" /> <xsd:fractionDigits value=" n" /> </xsd:restriction> </xsd:simpleType> </pre> <p>Dabei ist $p = m + n$.</p>

COBOL-Datenbeschreibung	Schema 'simpleType'
<p>PIC 9(<i>m</i>)V9(<i>n</i>) COMP-3</p> <p>PIC S9(<i>m</i>) COMP-3</p> <p>Unterstützt auf Zuordnungsebene 3.0, wenn DATETIME=PACKED15</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" <i>p</i>"/> <xsd:fractionDigits value=" <i>n</i>"/> <xsd:minInclusive value="0"/> </xsd:restriction> </xsd:simpleType> </pre> <p>Dabei ist $p = m + n$.</p> <pre> <xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType> </pre> <p>Das Format der Zeitmarke ist CICS ABSTIME.</p>
<p>PIC S9(<i>m</i>)V9(<i>n</i>) DISPLAY</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" <i>p</i>"/> <xsd:fractionDigits value=" <i>n</i>"/> </xsd:restriction> </xsd:simpleType> </pre> <p>Dabei ist $p = m + n$.</p>
<p>COMP-1</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:float-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-1-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType> </pre>

COBOL-Datenbeschreibung	Schema 'simpleType'
<p>COMP-2</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:double-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-2-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>
<p><i>datenbeschreibung</i> OCCURS <i>n</i> TIMES</p>	<pre><xsd:element name= "field-name" minOccurs= "n" maxOccurs= "n"> ... </xsd:element></pre> <p>Der Inhalt des Elements hängt vom verwendeten Datentyp ab.</p>
<p><i>datenbeschreibung</i> OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDENT ON <i>t</i></p> <p>Unterstützt auf Zuordnungsebene 4.0</p>	<pre><xsd:element name= "field-name" minOccurs= "n" maxOccurs= "m"> ... </xsd:element></pre>

COBOL-Datenbeschreibung	Schema 'simpleType'
<pre> PIC X OCCURS n TIMES PIC A OCCURS n TIMES PIC G DISPLAY-1 OCCURS n TIMES PIC N OCCURS n TIMES </pre>	<p>Wenn 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>Dies ist eine Zeichenfolge.</p> <p>Wenn 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>Dies ist ein Array einzelner Zeichen.</p>

COBOL-Datenbeschreibung	Schema 'simpleType'
<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>Wenn CHAR-OCCURS =STRING :</p> <pre> <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> </pre>
<pre> PIC N(n) USAGE NATIONAL </pre> <p>Wenn CHAR-USAGE =NATIONAL : PIC N(n)</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxlength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType> </pre> <p>Zur Laufzeit füllt CICS das Anwendungsdatenstrukturfeld mit UTF-16-Daten.</p>

Zuordnung von XML-Schema zu COBOL:

Die Dienstprogramme DFHSC2LS und DFHWS2LS unterstützen Zuordnungen zwischen XML-Schemadefinitionen und COBOL-Datenstrukturen.

Die CICS-Assistenten generieren eindeutige und gültige Namen für COBOL-Variablen aus den Schemaelementnamen mithilfe der folgenden Regeln:

1. Reservierten COBOL-Wörtern wird das Präfix X vorangestellt.
Beispiel: DISPLAY wird zu XDISPLAY.
2. Alle Zeichen außer A-Z, a-z, 0-9 oder Bindestrich werden durch X ersetzt.
Beispiel: monthly_total wird zu monthlyXtotal. Sie können den Parameter **MAPPING-OVERRIDES** verwenden, um anzupassen, wie andere Zeichen behandelt

werden. Wenn Sie beispielsweise den Wert `UNDERSCORES-AS-HYPHENS` festlegen, werden alle Unterstriche in der XML statt in ein X in Bindestriche konvertiert. `monthly_total` wird folglich zu `monthly-total`.

3. Wenn das letzte Zeichen ein Bindestrich ist, wird es durch X ersetzt.

Beispiel: `ca-request-` wird zu `ca-requestX`.

4. Wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat (d. h. `minOccurs` und `maxOccurs` werden mit unterschiedlichen Werten in `xsd:element` angegeben), und wenn der Name des Schemaelements länger als 23 Zeichen ist, wird er auf diese Länge abgeschnitten.

Wenn das Schema angibt, dass die Variable eine feste Kardinalität hat und wenn der Name des Schemaelements länger als 28 Zeichen ist, wird er auf diese Länge abgeschnitten.

5. Doppelte Namen in demselben Bereich werden durch das Hinzufügen von ein oder zwei Ziffern zu der zweiten Instanz und zu allen nachfolgenden Instanzen des Namens eindeutig gemacht.

Beispiel: Drei Instanzen von `year` werden zu `year`, `year1` und `year2`.

Sollte das oben beschriebene Verhalten unerwünscht sein, kann der Benutzer `MAPPING-OVERRIDES=NO-ARRAY-NAME-INDEXING` als Eingabe für das Dienstprogramm angeben, die das Hinzufügen einer oder mehrerer Ziffern zur zweiten Instanz und allen nachfolgenden Instanzen des Namens inaktiviert.

6. Fünf Zeichen sind reserviert für die Zeichenfolgen `-cont` oder `-num`, die verwendet werden, wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat, d. h. wenn `minOccurs` und `maxOccurs` mit unterschiedlichen Werten angegeben werden.

Weitere Informationen finden Sie unter „Variable Arrays von Elementen“ auf Seite 372.

7. Für Attribute werden die vorherigen Regeln auf den Elementnamen angewendet. Das Präfix `attr-` wird dem Elementnamen hinzugefügt, gefolgt von `-value` oder `-exist`. Wenn die Gesamtlänge mehr als 28 Zeichen beträgt, wird der Elementname abgeschnitten. Weitere Informationen finden Sie unter „Unterstützung für XML-Attribute“ auf Seite 377.

Für das auf Nil festlegbare Attribut gelten spezielle Regeln. Das Präfix `attr-` wird hinzugefügt, aber `nil-` wird ebenfalls am Anfang des Elementnamens hinzugefügt. Auf den Elementnamen folgt `-value`. Wenn die Gesamtlänge mehr als 28 Zeichen beträgt, wird der Elementname abgeschnitten.

Der resultierende Name ist höchstens 30 Zeichen lang.

DFHSC2LS und DFHWS2LS ordnen Schematypen zu COBOL-Datenbeschreibungselementen mithilfe der angegebenen Zuordnungsebene in der folgenden Tabelle zu. Beachten Sie die folgenden Punkte:

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL gesetzt ist, werden Zeichendaten mit variabler Länge auf null endenden Zeichenfolgen zugeordnet, und ein zusätzliches Zeichen wird dem Nullabschlusszeichen zugeordnet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf YES gesetzt ist, werden Zeichendaten mit variabler Länge zwei verwandten Elementen zugeordnet: einem Längenfeld und einem Datenfeld.

Beispiel:

```
<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"/>

```

Zuordnung zu:

```

15 textString-length PIC S9999 COMP-5 SYNC
15 textString PIC X(10000)

```

Einfacher Schematyp	COBOL-Datenbeschreibung
<pre> <xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Zuordnungsebene 2.0 und niedriger:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 2.1:</p> <p>Unterstützt</p>
<pre> <xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Zuordnungsebene 1.0:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 1.1 und höher:</p> <p>PIC X(255)</p>
<pre> <xsd:simpleType> <xsd:restriction base="xsd: typ" <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType> </pre> <p>Dabei nimmt <i>typ</i> einen der folgenden Werte an:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY hexBinary 	<p>Alle Zuordnungsebenen:</p> <p>PIC X(</p> <p>z</p> <p>)</p>

Einfacher Schematyp	COBOL-Datenbeschreibung
<pre><xsd:simpleType> <xsd:restriction base="xsd: typ"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre> <p>Dabei nimmt <i>typ</i> einen der folgenden Werte an:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>PIC N(^z) USAGE NATIONAL</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: typ" </xsd:restriction> </xsd:simpleType></pre> <p>Dabei nimmt <i>typ</i> einen der folgenden Werte an:</p> <ul style="list-style-type: none"> duration date time gDay gMonth gYear gMonthDay gYearMonth 	<p>Alle Zuordnungsebenen:</p> <p>PIC X(32)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre>	<p>Zuordnungsebene 1.2 und niedriger:</p> <p>PIC X(32)</p> <p>Zuordnungsebene 2.0 und höher:</p> <p>PIC X(40)</p> <p>Zuordnungsebene 3.0 und höher:</p> <p>PIC S9(15) COMP-3</p> <p>Das Format ist CICS ABSTIME.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: typ"> </xsd:restriction> </xsd:simpleType></pre> <p>Dabei nimmt <i>typ</i> einen der folgenden Werte an:</p> <ul style="list-style-type: none"> byte unsignedByte 	<p>Alle Zuordnungsebenen:</p> <p>PIC X DISPLAY</p>

Einfacher Schematyp	COBOL-Datenbeschreibung
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>PIC S9999 COMP-5 SYNC</p> <p>oder</p> <p>PIC S9999 DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>PIC 9999 COMP-5 SYNC</p> <p>oder</p> <p>PIC 9999 DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>PIC S9(18) COMP-3</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>PIC S9(9) COMP-5 SYNC</p> <p>oder</p> <p>PIC S9(9) DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>PIC 9(9) COMP-5 SYNC</p> <p>oder</p> <p>PIC 9(9) DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>PIC S9(18) COMP-5 SYNC</p> <p>oder</p> <p>PIC S9(18) DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>PIC 9(18) COMP-5 SYNC</p> <p>oder</p> <p>PIC 9(18) DISPLAY</p>

Einfacher Schematyp	COBOL-Datenbeschreibung
<pre> <xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" m" <xsd:fractionDigits value=" n" </xsd:restriction> </xsd:simpleType> </pre>	<p>Alle Zuordnungsebenen:</p> <p>Wenn WIDE-COMP3=FULL:</p> <pre> PIC 9(m)V9(n) COMP-3 </pre> <p>Andernfalls:</p> <pre> PIC 9(p)V9(n) COMP-3 </pre> <p>Dabei ist $p = m - n$.</p>
<pre> <xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Alle Zuordnungsebenen:</p> <pre> PIC X DISPLAY x'00' impliziert 'false', x'01' impliziert 'true'. </pre>
<pre> <xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType> </pre>	<p>Zuordnungsebene 1.0:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 1.1 und höher:</p> <pre> PIC X(255) </pre>
<pre> <xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType> </pre>	<p>Zuordnungsebene 1.0:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 1.1 und höher:</p> <pre> PIC X(255) </pre>

Einfacher Schematyp	COBOL-Datenbeschreibung
<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>Dabei ist die Länge nicht definiert.</p>	<p>Zuordnungsebene 1.0:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 1.1:</p> <p style="text-align: right;">PIC X(</p> <p style="text-align: center;">y</p> <p style="text-align: center;">)</p> <p>Dabei ist $y = 4 \times (\text{ceil}(z / 3))$. $\text{ceil}(x)$ ist die kleinste ganze Zahl größer-gleich x.</p> <p>Zuordnungsebene 1.2 und höher:</p> <p style="text-align: right;">PIC X(</p> <p style="text-align: center;">z</p> <p style="text-align: center;">)</p> <p>Dabei ist die Länge festgelegt.</p> <p style="text-align: right;">PIC X(16)</p> <p>Dabei ist die Länge nicht definiert. Das Feld enthält den 16-Byte-Namen des Containers, der die binären Daten speichert.</p>
<pre> <xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p style="text-align: right;">PIC X(32)</p> <p>Zuordnungsebene 1.2 und höher:</p> <p style="text-align: right;">COMP-1</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:float-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-1-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

Einfacher Schematyp	COBOL-Datenbeschreibung
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p>PIC X(32)</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>COMP-2</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:double-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-2-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

Einige der Schematypen, die in der Tabelle aufgeführt sind, werden dem COBOL-Format COMP-5 SYNC oder DISPLAY zugeordnet, abhängig von den Werten, die ggf. in den Facetten minInclusive und maxInclusive angegeben sind:

- Für Typen mit Vorzeichen (short , int und long) wird DISPLAY verwendet, wenn Folgendes angegeben ist:

```
<xsd:minInclusive value="-
a"/>
<xsd:maxInclusive value="
a
"/>
```

Dabei ist *a* eine aus Neunen bestehende Zeichenfolge.

- Für Typen ohne Vorzeichen (unsignedShort , unsignedInt und unsignedLong) wird DISPLAY verwendet, wenn Folgendes angegeben ist:

```
<xsd:minInclusive value="0"/>
<xsd:maxInclusive value="
a
"/>
```

Dabei ist *a* eine aus Neunen bestehende Zeichenfolge.

Ist ein anderer Wert oder kein Wert angegeben, wird COMP-5 SYNC verwendet.

Zuordnung von C und C++ zu XML-Schema:

Die Dienstprogramme DFHLS2SC und DFHLS2WS unterstützen die Zuordnungen zwischen C- und C++-Datentypen und XML-Schemadefinitionen.

C- und C++-Namen werden entsprechend den folgenden Regeln in XML-Namen konvertiert:

- Zeichen, die in XML-Elementnamen nicht gültig sind, werden durch X ersetzt. Beispiel: monthly-total wird zu monthlyXtotal.

2. Doppelte Namen werden durch Hinzufügen einer oder mehrerer Ziffern eindeutig gemacht.

Zwei Instanzen von year werden zu year und year1.

DFHLS2SC und DFHLS2WS ordnen C- und C++-Datentypen Schemaelementen entsprechend der folgenden Tabelle zu. C- und C++-Typen, die in der Tabelle nicht enthalten sind, werden von DFHLS2SC oder DFHLS2WS nicht unterstützt. Das Qualifikationsmerkmal `_Packed` wird für Strukturen unterstützt. Folgende Einschränkungen gelten:

- Headerdateien müssen eine struct-Instanz der höchsten Ebene enthalten.
- Sie können keinen Strukturtyp deklarieren, der sich selbst als Mitglied enthält.
- Die folgenden C- und C++-Datentypen werden nicht unterstützt:
 - `decimal`
 - `long double`
 - `wchar_t` (nur C++)
- Die folgenden Zeichen werden ignoriert, wenn sie in der Headerdatei vorhanden sind.

Speicherklassenkennungen:

- `auto`
- `register`
- `static`
- `extern`
- `mutable`

Qualifikationsmerkmale

- `const`
- `volatile`
- `_Export` (nur C++)

Funktionskennungen

- `inline` (nur C++)
- `virtual` (nur C++)

Anfangswerte

- Die Headerdatei darf die folgenden Werte nicht enthalten:
 - Unionen
 - Klassendeklarationen
 - Aufzählungsdantypen
 - Zeigertypvariablen
 - Schablonendeklarationen
 - Vordefinierte Makros, d. h. Makros mit Namen, die mit zwei Unterstrichen beginnen und enden (`__`)
 - Die Zeilenfortsetzungssequenz (ein `'\'`-Symbol, auf das unmittelbar ein Zeilenvorschubzeichen folgt)
 - Deklaratoren für Prototypfunktionen
 - Vorprozessoranweisungen
 - Bitfelder
 - Das Schlüsselwort `__cdecl` (oder `_cdecl`) (nur C++)
- Der Anwendungsprogrammierer muss einen 32-Bit-Compiler verwenden, um sicherzustellen, dass einem `int`-Wert 4 Bytes zugeordnet werden.
- Die folgenden C++-reservierten Schlüsselwörter werden nicht unterstützt:
 - `explicit`
 - `using`

namespace
 typename
 typeid

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL festgelegt ist, werden Zeichenarrays einem xsd:string-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf BINARY festgelegt ist, werden Zeichenarrays einem xsd:base64Binary-Element zugeordnet und als binäre Daten verarbeitet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf COLLAPSE festgelegt ist, wird <xsd:whiteSpace value="collapse"/> für Zeichenfolgen generiert.

C- und C++-Datentyp	Schema 'simpleType'
char[z]	<pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre>
char16_t[n]	<p>Auf Zuordnungsebene 4.0 und höher:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> <p>Zur Laufzeit füllt CICS das Anwendungsdatenstrukturfeld mit UTF-16-Daten.</p>
char[8] Unterstützt auf Zuordnungsebene 3.0 und höher, wenn DATETIME=PACKED15	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre> <p>Das Format der Zeitmarke ist 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>

C- und C++-Datentyp	Schema 'simpleType'
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 (nur C++)	<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>
float Unterstützt auf Zuordnungsebene 1.2 und höher	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:float-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt Gleitkommatentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>
double Unterstützt auf Zuordnungsebene 1.2 und höher	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:double-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt doppelter Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

Zuordnung von XML-Schema zu C und C++:

Die Dienstprogramme DFHSC2LS und DFHWS2LS unterstützen Zuordnungen zwischen den XML-Schemadefinitionen, die in den einzelnen Web-Service-Beschreibungen enthalten sind, und C- und C++-Datentypen.

Die CICS-Assistenten generieren eindeutige und gültige Namen für C- und C++-Variablen aus den Schemaelementnamen mithilfe der folgenden Regeln:

1. Alle Zeichen außer A-Z, a-z, 0-9 oder _ werden durch X ersetzt.

Beispiel: monthly-total wird zu monthlyXtotal.

2. Wenn das erste Zeichen kein alphabetisches Zeichen ist, wird es durch ein führendes X ersetzt.
Beispiel: `_monthlysummary` wird zu `Xmonthlysummary`.
3. Wenn der Name des Schemaelements länger als 50 Zeichen ist, wird er auf diese Länge abgeschnitten.
4. Doppelte Namen im selben Bereich werden durch Hinzufügen einer oder mehrerer Ziffern eindeutig gemacht.
Zwei Instanzen von `year` werden zu `year` und `year1`.
5. Fünf Zeichen sind reserviert für die Zeichenfolgen `_cont` oder `_num`, die verwendet werden, wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat, d. h. wenn `minOccurs` und `maxOccurs` in einem `xsd:element`-Element angegeben werden.
Weitere Informationen finden Sie unter „Variable Arrays von Elementen“ auf Seite 372.
6. Für Attribute werden die vorherigen Regeln auf den Elementnamen angewendet. Das Präfix `attr_` wird dem Elementnamen hinzugefügt, gefolgt von `_value` oder `_exist`. Wenn die Gesamtlänge mehr als 28 Zeichen beträgt, wird der Elementname abgeschnitten.
Für das auf Nil festlegbare Attribut gelten spezielle Regeln. Das Präfix `attr_` wird hinzugefügt, aber `nil_` wird ebenfalls am Anfang des Elementnamens hinzugefügt. Auf den Elementnamen folgt `_value`. Wenn die Gesamtlänge mehr als 28 Zeichen beträgt, wird der Elementname abgeschnitten.

Der resultierende Name ist höchstens 57 Zeichen lang.

DFHSC2LS und DFHWS2LS ordnen Schematypen zu C- und C++-Datentypen entsprechend der folgenden Tabelle zu. Es gelten außerdem die folgenden Regeln:

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL gesetzt ist, werden Zeichendaten mit variabler Länge auf null endenden Zeichenfolgen zugeordnet, und ein zusätzliches Zeichen wird dem Nullabschlusszeichen zugeordnet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf YES gesetzt ist, werden Zeichendaten mit variabler Länge zwei verwandten Elementen zugeordnet: einem Längenfeld und einem Datenfeld.

Schema 'simpleType'	C- und C++-Datentyp
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	<p>Zuordnungsebene 2.0 und niedriger:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 2.1 und höher:</p> <p>Unterstützt</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	<p>Zuordnungsebene 1.0:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 1.1 und höher:</p> <p><code>char[255]</code></p>

Schema 'simpleType'	C- und C++-Datentyp
<pre> <xsd:simpleType> <xsd:restriction base="xsd: typ"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType> </pre> <p>Dabei nimmt <i>typ</i> einen der folgenden Werte an:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY hexBinary 	<p>Alle Zuordnungsebenen:</p> <pre> char[z] </pre>
<pre> <xsd:simpleType> <xsd:restriction base="xsd: typ"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType> </pre> <p>Dabei nimmt <i>typ</i> einen der folgenden Werte an:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <pre> char16_t[z] </pre>

Schema 'simpleType'	C- und C++-Datentyp
<pre><xsd:simpleType> <xsd:restriction base="xsd: typ"> </xsd:restriction> </xsd:simpleType></pre> <p>Dabei nimmt <i>typ</i> einen der folgenden Werte an:</p> <ul style="list-style-type: none"> duration date decimal time gDay gMonth gYear gMonthDay gYearMonth 	<p>Alle Zuordnungsebenen:</p> <p>char[32]</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	<p>Zuordnungsebene 1.2 und niedriger:</p> <p>char[32]</p> <p>Zuordnungsebene 2.0 und höher:</p> <p>char[40]</p> <p>Zuordnungsebene 3.0 und höher:</p> <p>char[8]</p> <p>Das Format der Zeitmarke ist CICS ABSTIME.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>signed char</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>char</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>short</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>unsigned short</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>char[33]</p>

Schema 'simpleType'	C- und C++-Datentyp
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>int</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>unsigned int</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>long long</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>unsigned long long</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>bool (nur C++) short (nur C)</p>
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	<p>Zuordnungsebene 1.0:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 1.1 und höher:</p> <p>char[255]</p>
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	<p>Zuordnungsebene 1.0:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 1.1 und höher:</p> <p>char[255]</p>

Schema 'simpleType'	C- und C++-Datentyp
<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>Dabei ist die Länge nicht definiert.</p>	<p>Zuordnungsebene 1.1 und niedriger:</p> <pre> char[y] </pre> <p>Dabei ist $y = 4 \times (\text{ceil}(z / 3))$. $\text{ceil}(x)$ ist die kleinste ganze Zahl größer-gleich x.</p> <p>Zuordnungsebene 1.2 und höher:</p> <pre> char[z] </pre> <p>Dabei ist die Länge festgelegt.</p> <pre> char[16] </pre> <p>steht für den Namen des Containers, in dem die binären Daten gespeichert werden, wenn die Länge nicht definiert ist.</p>
<pre> <xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <pre> char[32] </pre> <p>Zuordnungsebene 1.2 und höher:</p> <pre> float(*) </pre> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:float-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt Gleitkommatentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

Schema 'simpleType'	C- und C++-Datentyp
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	<p>Zuordnungsebene 1.0 und niedriger:</p> <p style="text-align: center;">char[32]</p> <p>Zuordnungsebene 1.2 und höher:</p> <p style="text-align: center;">double(*)</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:double-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt doppelter Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

Zuordnung von PL/I zu XML-Schema:

Die Dienstprogramme DFHLS2SC und DFHLS2WS unterstützen die Zuordnungen zwischen PL/I-Datenstrukturen und XML-Schemadefinitionen. Da sich der Enterprise PL/I-Compiler von älteren PL/I-Compilern unterscheidet, werden zwei Sprachoptionen unterstützt: PLI-ENTERPRISE und PLI-OTHER.

PL/I-Namen werden entsprechend den folgenden Regeln in XML-Namen konvertiert:

1. Zeichen, die in XML-Elementnamen nicht gültig sind, werden durch x ersetzt.
Beispiel: monthly\$total wird zu monthlyxtotal.
2. Doppelte Namen werden durch Hinzufügen einer oder mehrerer Ziffern eindeutig gemacht.
Zwei Instanzen von year werden zu year und year1.

DFHLS2SC und DFHLS2WS ordnen PL/I-Datentypen Schemaelementen entsprechend der folgenden Tabelle zu. PL/I-Typen, die in der Tabelle nicht enthalten sind, werden von DFHLS2SC oder DFHLS2WS nicht unterstützt. Es gelten außerdem die folgenden Einschränkungen:

- Datenelemente mit dem Attribut COMPLEX werden nicht unterstützt.
- Datenelemente mit dem Attribut FLOAT werden auf Zuordnungsebene 1.2 oder höher unterstützt. Enterprise PL/I FLOAT IEEE wird nicht unterstützt.
- Die reinen DBCS-Zeichenfolgen VARYING und VARYINGZ werden auf Zuordnungsebene 1.2 oder höher unterstützt.
- Datenelemente, die als DECIMAL(*p* , *q*) angegeben sind, werden nur unterstützt, wenn $p \geq q$.
- Datenelemente, die als BINARY(*p* , *q*) angegeben sind, werden nur unterstützt, wenn $q = 0$.
- Wenn das Attribut PRECISION für ein Datenelement angegeben ist, wird es ignoriert.
- PICTURE-Zeichenfolgen werden nicht unterstützt.

- ORDINAL-Datenelemente werden als FIXED BINARY(7)-Datentypen behandelt.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL festgelegt ist, werden Zeichenarrays einem xsd:string-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet. Diese Zuordnung gilt nicht für Enterprise PL/I.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf BINARY festgelegt ist, werden Zeichenarrays einem xsd:base64Binary-Element zugeordnet und als binäre Daten verarbeitet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf COLLAPSE festgelegt ist, wird `<xsd:whiteSpace value="collapse"/>` für Zeichenfolgen generiert.

DFHLS2SC und DFHLS2WS implementieren die Padding-Algorithmen von PL/I nicht vollständig, deshalb müssen Sie Padding-Bytes in Ihrer Datenstruktur explizit deklarieren. DFHLS2SC und DFHLS2WS geben eine Nachricht aus, wenn sie erkennen, dass Padding-Bytes fehlen. Jede übergeordnete Struktur muss an einer Doppelwortgrenze beginnen und jedes Byte in der Struktur muss der korrekten Grenze zugeordnet werden. Sehen Sie sich dieses Codefragment an:

```
3 FIELD1 FIXED BINARY(7),
3 FIELD2 FIXED BINARY(31),
3 FIELD3 FIXED BINARY(63);
```

In diesem Beispiel gilt Folgendes:

- FIELD1 ist 1 Byte lang und kann an jeder Grenze ausgerichtet werden.
- FIELD2 ist 4 Bytes lang und muss an einer Vollwortgrenze ausgerichtet werden.
- FIELD3 ist 8 Bytes lang und muss an einer Doppelwortgrenze ausgerichtet werden.

Der Enterprise PL/I-Compiler richtet die Felder in der folgenden Reihenfolge aus:

1. FIELD3 wird zuerst ausgerichtet, weil es die stärksten Anforderungen an die Grenze hat.
2. FIELD2 wird an der Vollwortgrenze unmittelbar vor FIELD3 ausgerichtet.
3. FIELD1 wird an der Bytegrenze unmittelbar vor FIELD3 ausgerichtet.

Schließlich fügt der Compiler drei Padding-Bytes unmittelbar vor FIELD1 ein, so dass die gesamte Struktur an einer Vollwortgrenze ausgerichtet wird.

Da DFHLS2WS keine äquivalenten Padding-Bytes einfügt, müssen Sie diese explizit deklarieren, bevor die Struktur von DFHLS2WS verarbeitet wird. Beispiel:

```
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);
```

Alternativ können Sie die Struktur ändern, um alle Felder als nicht ausgerichtet zu deklarieren und die Anwendung erneut zu kompilieren, die die Struktur verwendet. Weitere Informationen zu den Anforderungen an die strukturelle PL/I-Speicherausrichtung finden Sie unter Enterprise PL/I for z/OS product information.

PL/I-Datenbeschreibung	Schema
FIXED BINARY (<i>n</i>) Dabei ist $n \leq 7$.	<code><xsd:simpleType></code> <code><xsd:restriction base="xsd:byte"/></code> <code></xsd:simpleType></code>

PL/I-Datenbeschreibung	Schema
FIXED BINARY (n) Dabei ist $8 \leq n \leq 15$.	<code><xsd:simpleType> <xsd:restriction base="xsd:short"/> </xsd:simpleType></code>
FIXED BINARY (n) Dabei ist $16 \leq n \leq 31$.	<code><xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType></code>
FIXED BINARY (n) Dabei ist $32 \leq n \leq 63$. Restriction: Nur Enterprise PL/I	<code><xsd:simpleType> <xsd:restriction base="xsd:long"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(n) Dabei ist $n \leq 8$. Restriction: Nur Enterprise PL/I	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(n) Dabei ist $9 \leq n \leq 16$. Restriction: Nur Enterprise PL/I	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(n) Dabei ist $17 \leq n \leq 32$. Restriction: Nur Enterprise PL/I	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(n) Dabei ist $33 \leq n \leq 64$. Restriction: Nur Enterprise PL/I	<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>
FIXED DECIMAL(15) Unterstützt auf Zuordnungsebene 3.0 und höher, wenn DATEIME=PACKED15	<code><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></code> Das Format der Zeitmarke ist CICS ABSTIME.
BIT(n) Dabei ist n ein Vielfaches von 8. Andere Werte werden nicht unterstützt.	<code><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="m"/> </xsd:restriction> </xsd:simpleType></code> Dabei ist $m = n / 8$.
CHARACTER(n) VARYING und VARYINGZ werden auch auf Zuordnungsebene 1.2 und höher unterstützt. Restriction: VARYINGZ wird nur von Enterprise PL/I un- terstützt.	<code><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></code>

PL/I-Datenbeschreibung	Schema
<p>GRAPHIC(n)</p> <p>VARYING und VARYINGZ werden auch auf Zuordnungsebene 1.2 und höher unterstützt.</p> <p>Restriction: VARYINGZ wird nur von Enterprise PL/I unterstützt.</p>	<p>Auf Zuordnungsebene 1.0 und 1.1, wobei $m = 2 * n$:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" m" /> </xsd:restriction> </xsd:simpleType></pre> <p>Auf Zuordnungsebene 1.2 oder höher:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value=" n" /> <xsd:whiteSpace value="preserve" /> </xsd:restriction> </xsd:simpleType></pre>
<p>WIDECHAR(n)</p> <p>Restriction: Nur Enterprise PL/I</p>	<p>Auf Zuordnungsebene 1.0 und 1.1, wobei $m = 2 * n$:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" m" /> </xsd:restriction> </xsd:simpleType></pre> <p>Auf Zuordnungsebene 1.2 oder höher:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" n" /> </xsd:restriction> </xsd:simpleType></pre> <p>Auf Zuordnungsebene 4.0 und höher füllt CICS das Anwendungsdatenstrukturfeld mit UTF-16-Daten.</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>Restriction: Nur Enterprise PL/I</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:byte" /> </xsd:simpleType></pre>
<p>BINARY FLOAT(n)</p> <p>Dabei ist $n \leq 21$.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
<p>BINARY FLOAT(n)</p> <p>Dabei ist $21 < n \leq 53$.</p> <p>Werte größer als 53 werden nicht unterstützt.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

PL/I-Datenbeschreibung	Schema
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>Dabei ist $n \leq 6$.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:float-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
<p>DECIMAL FLOAT(<i>n</i>) , wobei $6 < n \leq 16$ ist.</p> <p>Werte größer als 16 werden nicht unterstützt.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:double-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

Zuordnung von XML-Schema zu PL/I:

Die Dienstprogramme DFHSC2LS und DFHWS2LS unterstützen Zuordnungen zwischen XML-Schemadefinitionen und PL/I-Datenstrukturen. Da sich der Enterprise PL/I-Compiler von älteren PL/I-Compilern unterscheidet, werden zwei Sprachoptionen unterstützt: PLI-ENTERPRISE und PLI-OTHER.

Regeln für die Zuordnung von Schemaelementnamen zu PL/I

Die CICS-Assistenten generieren eindeutige und gültige Namen für PL/I-Variablen aus den Schemaelementnamen mithilfe der folgenden Regeln:

1. Alle Zeichen außer A-Z, a-z, 0-9, @, #, _ oder \$ werden ersetzt durch X .

Beispiel: monthly-total wird zu monthlyXtotal.

Sie können den Parameter **MAPPING-OVERRIDES** verwenden, um anzupassen, wie andere Zeichen behandelt werden. Wenn Sie beispielsweise den Wert **HYPHENS-AS-UNDERSCORES** festlegen, werden alle Bindestriche in der XML in einen Unterstrich statt in ein X konvertiert. Beispiel: monthly-total wird zu monthly_total.

2. Wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat (d. h. die Attribute minOccurs und maxOccurs werden mit unterschiedlichen

Werten in `xsd:element` angegeben), und wenn der Name des Schemaelements länger als 24 Zeichen ist, wird er auf diese Länge abgeschnitten.

Wenn das Schema angibt, dass die Variable eine feste Kardinalität hat und wenn der Name des Schemaelements länger als 29 Zeichen ist, wird er auf diese Länge abgeschnitten.

3. Doppelte Namen in demselben Bereich werden durch das Hinzufügen einer oder mehrerer Ziffern zu der zweiten Instanz und zu allen nachfolgenden Instanzen des Namens eindeutig gemacht.

Beispiel: Drei Instanzen von `year` werden zu `year`, `year1` und `year2`.

Sollte das oben beschriebene Verhalten unerwünscht sein, kann der Benutzer `MAPPING-OVERRIDES=NO-ARRAY-NAME-INDEXING` als Eingabe für das Dienstprogramm angeben, die das Hinzufügen einer oder mehrerer Ziffern zur zweiten Instanz und allen nachfolgenden Instanzen des Namens inaktiviert.

4. Fünf Zeichen sind reserviert für die Zeichenfolgen `_cont` oder `_num`, die verwendet werden, wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat, d. h. wenn die Attribute `minOccurs` und `maxOccurs` mit unterschiedlichen Werten angegeben werden.

Weitere Informationen finden Sie unter „Variable Arrays von Elementen“ auf Seite 372.

5. Für Attribute werden die vorherigen Regeln auf den Elementnamen angewendet. Das Präfix `attr-` wird dem Elementnamen hinzugefügt, gefolgt von `-value` oder `-exist`. Wenn die Gesamtlänge mehr als 28 Zeichen beträgt, wird der Elementname abgeschnitten. Weitere Informationen finden Sie unter „Unterstützung für XML-Attribute“ auf Seite 377.

Für das auf `Nil` festlegbare Attribut gelten spezielle Regeln. Das Präfix `attr-` wird hinzugefügt, aber `nil-` wird ebenfalls am Anfang des Elementnamens hinzugefügt. Auf den Elementnamen folgt `-value`. Wenn die Gesamtlänge mehr als 28 Zeichen beträgt, wird der Elementname abgeschnitten.

Der resultierende Name ist höchstens 31 Zeichen lang.

Regeln für die Zuordnung von Schematypen zu PL/I

DFHSC2LS und DFHWS2LS ordnen Schematypen zu PL/I-Datentypen entsprechend der folgenden Tabelle zu. Beachten Sie außerdem die folgenden Punkte:

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf `NULL` gesetzt ist, werden Zeichendaten mit variabler Länge auf null endenden Zeichenfolgen zugeordnet, und ein zusätzliches Zeichen wird dem Nullabschlusszeichen zugeordnet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher festgelegt und der Parameter **CHAR-VARYING** nicht angegeben ist, werden standardmäßig Zeichendaten mit variabler Länge zu einem `VARYINGZ`-Datentyp für Enterprise PL/I und einem `VARYING`-Datentyp für andere PL/I-Versionen zugeordnet.
- Binärdaten mit variabler Länge werden einem `VARYING`-Datentyp zugeordnet, wenn sie kleiner als 32.768 Bytes sind, und einem Container, wenn sie größer als 32.768 Bytes sind.

Schema	PL/I-Datenbeschreibung
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	<p>Zuordnungsebene 2.0 und niedriger:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 2.1 und höher:</p> <p>Unterstützt</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpleType"> </xsd:restriction> </xsd:simpleType></pre>	<p>Zuordnungsebene 1.1 und höher: CHAR(255)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: typ"> <xsd:maxLength value=" z"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> <p>Dabei nimmt <i>typ</i> einen der folgenden Werte an:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Alle Zuordnungsebenen: CHARACTER(z)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: typ"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre> <p>Dabei nimmt <i>typ</i> einen der folgenden Werte an:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>WIDECHAR(z)</p>

Schema	PL/I-Datenbeschreibung
<pre><xsd:simpleType> <xsd:restriction base="xsd: typ"> </xsd:restriction> </xsd:simpleType></pre> <p>Dabei nimmt <i>typ</i> einen der folgenden Werte an:</p> <ul style="list-style-type: none"> duration date time gDay gMonth gYear gMonthDay gYearMonth 	<p>Alle Zuordnungsebenen: CHAR(32)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	<p>Zuordnungsebene 1.2 und niedriger:</p> <p style="text-align: center;">CHAR(32)</p> <p>Zuordnungsebene 2.0 und höher:</p> <p style="text-align: center;">CHAR(40)</p> <p>Zuordnungsebene 3.0 und höher:</p> <p style="text-align: center;">FIXED DECIMAL(15)</p> <p>Das Format der Zeitmarke ist CICS ABSTIME.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" y"/> </xsd:restriction> </xsd:simpleType></pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p style="text-align: center;">BIT(z)</p> <p>Dabei ist $z = 8 \times y$ und $z < 4095$ Bytes.</p> <p style="text-align: center;">CHAR(z)</p> <p>Dabei ist $z = 8 \times y$ und $z > 4095$ Bytes.</p> <p>Zuordnungsebenen 1.2 und höher:</p> <p style="text-align: center;">CHAR(y)</p>

Schema	PL/I-Datenbeschreibung
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Andere PL/I-Versionen FIXED BINARY (7)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY (8)</p> <p>Andere PL/I-Versionen FIXED BINARY (8)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (15)</p> <p>Andere PL/I-Versionen FIXED BINARY (15)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY (16)</p> <p>Andere PL/I-Versionen FIXED BINARY (16)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>Enterprise PL/I FIXED DECIMAL(31,0)</p> <p>Andere PL/I-Versionen FIXED DECIMAL(15,0)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (31)</p> <p>Andere PL/I-Versionen FIXED BINARY (31)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	<p>Alle Zuordnungsebenen:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(32)</p> <p>Andere PL/I-Versionen BIT(64)</p>

Schema	PL/I-Datenbeschreibung
<pre> <xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p>Enterprise PL/I SIGNED FIXED BINARY(63) Anmerkung: Die LIMITS-Compilersteueranweisung kann sich auf die Interpretation dieses Felds durch den PL/I-Compiler auswirken. CICS erwartet eine deklarierte Größe dieses Felds, aber der Compiler optimiert das Feld unter Umständen, indem er es verkleinert, was zu einer Abweichung führt. Vermeiden Sie solche Probleme, indem Sie die Kompilierzeioption LIMITS(FIXEDBIN(63)) verwenden.</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I CHAR(y)</p> <p>Dabei ist y eine feste Länge kleiner 16 MB.</p> <p>Alle Zuordnungsebenen:</p> <p>Andere PL/I-Versionen BIT(64)</p>
<pre> <xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(64) Anmerkung: Die LIMITS-Compilersteueranweisung kann sich auf die Interpretation dieses Felds durch den PL/I-Compiler auswirken. CICS erwartet eine deklarierte Größe dieses Felds, aber der Compiler optimiert das Feld unter Umständen, indem er es verkleinert, was zu einer Abweichung führt. Vermeiden Sie solche Probleme, indem Sie die Kompilierzeioption LIMITS(FIXEDBIN(63)) verwenden.</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I CHAR(y)</p> <p>Dabei ist y eine feste Länge kleiner 16 MB.</p> <p>Alle Zuordnungsebenen:</p> <p>Andere PL/I-Versionen BIT(64)</p>

Schema	PL/I-Datenbeschreibung
<pre> <xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Andere PL/I-Versionen FIXED BINARY (7)</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I BIT(7) BIT(1)</p> <p>Andere PL/I-Versionen BIT(7) BIT(1)</p> <p>Dabei wird BIT(7) für die Ausrichtung angegeben und BIT(1) enthält den zugeordneten booleschen Wert.</p>
<pre> <xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" n"/> <xsd:fractionDigits value=" m"/> </xsd:restriction> </xsd:simpleType> </pre>	<p>Alle Zuordnungsebenen: 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>Alle Zuordnungsebenen: CHAR(255)</p>
<pre> <xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType> </pre>	<p>Alle Zuordnungsebenen: CHAR(255)</p>

Schema	PL/I-Datenbeschreibung
<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>Dabei ist die Länge nicht definiert.</p>	<p>Zuordnungsebene 1.0:</p> <p>Nicht unterstützt</p> <p>Zuordnungsebene 1.1:</p> <p style="text-align: right;">CHAR(z)</p> <p>Dabei ist $z = 4 \times (\text{ceil}(y/3))$. $\text{ceil}(x)$ ist die kleinste ganze Zahl größer-gleich x.</p> <p>Zuordnungsebene 1.2 und höher:</p> <p style="text-align: right;">CHAR(y)</p> <p>Dabei ist die Länge festgelegt.</p> <p style="text-align: right;">CHAR(16)</p> <p>Dabei ist die Länge nicht definiert. Das Feld enthält den 16-Byte-Namen des Containers, der die binären Daten speichert.</p>
<pre> <xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Zuordnungsebenen 1.0 und 1.1:</p> <p style="text-align: right;">CHAR(32)</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I DECIMAL FLOAT(6) HEXADEC</p> <p>Andere PL/I-Versionen DECIMAL FLOAT(6)</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:float-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

Schema	PL/I-Datenbeschreibung
<pre> <xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Zuordnungsebenen 1.0 und 1.1:</p> <p>CHAR(32)</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I DECIMAL FLOAT(16) HEXADEC</p> <p>Andere PL/I-Versionen DECIMAL FLOAT(16)</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für XML verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für xsd:double-Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

Variable Arrays von Elementen:

XML kann ein Array mit einer variierenden Anzahl von Elementen enthalten. Allgemein lassen sich WSDL-Dokumente und XML-Schemas, die eine variierende Anzahl von Elementen enthalten, nicht effizient einer einzelnen Datenstruktur einer höheren Programmiersprache zuordnen. CICS verwendet containerbasierte Zuordnungen oder Inlinezuordnungen, um eine variierende Anzahl von Elementen in XML zu verarbeiten.

Ein Array mit einer variierenden Anzahl von Elementen wird im XML-Schema mithilfe der Attribute minOccurs und maxOccurs in der Elementdeklaration dargestellt:

- Das Attribut minOccurs gibt an, wie oft das Element mindestens vorkommen kann. Es hat den Wert 0 oder eine beliebige positive ganze Zahl.
- Das Attribut maxOccurs gibt an, wie oft das Element höchstens vorkommen kann. Es kann als Wert jede positive ganze Zahl größer-gleich dem Wert des Attributs minOccurs haben. Es kann auch den Wert unbounded haben, der angibt, dass für die Anzahl der Vorkommen des Elements keine Obergrenze gilt.
- Der Standardwert für beide Attribute ist 1.

In diesem Beispiel wird eine 8-Byte-Zeichenfolge dargestellt, die optional ist. Das heißt, sie kann gar nicht oder einmal in der Anwendungs-XML oder der SOAP-Nachricht vorkommen:

```

<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>

```

Im folgenden Beispiel wird eine 8-Byte-Zeichenfolge dargestellt, die mindestens einmal auftreten muss:

```
<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>
```

Allgemein lassen sich WSDL-Dokumente, die eine variierende Anzahl von Elementen enthalten, nicht effizient einer einzelnen Datenstruktur einer höheren Programmiersprache zuordnen. Für diese Fälle verwendet CICS folglich eine Reihe von verbundenen Datenstrukturen, die in einer Reihe von Containern an das Anwendungsprogramm übergeben werden. Diese Strukturen werden als Eingabe und Ausgabe der Anwendung verwendet:

- Wenn CICS XML in Anwendungsdaten umsetzt, füllt es diese Strukturen mit den Anwendungsdaten, und die Anwendung liest sie.
- Wenn CICS die Anwendungsdaten in XML umsetzt, liest es die Anwendungsdaten in den Strukturen, die von der Anwendung gefüllt wurden.

Das Format dieser Datenstrukturen lässt sich am besten anhand einer Reihe von Beispielen erklären. Die XML kann aus einer SOAP-Nachricht oder aus einer Anwendung stammen. Diese Beispiele verwenden ein Array von einfachen 8-Byte-Feldern. Das Modell unterstützt jedoch Arrays komplexer Datentypen ebenso wie Arrays von Datentypen, die andere Arrays enthalten.

Feste Anzahl von Elementen

Das erste Beispiel stellt ein Element dar, das genau dreimal vorkommt.

```
<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>
```

Da in diesem Beispiel die Anzahl der Vorkommen des Elements vorab bekannt ist, kann es als Array fester Länge in einer einfachen COBOL-Deklaration (oder dem Äquivalent in anderen Sprachen) dargestellt werden:

```
05 component PIC X(8) OCCURS 3 TIMES
```

Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner

Dieses Beispiel stellt ein obligatorisches Element dar, das ein- bis fünfmal vorkommen kann:

```
<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>
```

Die Hauptdatenstruktur enthält eine Deklaration mit zwei Feldern. Wenn CICS die XML in binäre Daten umsetzt, enthält das erste Feld, `component-num`, die Anzahl von Vorkommen des Elements in der XML und das zweite Feld, `component-cont`, den Namen eines Containers:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Eine zweite Datenstruktur enthält die Deklaration des Elements selbst:

```
01 DFHWS-component
02 component PIC X(8)
```

Sie müssen den Wert von `component-num` untersuchen (Wert im Bereich zwischen 1 und 5), um herauszufinden, wie oft das Element vorkommt. Der Elementinhalt befindet sich im Container namens `component-cont`. Der Container enthält ein Array von Elementen, wobei jedes Element von der Datenstruktur `DFHWS-component` zugeordnet ist.

Wenn `minOccurs="0"` und `maxOccurs="1"`, ist das Element optional. Um die Datenstruktur in Ihrem Anwendungsprogramm zu verarbeiten, müssen Sie den Wert von `component-num` untersuchen:

- Ist er 0, hat die Nachricht kein Komponentenelement und der Inhalt von `component-cont` ist nicht definiert.
- Ist er 1, befindet sich das Komponentenelement im Container namens `component-cont`.

Der Inhalt des Containers wird von der Datenstruktur `DFHWS-component` zugeordnet.

Anmerkung: Wenn die SOAP-Nachricht aus einem einzelnen wiederkehrenden Element besteht, generiert DFHWS2LS zwei Sprachstrukturen. Die Hauptsprachstruktur enthält die Anzahl von Elementen in dem Array und den Namen eines Containers, der das Array von Elementen enthält. Die zweite Sprachstruktur ordnet eine einzelne Instanz des wiederkehrenden Elements zu.

Variierende Anzahl von Elementen auf Zuordnungsebene 2.1 und höher

Auf Zuordnungsebene 2.1 und höher können Sie den Parameter **INLINE-MAXOCCURS-LIMIT** in den CICS-Assistenten verwenden. Der Parameter **INLINE-MAXOCCURS-LIMIT** gibt an, wie die variierende Anzahl von Elementen verarbeitet wird. Bei den Zuordnungsoptionen für eine variierende Anzahl von Elementen handelt es sich containerbasierte Zuordnung, wie unter „Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner“ auf Seite 373 beschrieben, oder um eine Inline-Zuordnung. Der *wert* dieses Parameters kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein:

- Der Standardwert von **INLINE-MAXOCCURS-LIMIT** ist 1, was sicherstellt, dass optionale Elemente inline zugeordnet werden.
- Der Wert 0 für den Parameter **INLINE-MAXOCCURS-LIMIT** verhindert eine Inline-Zuordnung.
- Wenn `maxOccurs` kleiner-gleich dem Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die Inline-Zuordnung verwendet.
- Wenn `maxOccurs` größer als der Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die containerbasierte Zuordnung verwendet.

Die Inline-Zuordnung einer variierenden Anzahl von Elementen führt dazu, dass sowohl ein Array (wie in dem obigen Beispiel für die feste Anzahl von Vorkom-

men) als auch ein Zähler generiert wird. Das Feld `component-num` gibt an, wie viele Instanzen des Elements vorhanden sind, und auf diese wird vom Array verwiesen. Für das Beispiel in „Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner“ auf Seite 373, bei dem **INLINE-MAXOCCURS-LIMIT** kleiner-gleich 5 ist, sieht die generierte Datenstruktur wie folgt aus:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

Das erste Feld, `component-num`, ist identisch mit der Ausgabe für das Beispiel für die containerbasierte Zuordnung im vorherigen Abschnitt. Das zweite Feld enthält ein Array mit der Länge 5, das groß genug ist, um die maximale Anzahl von Elementen, die generiert werden können, zu enthalten.

Die Inline-Zuordnung unterscheidet sich von der containerbasierten Zuordnung, die die Anzahl von Vorkommen des Elements und den Namen des Containers, in dem die Daten enthalten sind, speichert, insofern als dass sie alle Daten in dem aktuellen Container speichert. Durch das Speichern der Daten im aktuellen Container wird die Leistung im Allgemeinen verbessert, was die Inline-Zuordnung zur präferierten Lösung macht.

Verschachtelte variierende Arrays

Komplexe WSDL-Dokumente und XML-Schemas können variierende wiederkehrende Elemente enthalten, die ihrerseits wiederum variierende wiederkehrende Elemente enthalten können. In diesem Fall erstreckt sich die beschriebene Struktur über die beiden in den Beispielen beschriebenen Ebenen hinaus.

Dieses Beispiel stellt ein optionales Element namens `<component2>` dar, das in einem obligatorischen Element namens `<component1>` verschachtelt ist, wobei das obligatorische Element ein- bis fünfmal vorkommen kann.

```
<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>
```

Die übergeordnete Datenstruktur ist identisch mit früheren Beispielen:

```
05 component1-num PIC S9(9) COMP-5  
05 component1-cont PIC X(16)
```

Die zweite Datenstruktur enthält jedoch die folgenden Elemente:

```
01 DFHWS-component1  
02 component2-num PIC S9(9) COMP-5  
02 component2-cont PIC X(16)
```

Eine Struktur auf dritter Ebene enthält die folgenden Elemente:

```
01 DFHWS-component2  
02 component2 PIC X(8)
```

Die Anzahl von Vorkommen des äußersten Elements `<component1>` ist in `component1-num` angegeben.

Der in `component1-cont` benannte Container enthält ein Array mit dieser Anzahl von Instanzen der zweiten Datenstruktur `DFHWS-component1`.

Jede Instanz von `component2-cont` benennt einen anderen Container, der jeweils die Datenstruktur enthält, die von der Struktur auf dritter Ebene, `DFHWS-component2`, zugeordnet wird.

Um diese Struktur zu veranschaulichen, ziehen Sie das XML-Fragment in Betracht, das dem folgenden Beispiel entspricht:

```
<component1><component2>
string1
</component2></component1>
<component1><component2>
string2
</component2></component1>
<component1></component1>
```

`<component1>` kommt dreimal vor. Die ersten beiden Vorkommen enthalten jeweils eine Instanz von `<component2>`, die dritte Instanz jedoch nicht.

In der übergeordneten Datenstruktur enthält `component1-num` den Wert 3. Der in `component1-cont` benannte Container enthält drei Instanzen von `DFHWS-component1` :

1. In der ersten Instanz hat `component2-num` den Wert 1 und der in `component2-cont` benannte Container enthält *string1*.
2. In der zweiten Instanz hat `component2-num` den Wert 1 und der in `component2-cont` benannte Container enthält *string2*.
3. In der dritten Instanz hat `component2-num` den Wert 0 und der Inhalt von `component2-cont` ist nicht definiert.

In dieser Instanz wird die gesamte Datenstruktur von vier Containern dargestellt:

- Stammdatenstruktur in Container `DFHWS-DATA`
- In `component1-cont` benannter Container
- In den ersten beiden Instanzen von `component2-cont` benannte zwei Container

Optionale Strukturen und 'xsd:choice'

`DFHWS2LS` und `DFHSC2LS` unterstützen die Verwendung von `maxOccurs` und `minOccurs` in den Elementen `<xsd:sequence>`, `<xsd:choice>` und `<xsd:all>` nur auf Zuordnungsebene 2.1 und höher, wobei die Attribute `minOccurs` und `maxOccurs` auf `minOccurs="0"` und `maxOccurs="1"` gesetzt sind.

Die Assistenten generieren Zuordnungen, die diese Elemente behandeln, als seien die in ihnen enthaltenen untergeordneten Elemente optional. Wenn Sie eine Anwendung mit diesen Elementen implementieren, stellen Sie sicher, dass keine ungültigen Kombinationen von Optionen von der Anwendung generiert werden. Jedes dieser Elemente hat ein eigenes Feld `count` in der generierten Sprachstruktur; diese Felder müssen entweder alle auf "0" oder alle auf "1" gesetzt sein. Jede andere Kombination von Werten ist ungültig, außer mit `<xsd:choice>`-Elementen.

`<xsd:choice>`-Elemente geben an, dass nur eine der Optionen in dem Element verwendet werden kann. Dies wird auf allen Zuordnungsebenen unterstützt. Die Assistenten verarbeiten alle diese Optionen in einem `<xsd:choice>`-Element als ob es sich um ein `<xsd:sequence>`-Element mit `minOccurs="0"` und `maxOccurs="1"` han-

delt. Stellen Sie beim Implementieren einer Anwendung mithilfe des `<xsd:choice>`-Elements sicher, dass keine ungültigen Kombinationen von Optionen von der Anwendung generiert werden. Jedes der Elemente hat ein eigenes Feld `count` in der generierten Sprachstruktur. Von diesen muss genau eines den Wert 1 haben, die anderen müssen alle den Wert 0 haben. Alle anderen Kombinationen von Werten sind ungültig, außer wenn das `<xsd:choice>`-Element selbst optional ist. In diesem Fall können alle Felder den Wert 0 haben.

Unterstützung für XML-Attribute:

XML-Schemas können Attribute angeben, die in XML zulässig oder erforderlich sind. Die Dienstprogramme DFHWS2LS und DFHSC2LS des CICS-Assistenten ignorieren XML-Attribute standardmäßig. Für die Verarbeitung von XML-Attributen, die im XML-Schema definiert sind, muss der Wert des Parameters **MAPPING-LEVEL** 1.1 oder höher sein.

Optionale Attribute

Attribute können optional oder erforderlich sein. Sie können jedem Element in einer SOAP-Nachricht oder in einer XML für eine Anwendung zugeordnet werden. Für jedes optionale Attribut, das im Schema definiert ist, werden zwei Felder in der entsprechenden Sprachstruktur generiert:

1. Ein Vorhandensein-Flag; dieses Feld wird als boolescher Datentyp behandelt und hat typischerweise eine Länge von 1 Byte.
2. Ein Wert; dieses Feld wird auf dieselbe Weise zugeordnet wie ein XML-Element eines äquivalenten Typs. Beispielsweise wird ein Attribut vom Typ `NMTOKEN` auf dieselbe Weise zugeordnet wie ein XML-Element vom Typ `NMTOKEN`.

Das Vorhandensein des Attributs und die Wertfelder werden in der generierten Sprachstruktur vor dem Feld für das Element angezeigt, dem Sie zugeordnet sind. Unerwartete Attribute, die im Instanzdokument angezeigt werden, werden ignoriert.

Sehen Sie sich beispielsweise die folgende Schemaattributdefinition an:

```
<xsd:attribute name="age" type="xsd:short"
use="optional" />
```

Dieses optionale Attribut ist der folgenden COBOL-Struktur zugeordnet:

```
05 attr-age-exist PIC X DISPLAY
05 attr-age-value PIC S9999 COMP-5 SYNC
```

Laufzeitverarbeitung optionaler Attribute

Die folgende Laufzeitverarbeitung erfolgt für optionale Attribute:

- Wenn das Attribut vorhanden ist, wird das Vorhandensein-Flag gesetzt und der Wert wird zugeordnet.
- Wenn das Attribut nicht vorhanden ist, wird das Vorhandensein-Flag nicht gesetzt.
- Wenn das Attribut einen Standardwert hat und vorhanden ist, wird der Wert zugeordnet.
- Wenn das Attribut einen Standardwert hat und nicht vorhanden ist, wird der Standardwert zugeordnet.

Optionale Attribute mit Standardwerten werden als erforderliche Attribute behandelt.

Wenn CICS die Daten in XML umsetzt, erfolgt die folgende Laufzeitverarbeitung:

- Wenn das Vorhandensein-Flag gesetzt ist, wird das Attribut umgesetzt und in die XML eingeschlossen.
- Wenn das Vorhandensein-Flag nicht gesetzt ist, wird das Attribut nicht in die XML eingeschlossen.

Erforderliche Attribute und Laufzeitverarbeitung

Für jedes erforderliche Attribut wird nur das Wertfeld in der entsprechenden Sprachstruktur generiert.

Wenn das Attribut in der XML-Datei vorhanden ist, wird der Wert zugeordnet. Wenn das Attribut nicht vorhanden ist, erfolgt die folgende Verarbeitung:

- Wenn es sich bei der Anwendung um einen Web-Service-Provider handelt, generiert CICS eine SOAP-Fehlernachricht, die auf einen Fehler in der Client-SOAP-Nachricht hinweist.
- Wenn es sich bei der Anwendung um einen Web-Service-Requester handelt, gibt CICS eine Nachricht aus und eine Konvertierungsfehlerantwort mit einem RESP2-Code von 13 an die Anwendung zurück.
- Wenn die Anwendung den Befehl **TRANSFORM XMLTODATA** verwendet, gibt CICS eine Nachricht aus und eine Antwort bezüglich einer ungültigen Anforderung mit einem RESP2-Code von 3 an die Anwendung zurück.

Wenn CICS eine SOAP-Nachricht basierend auf dem Inhalt eines Kommunikationsbereichs oder Containers erstellt, wird das Attribut umgesetzt und in die Nachricht eingeschlossen. Wenn eine Anwendung den Befehl **TRANSFORM DATATOXML** verwendet, wandelt CICS das Attribut ebenfalls um und schließt es in die XML ein.

Auf Nil festlegbares Attribut

Ein auf Nil festlegbares Attribut ist ein spezielles Attribut, das für ein `xsd:element` in einem XML-Schema angezeigt werden kann. Es gibt an, dass das Attribut `xsi:nil` für das Element in XML gültig ist. Wenn für ein Element das Attribut `xsi:nil` angegeben wurde, bedeutet dies, dass das Element vorhanden ist, aber keinen Wert hat. Daher ist ihm kein Inhalt zugeordnet.

Wenn in einem XML-Schema das auf Nil festlegbare Attribut als 'true' definiert ist, wird es als erforderliches Attribut zugeordnet, das einen booleschen Wert annimmt.

Wenn CICS eine SOAP-Nachricht empfängt oder XML für eine Anwendung umsetzen muss, die ein Attribut `xsi:nil` enthält, kann der Wert des Attributs 'true' oder 'false' sein. Lautet der Wert 'true', muss die Anwendung die Werte des Elements oder der verschachtelten Elemente im Geltungsbereich des Attributs `xsi:nil` ignorieren.

Wenn CICS eine SOAP-Nachricht oder XML basierend auf dem Inhalt eines Kommunikationsbereichs oder Containers erstellt, für den der Wert für das Attribut `xsi:nil` 'true' lautet, erfolgt die folgende Verarbeitung:

- Das Attribut `xsi:nil` wird in der XML- oder der SOAP-Nachricht generiert.
- Der Wert des zugeordneten Elements wird ignoriert.
- Alle in dem Element verschachtelten Elemente werden ignoriert.

Beispiel für eine SOAP-Nachricht

Sehen Sie sich das folgende XML-Beispielschema an, das Teil eines WSDL-Dokuments sein könnte:

```
<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>
```

Im Folgenden sehen Sie ein Beispiel für eine partielle SOAP-Nachricht, die diesem Schema entspricht:

```
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <num xsi:nil="true"/>
  <num>15</num>
  <num xsi:nil="true"/>
</root>
```

In COBOL wird diese SOAP-Nachricht den folgenden Elementen zugeordnet:

```
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)
```

Unterstützung für <xsd:any> und xsd:anyType:

DFHWS2LS und DFHSC2LS unterstützen die Verwendung von <xsd:any> und xsd:anyType im XML-Schema. Sie können das XML-Schemaelement <xsd:any> verwenden, um einen Abschnitt eines XML-Dokuments mit nicht definiertem Inhalt zu beschreiben. xsd:anyType ist der Basisdatentyp, aus dem alle einfachen und komplexen Datentypen abgeleitet werden. Er kennt keine Einschränkungen für die Dateninhalte.

Bevor Sie <xsd:any> und xsd:anyType mit den CICS-Assistenten verwenden können, legen Sie die folgenden Parameter fest:

- Legen Sie den Parameter **MAPPING-LEVEL** auf 2.1 oder höher fest.
- Für eine Web-Service-Provider-Anwendung geben Sie für den Parameter **PGMINT CHANNEL** an.

<xsd:any> - Beispiel

In diesem Beispiel wird ein <xsd:any>-Element verwendet, um optionalen unstrukturierten XML-Inhalt nach dem Tag "Surname" im Tag "Customer" zu beschreiben:

```
<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>
```

Im Folgenden sehen Sie ein Beispiel für eine SOAP-Nachricht, die diesem XML-Schema entspricht:

```
<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">
          <!-- Dieser Tag 'ExtraInformation' ist dem optionalen 'xsd:any'-Element
            in dem XML-Schema zugeordnet.
            Er kann jede korrekt formatierte XML enthalten. -->
          <ExampleField1>one</ExampleField1>
          <ExampleField2>two</ExampleField2>
        </ExtraInformation>
      </Customer>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Wenn diese SOAP-Nachricht an CICS gesendet wird, füllt CICS den Container Customer-xml-cont mit den folgenden XML-Daten:

```
<ExtraInformation
  xmlns="http://www.example.org/ExtraInformation">
  <!-- Dieser Tag 'ExtraInformation' ist dem optionalen 'xsd:any'-Element
    in dem XML-Schema zugeordnet.
    Er kann jede korrekt formatierte XML enthalten. -->
  <ExampleField1>one</ExampleField1>
  <ExampleField2>two</ExampleField2>
</ExtraInformation>
```

CICS füllt auch den Container Customer-xmlns-cont mit den folgenden gültigen XML-Namensbereichsdeklarationen. Diese Deklarationen sind durch ein Leerzeichen getrennt:

```
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://www.example.org/anyExample"
```

xsd:anyType - Beispiel

xsd:anyType ist der Basisdatentyp, aus dem alle einfachen und komplexen Datentypen abgeleitet werden. Er kennt keine Einschränkungen für Dateninhalte. Wenn Sie keinen Datentyp angeben, wird standardmäßig der Wert xsd:anyType angenommen. Die beiden folgenden XML-Fragmente sind beispielsweise äquivalent:

```
<xsd:element name="Name" type="xsd:anyType"/>
<xsd:element name="Name"/>
```

Generierte Sprachstrukturen

Die für <xsd:any> oder xsd:anyType generierten Sprachstrukturen haben das folgende Format in COBOL und ein äquivalentes Format für die anderen Sprachen:

elementName-xml-cont PIC X(16)

Der Name eines Containers, der die unformatierte XML enthält. Wenn CICS eine eingehende SOAP-Nachricht verarbeitet, stellt es das Subset der SOAP-Nachricht, das von <xsd:any> oder xsd:anyType definiert wird, in diesen Container. Die Anwendung kann die XML-Daten nur nativ verarbeiten. Die Anwendung muss die XML generieren, diesen Container mit Daten füllen und den Containernamen angeben.

Dieser Container muss im Textmodus gefüllt werden. Wenn CICS diesen Container mit Daten füllt, geschieht dies unter Verwendung derselben Variante von EBCDIC wie für den Web-Service definiert wurde. Zeichen, die in der EBCDIC-Zielcodepage nicht vorhanden sind, werden durch Substitutionszeichen ersetzt, auch wenn der Container von der Anwendung in UTF-8 gelesen wird.

elementName-xmlns-cont PIC X(16)

Der Name eines Containers, der alle gültigen Namensbereichspräfixdeklarationen enthält. Der Inhalt dieses Containers ist dem Inhalt des Containers DFHWS-XMLNS ähnlich, mit der Ausnahme, dass er alle gültigen und relevanten Namensbereichsdeklarationen enthält, und nicht nur das Subset des SOAP-Envelope-Tags.

Dieser Container muss im Textmodus gefüllt werden. Wenn CICS diesen Container mit Daten füllt, geschieht dies unter Verwendung derselben Variante von EBCDIC wie für den Web-Service definiert wurde. Zeichen, die in der EBCDIC-Zielcodepage nicht vorhanden sind, werden durch Substitutionszeichen ersetzt, auch wenn der Container von der Anwendung in UTF-8 gelesen wird.

Dieser Container wird nur verwendet, wenn SOAP-Nachrichten verarbeitet werden, die an CICS gesendet wurden. Falls die Anwendung versucht, einen Container mit Namensbereichsdeklarationen bereitzustellen, wenn eine SOAP-Ausgabenachricht generiert wird, werden der Container und sein Inhalt von CICS ignoriert. CICS erfordert, dass die von der Anwendung bereitgestellte XML im Hinblick auf Namensbereichsdeklarationen vollkommen eigenständig ist.

Der Name des XML-Elements, das das `<xsd:any>`-Element enthält, ist in den Variablennamen enthalten, die für das `<xsd:any>`-Element generiert werden. In dem Beispiel für `<xsd:any>`, ist das `<xsd:any>`-Element in dem Element `<xsd:element name="Customer">` verschachtelt und die Variablennamen, die für das `<xsd:any>`-Element generiert werden, lauten `Customer-xml-cont PIC X(16)` und `Customer-xmlns-cont PIC X(16)`.

Für einen `xsd:anyType`-Typ wird der direkte XML-Elementname verwendet. In dem obigen Beispiel für `xsd:anyType` lauten die Variablennamen `Name-xml-cont PIC X(16)` und `Name-xmlns-cont PIC X(16)`.

Unterstützung für `<xsd:choice>`:

Ein Element `<xsd:choice>` gibt an, dass nur eine der Optionen im Element verwendet werden kann. Die CICS-Assistenten stellen in unterschiedlichem Umfang Unterstützung für `<xsd:choice>`-Elemente auf den verschiedenen Zuordnungsebenen bereit.

Unterstützung für `<xsd:choice>` auf Zuordnungsebene 2.2 und höher

Auf Zuordnungsebene 2.2 und höher stellen DFHWS2LS und DFHSC2LS eine verbesserte Unterstützung für `<xsd:choice>`-Elemente bereit. Die Assistenten generieren einen neuen Container, der den Wert speichert, der dem `<xsd:choice>`-Element zugeordnet ist. Die Assistenten generieren Sprachstrukturen, die den Namen eines neuen Containers und ein zusätzliches Feld enthalten:

fieldname -enum

Das differenzierende Feld zur Angabe, welche der Optionen von dem `<xsd:choice>`-Element verwendet werden.

feldname -cont

Der Name des Containers, in dem die zu verwendende Option gespeichert wird. Es wird eine weitere Sprachstruktur generiert, um den Optionswert zuzuordnen.

Das folgende XML-Schemafragment enthält ein `<xsd:choice>`-Element:

```
<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>
```

Wenn dieses XML-Schemafragment auf Zuordnungsebene 2.2 oder höher verarbeitet wird, generiert der Assistent die folgenden COBOL-Sprachstrukturen:

```
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.
```

Einschränkungen für `<xsd:choice>` auf Zuordnungsebene 2.2 und höher

DFHSC2LS und DFHWS2LS unterstützen keine verschachtelten `<xsd:choice>`-Elemente. Beispielsweise wird die folgende XML nicht unterstützt:

```
<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 und DFHWS2LS unterstützen keine wiederkehrenden `<xsd:choice>`-Elemente. Beispielsweise wird die folgende XML nicht unterstützt:

```
<xsd:choice maxOccurs="2">
  <xsd:element name="name1" type="string"/>
</xsd:choice>
```

DFHSC2LS und DFHWS2LS unterstützen höchstens 255 Optionen in einem `<xsd:choice>`-Element.

Unterstützung für <xsd:choice> auf Zuordnungsebene 2.1 und niedriger

Auf Zuordnungsebene 2.1 und niedriger stellt DFHWS2LS eingeschränkte Unterstützung für <xsd:choice>-Elemente bereit. DFHWS2LS behandelt jede einzelne Option in einem <xsd:choice>-Element, als ob es sich um ein <xsd:sequence>-Element handelte, das höchstens einmal vorkommen kann.

Nur eine der Optionen in einem <xsd:choice>-Element kann verwendet werden, deshalb müssen Sie beim Implementieren einer Anwendung mit dem <xsd:choice>-Element darauf achten, dass Sie nur gültige Kombinationen von Optionen generieren. Jedes der Elemente hat ein eigenes Feld count in der generierten Sprachstruktur. Von diesen muss genau eines den Wert 1 haben, die anderen müssen alle den Wert 0 haben. Alle anderen Kombinationen von Werten sind falsch, außer wenn <xsd:choice> selbst optional ist. In diesem Fall können alle Felder den Wert 0 haben.

Unterstützung für Substitutionsgruppen:

Mit einer Substitutionsgruppe können Sie eine Gruppe von austauschbaren XML-Elementen definieren. Die CICS-Assistenten stellen Unterstützung für Substitutionsgruppen auf Zuordnungsebene 2.2 und höher bereit.

Auf Zuordnungsebene 2.2 und höher unterstützen DFHSC2LS und DFHWS2LS Substitutionsgruppen mithilfe ähnlicher Zuordnungen wie denjenigen für <xsd:choice>-Elemente. Der Assistent generiert ein Aufzählungsfeld und einen neuen Containernamen in der Sprachstruktur.

Das folgende XML-Schemafragment umfasst ein Array von zwei subGroupParent-Elementen, von denen jedes durch replacementOption1 oder replacementOption2 ersetzt werden kann.

```
<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" />
```

Die Verarbeitung dieses XML-Fragments mit dem Assistenten generiert die folgenden COBOL-Sprachstrukturen:

```
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).
```

Weitere Informationen zu Substitutionsgruppen finden Sie unter XML Schema Part 1: Structures Second Edition.

Unterstützung für abstrakte Elemente und abstrakte Datentypen:

Die CICS-Assistenten stellen Unterstützung für abstrakte Elemente und abstrakte Datentypen auf Zuordnungsebene 2.2 und höher bereit. Die CICS-Assistenten ordnen abstrakte Elemente und abstrakte Datentypen in ähnlicher Weise zu wie Substitutionsgruppen.

Unterstützung für abstrakte Elemente auf Zuordnungsebene 2.2 und höher

Auf Zuordnungsebene 2.2 und höher behandeln DFHSC2LS und DFHWS2LS abstrakte Elemente fast genauso wie Substitutionsgruppen, mit der Ausnahme, dass das abstrakte Element kein gültiges Mitglied der Gruppe ist. Wenn keine Ersetzungselemente vorhanden sind, wird das abstrakte Element als `<xsd:any>`-Element behandelt und verwendet dieselben Zuordnungen wie ein `<xsd:any>`-Element auf Zuordnungsebene 2.1.

Das folgende XML-Schemafragment gibt zwei Optionen an, die anstelle des abstrakten Elements verwendet werden können. Das abstrakte Element selbst ist keine gültige Option:

```
<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" />
```

Die Verarbeitung dieses XML-Fragments mit dem Assistenten generiert die folgenden COBOL-Sprachstrukturen:

```
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.
```


Weitere Informationen zu abstrakten Elementen finden Sie unter XML Schema Part 0: Primer Second Edition.

Unterstützung für abstrakte Datentypen auf Zuordnungsebene 2.2 und höher

Auf Zuordnungsebene 2.2 und höher behandeln DFHSC2LS und DFHWS2LS abstrakte Datentypen als Substitutionsgruppen. Der Assistent generiert ein Aufzählungsfeld und einen neuen Containernamen in der Sprachstruktur.

Das folgende XML-Schemafragment gibt zwei Alternativen an, die anstelle des abstrakten Typs verwendet werden können:

```
<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>
```

Die Verarbeitung dieses XML-Fragments mit dem Assistenten generiert die folgenden COBOL-Sprachstrukturen:

```
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).
```

Die Sprachstrukturen werden in separaten Copybooks generiert. Die Sprachstruktur, die für option1 generiert wird, wird in einem Copybook generiert:

```
03 option1 PIC X(5).
```

Die Sprachstruktur für option2 wird in einem anderen Copybook generiert:

```
03 option2 PIC X(10).
```

Weitere Informationen zu abstrakten Datentypen finden Sie unter XML Schema Part 0: Primer Second Edition.

Informationen zur Verarbeitung variabel wiederkehrender Inhalte in COBOL:

In COBOL können Sie variabel wiederkehrende Inhalte nicht mithilfe von Verweissarithmetik verarbeiten, um die einzelnen Dateninstanzen zu adressieren. Andere Programmiersprachen haben diese Einschränkungen nicht. Im folgenden Beispiel wird gezeigt, wie variabel wiederkehrende Inhalte in COBOL für eine Web-Service-Anwendung verarbeitet werden.

Dieses Verfahren gilt auch für die Umsetzung von XML in Anwendungsdaten unter Verwendung der **TRANSFORM-API**-Befehle. Das folgende WSDL-Beispieldokument stellt einen Web-Service mit Anwendungsdaten in Form einer 8-stelligen Zeichenfolge dar, die unregelmäßig wiederkehrt:

```
<?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>
```

Die Verarbeitung dieses WSDL-Dokuments durch DFHWS2LS generiert die folgenden COBOL-Sprachstrukturen:

```
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).
```

Beachten Sie, dass das acht Zeichen lange Feld `component` in einer separaten Struktur namens `DFHWS-component` definiert ist. Die Hauptdatenstruktur heißt `applicationData` und enthält zwei Felder: `component-num` und `component-cont`. Das

Feld component-num gibt an, wie viele Instanzen der component-Daten vorhanden sind, und das Feld component-cont gibt den Namen eines Containers an, der die verkettete Liste von component-Feldern enthält.

Der folgende COBOL-Code stellt einen Weg zur Verarbeitung der Liste von variabel wiederkehrenden Daten dar. Er setzt ein Verknüpfungsabschnittsarray ein, um nachfolgende Instanzen der Daten zu adressieren, wobei jede Instanz mithilfe der Anweisung DISPLAY angezeigt wird:

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. EXVARY.
```

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.
```

```
* Arbeitsspeichervariablen  
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.
```

```
* umfangreiches Verknüpfungsabschnittsarray  
01 BIG-ARRAY PIC X(659999).  
  
* von DFHWS2LS erstellte Anwendungsdatenstrukturen  
* wird üblicherweise mit einer COPY-Anweisung referenziert  
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.
```

```
* Abrufen des Containers DFHWS-DATA  
EXEC CICS GET CONTAINER('DFHWS-DATA')  
SET(APP-DATA-PTR)  
FLENGTH(APP-DATA-LENGTH)  
END-EXEC  
SET ADDRESS OF DFHWS2LS-data TO APP-DATA-PTR
```

```
* Abrufen der wiederkehrenden Komponentendaten  
EXEC CICS GET CONTAINER(component-cont)  
SET(COMPONENT-PTR)  
FLENGTH(COMPONENT-DATA-LENGTH)  
END-EXEC
```

```
* Verweisen der Komponentenstruktur auf die erste Instanz der Daten  
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR
```

```
* Speichern der Länge einer einzelnen Komponente  
MOVE LENGTH OF DFHWS-component TO COMPONENT-LENGTH
```

```
* Verarbeiten aller Instanzen von Komponentendaten nacheinander  
PERFORM WITH TEST AFTER
```

```

UNTIL COMPONENT-COUNT = component-num

* Anzeigen der aktuellen Instanz der Daten
DISPLAY 'component value is: ' component

* Adressieren der nächsten Instanz der Komponentendaten
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

* Beenden der Schleife
END-PERFORM.

* Verweisen der Komponentenstruktur zurück zur ersten Instanz der
* Daten, für alle weiteren Verarbeitungen, die durchgeführt werden sollen
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

* Zurückkehren zu CICS

EXEC CICS
RETURN
END-EXEC

GOBACK.

```

Der obige Code liefert eine generische Lösung für die Behandlung von variabel wiederkehrenden Inhalten. Das Array, BIG-ARRAY, bewegt sich der Reihe nach zum Anfang jeder einzelnen Komponente und verbleibt nicht fix am Anfang der Daten. Die Komponentendatenstruktur wird dann bewegt, um auf das erste Byte der nächsten Komponente zu verweisen. COMPONENT-PTR kann bei Bedarf verwendet werden, um die Anfangsposition der Komponentendaten wiederherzustellen.

Im Folgenden sehen Sie ein Beispiel einer SOAP-Nachricht, die dem WSDL-Dokument entspricht:

```

<?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>

```

Dies ist die Ausgabe, die von dem COBOL-Programm erzeugt wird, wenn es die SOAP-Nachricht verarbeitet:

```

CPIH 20080115103151 component value is: VALUE1
CPIH 20080115103151 component value is: VALUE2
CPIH 20080115103151 component value is: VALUE3

```

Variable Arrays von Elementen

XML kann ein Array mit einer variierenden Anzahl von Elementen enthalten. Allgemein lassen sich WSDL-Dokumente und XML-Schemas, die eine variierende Anzahl von Elementen enthalten, nicht effizient einer einzelnen Datenstruktur einer höheren Programmiersprache zuordnen. CICS verwendet containerbasierte Zuordnungen oder Inlinezuordnungen, um eine variierende Anzahl von Elementen in XML zu verarbeiten.

Ein Array mit einer variierenden Anzahl von Elementen wird im XML-Schema mithilfe der Attribute `minOccurs` und `maxOccurs` in der Elementdeklaration dargestellt:

- Das Attribut `minOccurs` gibt an, wie oft das Element mindestens vorkommen kann. Es hat den Wert 0 oder eine beliebige positive ganze Zahl.
- Das Attribut `maxOccurs` gibt an, wie oft das Element höchstens vorkommen kann. Es kann als Wert jede positive ganze Zahl größer-gleich dem Wert des Attributs `minOccurs` haben. Es kann auch den Wert `unbounded` haben, der angibt, dass für die Anzahl der Vorkommen des Elements keine Obergrenze gilt.
- Der Standardwert für beide Attribute ist 1.

In diesem Beispiel wird eine 8-Byte-Zeichenfolge dargestellt, die optional ist. Das heißt, sie kann gar nicht oder einmal in der Anwendungs-XML oder der SOAP-Nachricht vorkommen:

```
<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>
```

Im folgenden Beispiel wird eine 8-Byte-Zeichenfolge dargestellt, die mindestens einmal auftreten muss:

```
<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>
```

Allgemein lassen sich WSDL-Dokumente, die eine variierende Anzahl von Elementen enthalten, nicht effizient einer einzelnen Datenstruktur einer höheren Programmiersprache zuordnen. Für diese Fälle verwendet CICS folglich eine Reihe von verbundenen Datenstrukturen, die in einer Reihe von Containern an das Anwendungsprogramm übergeben werden. Diese Strukturen werden als Eingabe und Ausgabe der Anwendung verwendet:

- Wenn CICS XML in Anwendungsdaten umsetzt, füllt es diese Strukturen mit den Anwendungsdaten, und die Anwendung liest sie.
- Wenn CICS die Anwendungsdaten in XML umsetzt, liest es die Anwendungsdaten in den Strukturen, die von der Anwendung gefüllt wurden.

Das Format dieser Datenstrukturen lässt sich am besten anhand einer Reihe von Beispielen erklären. Die XML kann aus einer SOAP-Nachricht oder aus einer Anwendung stammen. Diese Beispiele verwenden ein Array von einfachen 8-Byte-Feldern. Das Modell unterstützt jedoch Arrays komplexer Datentypen ebenso wie Arrays von Datentypen, die andere Arrays enthalten.

Feste Anzahl von Elementen

Das erste Beispiel stellt ein Element dar, das genau dreimal vorkommt.

```
<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>

```

Da in diesem Beispiel die Anzahl der Vorkommen des Elements vorab bekannt ist, kann es als Array fester Länge in einer einfachen COBOL-Deklaration (oder dem Äquivalent in anderen Sprachen) dargestellt werden:

```
05 component PIC X(8) OCCURS 3 TIMES
```

Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner

Dieses Beispiel stellt ein obligatorisches Element dar, das ein- bis fünfmal vorkommen kann:

```

<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>

```

Die Hauptdatenstruktur enthält eine Deklaration mit zwei Feldern. Wenn CICS die XML in binäre Daten umsetzt, enthält das erste Feld, component-num, die Anzahl von Vorkommen des Elements in der XML und das zweite Feld, component-cont, den Namen eines Containers:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Eine zweite Datenstruktur enthält die Deklaration des Elements selbst:

```
01 DFHWS-component
02 component PIC X(8)
```

Sie müssen den Wert von component-num untersuchen (Wert im Bereich zwischen 1 und 5), um herauszufinden, wie oft das Element vorkommt. Der Elementinhalt befindet sich im Container namens component-cont. Der Container enthält ein Array von Elementen, wobei jedes Element von der Datenstruktur DFHWS-component zugeordnet ist.

Wenn minOccurs="0" und maxOccurs="1", ist das Element optional. Um die Datenstruktur in Ihrem Anwendungsprogramm zu verarbeiten, müssen Sie den Wert von component-num untersuchen:

- Ist er 0, hat die Nachricht kein Komponentenelement und der Inhalt von component-cont ist nicht definiert.
- Ist er 1, befindet sich das Komponentenelement im Container namens component-cont.

Der Inhalt des Containers wird von der Datenstruktur DFHWS-component zugeordnet.

Anmerkung: Wenn die SOAP-Nachricht aus einem einzelnen wiederkehrenden Element besteht, generiert DFHWS2LS zwei Sprachstrukturen. Die Hauptsprachstruktur enthält die Anzahl von Elementen in dem Array und den Namen eines Containers, der das Array von Elementen enthält. Die zweite Sprachstruktur ordnet eine einzelne Instanz des wiederkehrenden Elements zu.

Variierende Anzahl von Elementen auf Zuordnungsebene 2.1 und höher

Auf Zuordnungsebene 2.1 und höher können Sie den Parameter **INLINE-MAXOCCURS-LIMIT** in den CICS-Assistenten verwenden. Der Parameter **INLINE-MAXOCCURS-LIMIT** gibt an, wie die variierende Anzahl von Elementen verarbeitet wird. Bei den Zuordnungsoptionen für eine variierende Anzahl von Elementen handelt es sich containerbasierte Zuordnung, wie unter „Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner“ auf Seite 373 beschrieben, oder um eine Inline-Zuordnung. Der *wert* dieses Parameters kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein:

- Der Standardwert von **INLINE-MAXOCCURS-LIMIT** ist 1 , was sicherstellt, dass optionale Elemente inline zugeordnet werden.
- Der Wert 0 für den Parameter **INLINE-MAXOCCURS-LIMIT** verhindert eine Inline-Zuordnung.
- Wenn maxOccurs kleiner-gleich dem Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die Inline-Zuordnung verwendet.
- Wenn maxOccurs größer als der Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die containerbasierte Zuordnung verwendet.

Die Inline-Zuordnung einer variierenden Anzahl von Elementen führt dazu, dass sowohl ein Array (wie in dem obigen Beispiel für die feste Anzahl von Vorkommen) als auch ein Zähler generiert wird. Das Feld component-num gibt an, wie viele Instanzen des Elements vorhanden sind, und auf diese wird vom Array verwiesen. Für das Beispiel in „Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner“ auf Seite 373 , bei dem **INLINE-MAXOCCURS-LIMIT** kleiner-gleich 5 ist, sieht die generierte Datenstruktur wie folgt aus:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

Das erste Feld, component-num , ist identisch mit der Ausgabe für das Beispiel für die containerbasierte Zuordnung im vorherigen Abschnitt. Das zweite Feld enthält ein Array mit der Länge 5, das groß genug ist, um die maximale Anzahl von Elementen, die generiert werden können, zu enthalten.

Die Inline-Zuordnung unterscheidet sich von der containerbasierten Zuordnung, die die Anzahl von Vorkommen des Elements und den Namen des Containers, in dem die Daten enthalten sind, speichert, insofern als dass sie alle Daten in dem aktuellen Container speichert. Durch das Speichern der Daten im aktuellen Container wird die Leistung im Allgemeinen verbessert, was die Inline-Zuordnung zur präferierten Lösung macht.

Verschachtelte variierende Arrays

Komplexe WSDL-Dokumente und XML-Schemas können variierende wiederkehrende Elemente enthalten, die ihrerseits wiederum variierende wiederkehrende Elemente enthalten können. In diesem Fall erstreckt sich die beschriebene Struktur über die beiden in den Beispielen beschriebenen Ebenen hinaus.

Dieses Beispiel stellt ein optionales Element namens <component2> dar, das in einem obligatorischen Element namens <component1> verschachtelt ist, wobei das obligatorische Element ein- bis fünfmal vorkommen kann.

```
<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>

```

Die übergeordnete Datenstruktur ist identisch mit früheren Beispielen:

```

05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)

```

Die zweite Datenstruktur enthält jedoch die folgenden Elemente:

```

01 DFHWS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)

```

Eine Struktur auf dritter Ebene enthält die folgenden Elemente:

```

01 DFHWS-component2
02 component2 PIC X(8)

```

Die Anzahl von Vorkommen des äußersten Elements `<component1>` ist in `component1-num` angegeben.

Der in `component1-cont` benannte Container enthält ein Array mit dieser Anzahl von Instanzen der zweiten Datenstruktur `DFHWS-component1`.

Jede Instanz von `component2-cont` benennt einen anderen Container, der jeweils die Datenstruktur enthält, die von der Struktur auf dritter Ebene, `DFHWS-component2`, zugeordnet wird.

Um diese Struktur zu veranschaulichen, ziehen Sie das XML-Fragment in Betracht, das dem folgenden Beispiel entspricht:

```

<component1><component2>
  string1
</component2></component1>
<component1><component2>
  string2
</component2></component1>
<component1></component1>

```

`<component1>` kommt dreimal vor. Die ersten beiden Vorkommen enthalten jeweils eine Instanz von `<component2>`, die dritte Instanz jedoch nicht.

In der übergeordneten Datenstruktur enthält `component1-num` den Wert 3. Der in `component1-cont` benannte Container enthält drei Instanzen von `DFHWS-component1` :

1. In der ersten Instanz hat `component2-num` den Wert 1 und der in `component2-cont` benannte Container enthält *string1*.
2. In der zweiten Instanz hat `component2-num` den Wert 1 und der in `component2-cont` benannte Container enthält *string2*.
3. In der dritten Instanz hat `component2-num` den Wert 0 und der Inhalt von `component2-cont` ist nicht definiert.

In dieser Instanz wird die gesamte Datenstruktur von vier Containern dargestellt:

- Stammdatenstruktur in Container DFHWS-DATA
- In component1-cont benannter Container
- In den ersten beiden Instanzen von component2-cont benannte zwei Container

Optionale Strukturen und 'xsd:choice'

DFHWS2LS und DFHSC2LS unterstützen die Verwendung von maxOccurs und minOccurs in den Elementen <xsd:sequence>, <xsd:choice> und <xsd:all> nur auf Zuordnungsebene 2.1 und höher, wobei die Attribute minOccurs und maxOccurs auf minOccurs="0" und maxOccurs="1" gesetzt sind.

Die Assistenten generieren Zuordnungen, die diese Elemente behandeln, als seien die in ihnen enthaltenen untergeordneten Elemente optional. Wenn Sie eine Anwendung mit diesen Elementen implementieren, stellen Sie sicher, dass keine ungültigen Kombinationen von Optionen von der Anwendung generiert werden. Jedes dieser Elemente hat ein eigenes Feld count in der generierten Sprachstruktur; diese Felder müssen entweder alle auf "0" oder alle auf "1" gesetzt sein. Jede andere Kombination von Werten ist ungültig, außer mit <xsd:choice>-Elementen.

<xsd:choice>-Elemente geben an, dass nur eine der Optionen in dem Element verwendet werden kann. Dies wird auf allen Zuordnungsebenen unterstützt. Die Assistenten verarbeiten alle diese Optionen in einem <xsd:choice>-Element als ob es sich um ein <xsd:sequence>-Element mit minOccurs="0" und maxOccurs="1" handelt. Stellen Sie beim Implementieren einer Anwendung mithilfe des <xsd:choice>-Elements sicher, dass keine ungültigen Kombinationen von Optionen von der Anwendung generiert werden. Jedes der Elemente hat ein eigenes Feld count in der generierten Sprachstruktur. Von diesen muss genau eines den Wert 1 haben, die anderen müssen alle den Wert 0 haben. Alle anderen Kombinationen von Werten sind ungültig, außer wenn das <xsd:choice>-Element selbst optional ist. In diesem Fall können alle Felder den Wert 0 haben.

Unterstützung für Werte variabler Länge und Leerzeichen

Sie können anpassen, wie Werte variabler Länge und Leerzeichen verarbeitet werden, indem Sie Einstellungen in den CICS-Assistenten nutzen und indem Sie Facetten direkt im XML-Schema hinzufügen.

Typischerweise ordnen der CICS-XML-Assistent und der CICS-Web-Service-Assistent Datenzeichenfolgen Zeichenarrays fester Länge zu. Diese Arrays müssen mit Leerzeichen oder Nullen gefüllt werden. Das Zuordnen von Werten variabler Länge zu Datenarrays fester Länge kann ineffizient sein und sinnlos Speicherplatz belegen. Wenn die Länge Ihrer Daten variabel ist, wird empfohlen, die Art und Weise anzupassen, in der diese Zuordnungen gehandhabt werden.

Bei einer WSDL-Dokumentkonvertierung aus einer Sprachstruktur in ein XML-Schema oder WSDL-Dokument wird empfohlen, die Facetten whitespace und maxLength in Ihrem XML-Schema anzupassen und den Parameter **CHAR-VARYING-LIMIT** in den Assistenten anzugeben.

Bei einer Konvertierung aus einem XML-Schema oder WSDL-Dokument in eine Sprachstruktur wird empfohlen, einen entsprechenden Wert für den Parameter **CHAR-VARYING** in den Assistenten festzulegen.

Anmerkung: Nullzeichen ('x00') sind in XML-Dokumenten nicht gültig. Alle Nullzeichen aus Anwendungsdaten, die von CICS geparkt werden, werden als Ende einer Zeichenfolge interpretiert und der Wert wird abgeschnitten. Wenn CICS Anwendungsdaten generiert, tut es dies entsprechend des Werts des Parameters **CHAR-**

VARYING. Beispiel: Wenn die Option **CHAR-VARYING=NULL** angegeben ist, enden Zeichenfolgen variabler Länge, die von CICS generiert werden, auf ein Nullzeichen.

Werte variabler Länge aus XML zu Sprachstrukturen zuordnen

Verwenden Sie Facetten im XML-Schema oder geben Sie bestimmte Parameter in den CICS-Assistenten an, um anzupassen, wie Zuordnungen zwischen Ihrem XML-Schema oder WSDL-Dokument und der Sprachstruktur gehandhabt werden.

XML-Datentypen können mithilfe von Facetten eingeschränkt werden. Verwenden Sie die Längenfacetten (`length`, `maxLength` und `minLength`) und die Facette `whiteSpace`, um anzupassen, wie Daten variabler Länge in Ihrer XML gehandhabt werden.

length Gibt an, dass die Daten eine feste Länge haben.

maxLength

Gibt die maximale Länge für den Datentyp an. Wenn dieser Wert für einen zeichenfolgebasierten Datentyp nicht festgelegt wird, gibt es keine maximale Länge.

minLength

Gibt die minimale Länge für den Datentyp an. Wenn dieser Wert für einen zeichenfolgebasierten Datentyp nicht festgelegt wird, ist die minimale Länge '0'.

whiteSpace

Gibt an, wie Leerraum um einen Datenwert herum gehandhabt wird. Als Leerraum werden Leerzeichen, Tabulatoren und Zeilenumbrüche bezeichnet. Die Facette `whiteSpace` kann auf `preserve`, `replace` oder `collapse` festgelegt werden:

- Der Wert `preserve` behält alle Leerräume in dem Datenwert bei.
- Bei dem Wert `replace` werden alle Tabulatoren oder Zeilenumbrüche durch die entsprechende Anzahl von Leerzeichen ersetzt.
- Bei dem Wert `collapse` werden führende, nachfolgende und eingebettete Leerräume entfernt und alle Tabulatoren, Zeilenumbrüche und aufeinanderfolgenden Leerzeichen durch einzelne Leerzeichen ersetzt.

Wenn die Facette `whiteSpace` nicht festgelegt ist, wird der Leerraum beibehalten.

Weitere Informationen zu XML-Schemafacetten finden Sie unter XML Schema Part 2: Datatypes Second Edition.

Die folgenden Parameter in den CICS-Assistenten DFHSC2LS und DFHWS2LS können verwendet werden, um die Art und Weise zu ändern, in der Daten variabler Länge aus dem XML-Schema der Sprachstruktur zugeordnet werden. Diese Parameter sind auf Zuordnungsebene 1.2 oder höher verfügbar.

DEFAULT-CHAR-MAXLENGTH

Gibt die Standardarraylänge von Zeichendaten in Zeichen für Zuordnungen an, in deren XML-Schema oder WSDL-Dokument keine Länge eingeschlossen ist. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 1 bis 2.147.483.647 sein.

Es wird jedoch empfohlen, die maximale Zeichenlänge, die DFHSC2LS oder DFHWS2LS verwenden sollen, direkt in Ihrem XML-Schema oder WSDL-Dokument mithilfe der Facette `maxLength` anzugeben. Indem Sie die

maximale Länge direkt in dem XML-Schema oder WSDL-Dokument angeben, vermeiden Sie Probleme, die auftreten können, wenn eine globale Standardeinstellung auf alle zeichenfolgebasierten Datentypen angewendet wird.

CHAR-VARYING-LIMIT

Gibt die maximale Größe von Zeichendaten variabler Länge an, die der Sprachstruktur zugeordnet werden. Wenn die Zeichendaten größer sind als der in diesem Parameter angegebene Wert, werden sie einem Container zugeordnet, und der Containername wird in der generierten Sprachstruktur verwendet. Der Wert kann zwischen 0 und dem Standardwert 32.767 Bytes liegen.

CHAR-VARYING

Gibt an, wie die Zeichendaten variabler Länge zugeordnet werden. Wenn Sie diesen Parameter nicht angeben, hängt die Standardzuordnung von der angegebenen Sprache ab. Sie haben folgende Optionen:

- **CHAR-VARYING=NO** gibt an, dass Zeichendaten variabler Länge als Zeichenfolgen mit fester Länge zugeordnet werden.
- **CHAR-VARYING=NULL** gibt an, dass Zeichendaten variabler Länge auf null endenden Zeichenfolgen zugeordnet werden.
- **CHAR-VARYING=YES** gibt an, dass Zeichendaten variabler Länge in PL/I einem CHAR VARYING-Datentyp zugeordnet werden. In den COBOL-, C- und C++-Sprachen werden Zeichendaten variabler Länge einer äquivalenten Darstellung zugeordnet, die zwei verwandte Elemente umfasst: die Datenlänge und die Daten.

CHAR-VARYING=YES führt in der Regel zu einer optimalen Leistung.

Werte variabler Länge aus Sprachstrukturen zu XML zuordnen

Sie können anpassen, wie Zuordnungen zwischen Ihrer Sprachstruktur und dem XML-Schema oder WSDL-Dokument gehandhabt werden. Legen Sie den Parameter **CHAR-VARYING** in DFHLS2SC oder DFHLS2WS auf COLLAPSE oder NULL fest, um zu ändern, wie diese Zeichenarrays generiert werden.

Das Festlegen der Option **CHAR-VARYING=NULL** weist CICS an, beim Generieren von XML ein Nullzeichen am Ende jedes Zeichenarrays hinzuzufügen.

Das Festlegen der Option **CHAR-VARYING=COLLAPSE** weist CICS an, beim Generieren von XML alle abschließenden Leerzeichen von Zeichenarrays automatisch zu entfernen. Diese Option ist nur auf Zuordnungsebene 2.1 oder höher verfügbar und **CHAR-VARYING=COLLAPSE** ist der Standardwert auf Zuordnungsebene 2.1 oder höher für alle Sprachen außer C und C++. Wenn die XML geparkt wird, werden alle führenden, abschließenden und eingebetteten Leerräume entfernt.

Variable Arrays von Elementen in DFHJS2LS

JSON kann Gruppen mit einer unterschiedlichen Anzahl an Elementen enthalten. Im Allgemeinen werden JSON-Schemas, die eine unterschiedliche Anzahl an Elementen enthalten, einer einzelnen Datenstruktur einer höheren Programmiersprache nicht effizient zugeordnet. CICS verarbeitet eine unterschiedliche Anzahl an Elementen in JSON-Daten mithilfe von containerbasierten Zuordnungen oder Inline-Zuordnungen.

Ein Array mit einer variierenden Anzahl von Elementen wird in dem JSON-Schema mithilfe der Schlüsselwörter `minItems` und `maxItems` in dem Schema mit einem "type"-Wert von "array" dargestellt:

- Das Schlüsselwort `minItems` gibt die Mindestanzahl von Vorkommen des Elements an. Es hat den Wert 0 oder eine beliebige positive ganze Zahl. Der Standardwert ist 0.
- Das Schlüsselwort `maxItems` gibt die Höchstanzahl von Vorkommen des Elements an. Es kann als Wert jede positive ganze Zahl größer-gleich dem Wert des Schlüsselworts `minItems` haben.
- Wenn das Schlüsselwort `maxItems` nicht angegeben ist, bedeutet das, dass das Array unbegrenzt ist.

Ein optionales Feld kann durch ein variables Array von `"maxItems":1` bezeichnet werden. Beispiel: Eine optionale 8-Byte-Zeichenfolge namens `"component"` :

```
"properties":{
  "component": {
    "type":"array",
    "maxItems":1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

Derselbe Effekt kann erzeugt werden, indem der Feldname nicht in den Schlüsselwortwert `"required"` eingeschlossen wird:

```
"properties":{
  "component": {
    "type": "string",
    "maxLength": 8
  }
}
```

Im Allgemeinen werden JSON-Schemas, die eine unterschiedliche Anzahl an Elementen enthalten, einer einzelnen Datenstruktur einer höheren Programmiersprache nicht effizient zugeordnet. Für diese Fälle verwendet CICS eine Reihe von verbundenen Datenstrukturen, die in einer Reihe von Containern an das Anwendungsprogramm übergeben werden. Diese Strukturen werden als Eingabe und Ausgabe der Anwendung verwendet:

- Wenn CICS JSON-Daten in Anwendungsdaten umsetzt, füllt es diese Strukturen mit den Anwendungsdaten, und die Anwendung liest sie.
- Wenn CICS die Anwendungsdaten in JSON-Daten umsetzt, liest es die Anwendungsdaten in den Strukturen, die von der Anwendung gefüllt wurden.

Das folgende Beispiel veranschaulicht das Format dieser Datenstrukturen. Diese Beispiele verwenden ein Array von einfachen 8-Byte-Feldern. Das Modell unterstützt jedoch Arrays komplexer Datentypen ebenso wie Arrays von Datentypen, die andere Arrays enthalten.

Beispiel 1. Feste Anzahl von Elementen

Dieses Beispiel stellt ein Element dar, das genau dreimal vorkommt.

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 3,
    "minItems": 3,
    "items": {
      "type": "string",

```

```
"maxLength": 8
}
},
"required": ["component"]
```

In diesem Beispiel ist die Anzahl der Vorkommen des Elements vorab bekannt, deshalb kann es als Array fester Länge in einer einfachen COBOL-Deklaration oder als das Äquivalent in anderen Sprachen dargestellt werden:

```
05 component PIC X(8) OCCURS 3 TIMES
```

Beispiel 2. Variierende Anzahl von Elementen auf Zuordnungsebene 2 und niedriger

Dieses Beispiel stellt ein obligatorisches Element dar, das ein- bis fünfmal vorkommen kann:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  }
},
"required": ["component"]
```

Die Hauptdatenstruktur enthält eine Deklaration mit zwei Feldern. Wenn CICS die JSON-Daten in binäre Daten umsetzt, enthält das erste Feld, `component-num`, die Anzahl von Vorkommen des Elements in den JSON-Daten und das zweite Feld, `component-cont`, den Namen eines Containers:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Eine zweite Datenstruktur enthält die Deklaration des Elements selbst:

```
01 DFHJS-component
02 component PIC X(8)
```

Sie müssen den Wert von `component-num` untersuchen, wobei es sich um einen Wert im Bereich zwischen 1 und 5 handelt, um herauszufinden, wie oft das Element vorkommt. Der Elementinhalt befindet sich im Container namens `component-cont`. Der Container enthält ein Array von Elementen, wobei jedes Element von der Datenstruktur `DFHJS-component` zugeordnet wird.

Wenn `minItems="0"` oder nicht angegeben ist und wenn `maxItems="1"`, ist das Element optional. Um die Datenstruktur in Ihrem Anwendungsprogramm zu verarbeiten, müssen Sie den Wert von `component-num` untersuchen:

- Ist er 0, hat die Nachricht kein Komponentenelement und der Inhalt von `component-cont` ist nicht definiert.
- Ist er 1, befindet sich das Komponentenelement im Container namens `component-cont`.

Der Inhalt des Containers wird von der Datenstruktur `DFHJS-component` zugeordnet.

Anmerkung: Wenn die JSON-Daten aus einem einzelnen wiederkehrenden Element bestehen, generiert DFHJS2LS zwei Sprachstrukturen. Die Hauptsprachstruktur enthält die Anzahl von Elementen in dem Array und den Namen eines Containers, der das Array von Elementen enthält. Die zweite Sprachstruktur ordnet eine einzelne Instanz des wiederkehrenden Elements zu.

Beispiel 3. Variierende Anzahl von Elementen auf Zuordnungsebene 2.1 und höher

Auf Zuordnungsebene 2.1 und höher können Sie den Parameter **INLINE-MAXOCCURS-LIMIT** in den CICS-Assistenten verwenden. Der Parameter **INLINE-MAXOCCURS-LIMIT** gibt an, wie die variierende Anzahl von Elementen verarbeitet wird. Bei den Zuordnungsoptionen für eine variierende Anzahl von Elementen handelt es sich containerbasierte Zuordnung, wie unter „Beispiel 2. Variierende Anzahl von Elementen auf Zuordnungsebene 2 und niedriger“ auf Seite 305 beschrieben, oder um eine Inline-Zuordnung. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein:

- Der Standardwert von **INLINE-MAXOCCURS-LIMIT** ist 1, was sicherstellt, dass optionale Elemente inline zugeordnet werden.
- Der Wert 0 für den Parameter **INLINE-MAXOCCURS-LIMIT** verhindert eine Inline-Zuordnung.
- Wenn `maxItems` kleiner-gleich dem Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die Inline-Zuordnung verwendet.
- Wenn `maxItems` größer als der Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die containerbasierte Zuordnung verwendet.

Die Inline-Zuordnung einer variierenden Anzahl von Elementen führt dazu, dass sowohl ein Array (wie in dem obigen Beispiel für die feste Anzahl von Vorkommen) als auch ein Zähler generiert wird. Das Feld `component-num` gibt an, wie viele Instanzen des Elements vorhanden sind, und auf diese wird vom Array verwiesen. Für das Beispiel in „Beispiel 2. Variierende Anzahl von Elementen auf Zuordnungsebene 2 und niedriger“ auf Seite 305, bei dem **INLINE-MAXOCCURS-LIMIT** kleiner-gleich 5 ist, sieht die generierte Datenstruktur wie folgt aus:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

Das erste Feld, `component-num`, ist identisch mit der Ausgabe für das Beispiel für die containerbasierte Zuordnung im vorherigen Abschnitt. Das zweite Feld enthält ein Array mit der Länge 5, das groß genug ist, um die maximale Anzahl von Elementen, die generiert werden können, zu enthalten.

Die Inline-Zuordnung unterscheidet sich von der containerbasierten Zuordnung, die die Anzahl von Vorkommen des Elements und den Namen des Containers, in dem die Daten enthalten sind, speichert, insofern als dass sie alle Daten in dem aktuellen Container speichert. Durch das Speichern der Daten im aktuellen Container wird die Leistung im Allgemeinen verbessert, was die Inline-Zuordnung zur präferierten Lösung macht.

Beispiel 4. Verschachtelte variierende Arrays

Komplexe JSON-Schemas können variierende wiederkehrende Elemente enthalten, die ihrerseits wiederum variierende wiederkehrende Elemente enthalten können. In diesem Fall erstreckt sich die beschriebene Struktur über die beiden in den Beispielen beschriebenen Ebenen hinaus.

Dieses Beispiel stellt ein optionales Element namens "component2" dar, das in einem obligatorischen Element namens "component1" verschachtelt ist, wobei das obligatorische Element ein- bis fünfmal vorkommen kann:

```
"properties":{
  "component1": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "object",
      "properties":{
        "component2":{
          "type": "string",
          "maxLength": 8
        }
      },
      "required": ["component2"]
    }
  },
  "required": ["component1"]
}
```

Die übergeordnete Datenstruktur ist identisch mit früheren Beispielen:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

Die zweite Datenstruktur enthält jedoch die folgenden Elemente:

```
01 DFHJS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Eine Struktur auf dritter Ebene enthält die folgenden Elemente:

```
01 DFHJS-component2
02 component2 PIC X(8)
```

Die Anzahl von Vorkommen des äußersten Elements "component1" ist in component1-num angegeben.

Der in component1-cont benannte Container enthält ein Array mit dieser Anzahl von Instanzen der zweiten Datenstruktur DFHJS-component1.

Jede Instanz von component2-cont benennt einen anderen Container, der jeweils die Datenstruktur enthält, die von der Struktur auf dritter Ebene, DFHJS-component2, zugeordnet wird.

Um diese Struktur zu veranschaulichen, ziehen Sie das Fragment von JSON-Daten in Betracht, das dem folgenden Beispiel entspricht:

```
{ "component1":
  [
    {
      "component2": "string1"
    },
    {
      "component2": "string2"
    }
  ]
}
```

"component1" kommt dreimal vor. Die ersten beiden Vorkommen enthalten eine Instanz von "component2" , das dritte Vorkommen nicht.

In der übergeordneten Datenstruktur enthält component-num den Wert 3. Der in component1-cont benannte Container enthält drei Instanzen von DFHJS-component1 :

1. In der ersten Instanz hat component2-num den Wert 1 und der in component2-cont benannte Container enthält *string1*.
2. In der zweiten Instanz hat component2-num den Wert 1 und der in component2-cont benannte Container enthält *string2*.
3. In der dritten Instanz hat component2-num den Wert 0 und der Inhalt von component2-cont ist nicht definiert.

In dieser Instanz wird die gesamte Datenstruktur von vier Containern dargestellt:

- Stammdatenstruktur in Container DFHJS-DATA
- In component1-cont benannter Container
- In den ersten beiden Instanzen von component2-cont benannte zwei Container

Optionale Strukturen und das Schlüsselwort **required**

Datenstrukturen werden durch das JSON-Schema "type" von "object" definiert. Die Schemas setzen Feldnamen zu einzelnen Typen in Bezug mithilfe des durch das Schlüsselwort "properties" bereitgestellten Objekts. Die Anforderung, dass diese Felder Bestandteil der JSON-Daten sein müssen, die durch das JSON-Schema beschrieben werden, wird von dem im Schlüsselwort "required" angegebenen Array gesteuert. In diesem Array werden alle Feldnamen aufgelistet, die in den JSON-Daten vorhanden sein müssen. Optionale Felder werden daher entweder durch ihr Fehlen in diesem Array dargestellt oder durch das Fehlen des Schlüsselworts "required", da das Array nicht leer sein darf. In diesem Fall sind alle Felder optional.

Optionale Felder werden als variierendes Array von null oder einem Element behandelt. Dadurch wird ein zusätzliches Feld mit dem Suffix "-num" an den Elementnamen angehängt. Wenn die Gesamtlänge mehr als 28 Zeichen beträgt, wird der Elementname abgeschnitten. Zur Laufzeit wird dies ungleich null sein, um anzugeben, dass der Wert in den JSON-Daten vorhanden war, und gleich null, wenn er nicht vorhanden war.

In diesem Beispiel werden zwei Felder dargestellt, ein obligatorisches namens "required-structure" und ein optionales namens "optional-structure" :

```
{
  "type": "object",
  "properties": {
    "required-structure": {
      "type": "string",
      "maxLength": 8
    },
    "optional-structure": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": [
    "required-structure"
  ]
}
```

Die generierte COBOL-Struktur zeigt beide Felder an, aber dem zweiten wird "optional-structure-num" vorangestellt. Dies ist eine ganzzahlige Angabe der Elemente, wobei 0 angibt, dass keine Elemente vorhanden sind, und 1 angibt, dass

eines vorhanden ist. Der Wert wird festgelegt, um anzugeben, ob "optional-structure" gültige Daten enthält oder nicht.

03 OutputData.

06 required-structure PIC X(8).

06 optional-structure-num PIC S9(9) COMP-5 SYNC.

06 optional-structure PIC X(8).

Variable Arrays von Elementen

XML kann ein Array mit einer variierenden Anzahl von Elementen enthalten. Allgemein lassen sich WSDL-Dokumente und XML-Schemas, die eine variierende Anzahl von Elementen enthalten, nicht effizient einer einzelnen Datenstruktur einer höheren Programmiersprache zuordnen. CICS verwendet containerbasierte Zuordnungen oder Inlinezuordnungen, um eine variierende Anzahl von Elementen in XML zu verarbeiten.

Ein Array mit einer variierenden Anzahl von Elementen wird im XML-Schema mithilfe der Attribute `minOccurs` und `maxOccurs` in der Elementdeklaration dargestellt:

- Das Attribut `minOccurs` gibt an, wie oft das Element mindestens vorkommen kann. Es hat den Wert 0 oder eine beliebige positive ganze Zahl.
- Das Attribut `maxOccurs` gibt an, wie oft das Element höchstens vorkommen kann. Es kann als Wert jede positive ganze Zahl größer-gleich dem Wert des Attributs `minOccurs` haben. Es kann auch den Wert `unbounded` haben, der angibt, dass für die Anzahl der Vorkommen des Elements keine Obergrenze gilt.
- Der Standardwert für beide Attribute ist 1.

In diesem Beispiel wird eine 8-Byte-Zeichenfolge dargestellt, die optional ist. Das heißt, sie kann gar nicht oder einmal in der Anwendungs-XML oder der SOAP-Nachricht vorkommen:

```
<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>
```

Im folgenden Beispiel wird eine 8-Byte-Zeichenfolge dargestellt, die mindestens einmal auftreten muss:

```
<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>
```

Allgemein lassen sich WSDL-Dokumente, die eine variierende Anzahl von Elementen enthalten, nicht effizient einer einzelnen Datenstruktur einer höheren Programmiersprache zuordnen. Für diese Fälle verwendet CICS folglich eine Reihe von verbundenen Datenstrukturen, die in einer Reihe von Containern an das Anwendungsprogramm übergeben werden. Diese Strukturen werden als Eingabe und Ausgabe der Anwendung verwendet:

- Wenn CICS XML in Anwendungsdaten umsetzt, füllt es diese Strukturen mit den Anwendungsdaten, und die Anwendung liest sie.

- Wenn CICS die Anwendungsdaten in XML umsetzt, liest es die Anwendungsdaten in den Strukturen, die von der Anwendung gefüllt wurden.

Das Format dieser Datenstrukturen lässt sich am besten anhand einer Reihe von Beispielen erklären. Die XML kann aus einer SOAP-Nachricht oder aus einer Anwendung stammen. Diese Beispiele verwenden ein Array von einfachen 8-Byte-Feldern. Das Modell unterstützt jedoch Arrays komplexer Datentypen ebenso wie Arrays von Datentypen, die andere Arrays enthalten.

Feste Anzahl von Elementen

Das erste Beispiel stellt ein Element dar, das genau dreimal vorkommt.

```
<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>
```

Da in diesem Beispiel die Anzahl der Vorkommen des Elements vorab bekannt ist, kann es als Array fester Länge in einer einfachen COBOL-Deklaration (oder dem Äquivalent in anderen Sprachen) dargestellt werden:

```
05 component PIC X(8) OCCURS 3 TIMES
```

Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner

Dieses Beispiel stellt ein obligatorisches Element dar, das ein- bis fünfmal vorkommen kann:

```
<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>
```

Die Hauptdatenstruktur enthält eine Deklaration mit zwei Feldern. Wenn CICS die XML in binäre Daten umsetzt, enthält das erste Feld, component-num, die Anzahl von Vorkommen des Elements in der XML und das zweite Feld, component-cont, den Namen eines Containers:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Eine zweite Datenstruktur enthält die Deklaration des Elements selbst:

```
01 DFHWS-component
02 component PIC X(8)
```

Sie müssen den Wert von component-num untersuchen (Wert im Bereich zwischen 1 und 5), um herauszufinden, wie oft das Element vorkommt. Der Elementinhalt befindet sich im Container namens component-cont. Der Container enthält ein Array von Elementen, wobei jedes Element von der Datenstruktur DFHWS-component zugeordnet ist.

Wenn `minOccurs="0"` und `maxOccurs="1"`, ist das Element optional. Um die Datenstruktur in Ihrem Anwendungsprogramm zu verarbeiten, müssen Sie den Wert von `component-num` untersuchen:

- Ist er 0, hat die Nachricht kein Komponentenelement und der Inhalt von `component-cont` ist nicht definiert.
- Ist er 1, befindet sich das Komponentenelement im Container namens `component-cont`.

Der Inhalt des Containers wird von der Datenstruktur `DFHWS-component` zugeordnet.

Anmerkung: Wenn die SOAP-Nachricht aus einem einzelnen wiederkehrenden Element besteht, generiert `DFHWS2LS` zwei Sprachstrukturen. Die Hauptsprachstruktur enthält die Anzahl von Elementen in dem Array und den Namen eines Containers, der das Array von Elementen enthält. Die zweite Sprachstruktur ordnet eine einzelne Instanz des wiederkehrenden Elements zu.

Variierende Anzahl von Elementen auf Zuordnungsebene 2.1 und höher

Auf Zuordnungsebene 2.1 und höher können Sie den Parameter **INLINE-MAXOCCURS-LIMIT** in den CICS-Assistenten verwenden. Der Parameter **INLINE-MAXOCCURS-LIMIT** gibt an, wie die variierende Anzahl von Elementen verarbeitet wird. Bei den Zuordnungsoptionen für eine variierende Anzahl von Elementen handelt es sich containerbasierte Zuordnung, wie unter „Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner“ auf Seite 373 beschrieben, oder um eine Inline-Zuordnung. Der *wert* dieses Parameters kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein:

- Der Standardwert von **INLINE-MAXOCCURS-LIMIT** ist 1, was sicherstellt, dass optionale Elemente inline zugeordnet werden.
- Der Wert 0 für den Parameter **INLINE-MAXOCCURS-LIMIT** verhindert eine Inline-Zuordnung.
- Wenn `maxOccurs` kleiner-gleich dem Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die Inline-Zuordnung verwendet.
- Wenn `maxOccurs` größer als der Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die containerbasierte Zuordnung verwendet.

Die Inline-Zuordnung einer variierenden Anzahl von Elementen führt dazu, dass sowohl ein Array (wie in dem obigen Beispiel für die feste Anzahl von Vorkommen) als auch ein Zähler generiert wird. Das Feld `component-num` gibt an, wie viele Instanzen des Elements vorhanden sind, und auf diese wird vom Array verwiesen. Für das Beispiel in „Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner“ auf Seite 373, bei dem **INLINE-MAXOCCURS-LIMIT** kleiner-gleich 5 ist, sieht die generierte Datenstruktur wie folgt aus:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

Das erste Feld, `component-num`, ist identisch mit der Ausgabe für das Beispiel für die containerbasierte Zuordnung im vorherigen Abschnitt. Das zweite Feld enthält ein Array mit der Länge 5, das groß genug ist, um die maximale Anzahl von Elementen, die generiert werden können, zu enthalten.

Die Inline-Zuordnung unterscheidet sich von der containerbasierten Zuordnung, die die Anzahl von Vorkommen des Elements und den Namen des Containers, in dem die Daten enthalten sind, speichert, insofern als dass sie alle Daten in dem aktuellen Container speichert. Durch das Speichern der Daten im aktuellen Container

wird die Leistung im Allgemeinen verbessert, was die Inline-Zuordnung zur präferierten Lösung macht.

Verschachtelte variierende Arrays

Komplexe WSDL-Dokumente und XML-Schemas können variierende wiederkehrende Elemente enthalten, die ihrerseits wiederum variierende wiederkehrende Elemente enthalten können. In diesem Fall erstreckt sich die beschriebene Struktur über die beiden in den Beispielen beschriebenen Ebenen hinaus.

Dieses Beispiel stellt ein optionales Element namens <component2> dar, das in einem obligatorischen Element namens <component1> verschachtelt ist, wobei das obligatorische Element ein- bis fünfmal vorkommen kann.

```
<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>
```

Die übergeordnete Datenstruktur ist identisch mit früheren Beispielen:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

Die zweite Datenstruktur enthält jedoch die folgenden Elemente:

```
01 DFHWS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Eine Struktur auf dritter Ebene enthält die folgenden Elemente:

```
01 DFHWS-component2
02 component2 PIC X(8)
```

Die Anzahl von Vorkommen des äußersten Elements <component1> ist in component1-num angegeben.

Der in component1-cont benannte Container enthält ein Array mit dieser Anzahl von Instanzen der zweiten Datenstruktur DFHWS-component1.

Jede Instanz von component2-cont benennt einen anderen Container, der jeweils die Datenstruktur enthält, die von der Struktur auf dritter Ebene, DFHWS-component2, zugeordnet wird.

Um diese Struktur zu veranschaulichen, ziehen Sie das XML-Fragment in Betracht, das dem folgenden Beispiel entspricht:

```
<component1><component2>
  string1
</component2></component1>
```

```

<component1><component2>
string2
</component2></component1>
<component1></component1>

```

<component1> kommt dreimal vor. Die ersten beiden Vorkommen enthalten jeweils eine Instanz von <component2>, die dritte Instanz jedoch nicht.

In der übergeordneten Datenstruktur enthält component1-num den Wert 3. Der in component1-cont benannte Container enthält drei Instanzen von DFHWS-component1 :

1. In der ersten Instanz hat component2-num den Wert 1 und der in component2-cont benannte Container enthält *string1*.
2. In der zweiten Instanz hat component2-num den Wert 1 und der in component2-cont benannte Container enthält *string2*.
3. In der dritten Instanz hat component2-num den Wert 0 und der Inhalt von component2-cont ist nicht definiert.

In dieser Instanz wird die gesamte Datenstruktur von vier Containern dargestellt:

- Stammdatenstruktur in Container DFHWS-DATA
- In component1-cont benannter Container
- In den ersten beiden Instanzen von component2-cont benannte zwei Container

Optionale Strukturen und 'xsd:choice'

DFHWS2LS und DFHSC2LS unterstützen die Verwendung von minOccurs und maxOccurs in den Elementen <xsd:sequence>, <xsd:choice> und <xsd:all> nur auf Zuordnungsebene 2.1 und höher, wobei die Attribute minOccurs und maxOccurs auf minOccurs="0" und maxOccurs="1" gesetzt sind.

Die Assistenten generieren Zuordnungen, die diese Elemente behandeln, als seien die in ihnen enthaltenen untergeordneten Elemente optional. Wenn Sie eine Anwendung mit diesen Elementen implementieren, stellen Sie sicher, dass keine ungültigen Kombinationen von Optionen von der Anwendung generiert werden. Jedes dieser Elemente hat ein eigenes Feld count in der generierten Sprachstruktur; diese Felder müssen entweder alle auf "0" oder alle auf "1" gesetzt sein. Jede andere Kombination von Werten ist ungültig, außer mit <xsd:choice>-Elementen.

<xsd:choice>-Elemente geben an, dass nur eine der Optionen in dem Element verwendet werden kann. Dies wird auf allen Zuordnungsebenen unterstützt. Die Assistenten verarbeiten alle diese Optionen in einem <xsd:choice>-Element als ob es sich um ein <xsd:sequence>-Element mit minOccurs="0" und maxOccurs="1" handelt. Stellen Sie beim Implementieren einer Anwendung mithilfe des <xsd:choice>-Elements sicher, dass keine ungültigen Kombinationen von Optionen von der Anwendung generiert werden. Jedes der Elemente hat ein eigenes Feld count in der generierten Sprachstruktur. Von diesen muss genau eines den Wert 1 haben, die anderen müssen alle den Wert 0 haben. Alle anderen Kombinationen von Werten sind ungültig, außer wenn das <xsd:choice>-Element selbst optional ist. In diesem Fall können alle Felder den Wert 0 haben.

Unterstützung für XML-Attribute

XML-Schemas können Attribute angeben, die in XML zulässig oder erforderlich sind. Die Dienstprogramme DFHWS2LS und DFHSC2LS des CICS-Assistenten ignorieren XML-Attribute standardmäßig. Für die Verarbeitung von XML-Attributen, die im XML-Schema definiert sind, muss der Wert des Parameters **MAPPING-LEVEL** 1.1 oder höher sein.

Optionale Attribute

Attribute können optional oder erforderlich sein. Sie können jedem Element in einer SOAP-Nachricht oder in einer XML für eine Anwendung zugeordnet werden. Für jedes optionale Attribut, das im Schema definiert ist, werden zwei Felder in der entsprechenden Sprachstruktur generiert:

1. Ein Vorhandensein-Flag; dieses Feld wird als boolescher Datentyp behandelt und hat typischerweise eine Länge von 1 Byte.
2. Ein Wert; dieses Feld wird auf dieselbe Weise zugeordnet wie ein XML-Element eines äquivalenten Typs. Beispielsweise wird ein Attribut vom Typ NMTOKEN auf dieselbe Weise zugeordnet wie ein XML-Element vom Typ NMTOKEN.

Das Vorhandensein des Attributs und die Wertfelder werden in der generierten Sprachstruktur vor dem Feld für das Element angezeigt, dem Sie zugeordnet sind. Unerwartete Attribute, die im Instanzdokument angezeigt werden, werden ignoriert.

Sehen Sie sich beispielsweise die folgende Schemaattributdefinition an:

```
<xsd:attribute name="age" type="xsd:short"
use="optional" />
```

Dieses optionale Attribut ist der folgenden COBOL-Struktur zugeordnet:

```
05 attr-age-exist PIC X DISPLAY
05 attr-age-value PIC S9999 COMP-5 SYNC
```

Laufzeitverarbeitung optionaler Attribute

Die folgende Laufzeitverarbeitung erfolgt für optionale Attribute:

- Wenn das Attribut vorhanden ist, wird das Vorhandensein-Flag gesetzt und der Wert wird zugeordnet.
- Wenn das Attribut nicht vorhanden ist, wird das Vorhandensein-Flag nicht gesetzt.
- Wenn das Attribut einen Standardwert hat und vorhanden ist, wird der Wert zugeordnet.
- Wenn das Attribut einen Standardwert hat und nicht vorhanden ist, wird der Standardwert zugeordnet.

Optionale Attribute mit Standardwerten werden als erforderliche Attribute behandelt.

Wenn CICS die Daten in XML umsetzt, erfolgt die folgende Laufzeitverarbeitung:

- Wenn das Vorhandensein-Flag gesetzt ist, wird das Attribut umgesetzt und in die XML eingeschlossen.
- Wenn das Vorhandensein-Flag nicht gesetzt ist, wird das Attribut nicht in die XML eingeschlossen.

Erforderliche Attribute und Laufzeitverarbeitung

Für jedes erforderliche Attribut wird nur das Wertfeld in der entsprechenden Sprachstruktur generiert.

Wenn das Attribut in der XML-Datei vorhanden ist, wird der Wert zugeordnet. Wenn das Attribut nicht vorhanden ist, erfolgt die folgende Verarbeitung:

- Wenn es sich bei der Anwendung um einen Web-Service-Provider handelt, generiert CICS eine SOAP-Fehlernachricht, die auf einen Fehler in der Client-SOAP-Nachricht hinweist.
- Wenn es sich bei der Anwendung um einen Web-Service-Requester handelt, gibt CICS eine Nachricht aus und eine Konvertierungsfehlerantwort mit einem RESP2-Code von 13 an die Anwendung zurück.
- Wenn die Anwendung den Befehl **TRANSFORM XMLTODATA** verwendet, gibt CICS eine Nachricht aus und eine Antwort bezüglich einer ungültigen Anforderung mit einem RESP2-Code von 3 an die Anwendung zurück.

Wenn CICS eine SOAP-Nachricht basierend auf dem Inhalt eines Kommunikationsbereichs oder Containers erstellt, wird das Attribut umgesetzt und in die Nachricht eingeschlossen. Wenn eine Anwendung den Befehl **TRANSFORM DATATOXML** verwendet, wandelt CICS das Attribut ebenfalls um und schließt es in die XML ein.

Auf Nil festlegbares Attribut

Ein auf Nil festlegbares Attribut ist ein spezielles Attribut, das für ein `xsd:element` in einem XML-Schema angezeigt werden kann. Es gibt an, dass das Attribut `xsi:nil` für das Element in XML gültig ist. Wenn für ein Element das Attribut `xsi:nil` angegeben wurde, bedeutet dies, dass das Element vorhanden ist, aber keinen Wert hat. Daher ist ihm kein Inhalt zugeordnet.

Wenn in einem XML-Schema das auf Nil festlegbare Attribut als 'true' definiert ist, wird es als erforderliches Attribut zugeordnet, das einen booleschen Wert annimmt.

Wenn CICS eine SOAP-Nachricht empfängt oder XML für eine Anwendung umsetzen muss, die ein Attribut `xsi:nil` enthält, kann der Wert des Attributs 'true' oder 'false' sein. Lautet der Wert 'true', muss die Anwendung die Werte des Elements oder der verschachtelten Elemente im Geltungsbereich des Attributs `xsi:nil` ignorieren.

Wenn CICS eine SOAP-Nachricht oder XML basierend auf dem Inhalt eines Kommunikationsbereichs oder Containers erstellt, für den der Wert für das Attribut `xsi:nil` 'true' lautet, erfolgt die folgende Verarbeitung:

- Das Attribut `xsi:nil` wird in der XML- oder der SOAP-Nachricht generiert.
- Der Wert des zugeordneten Elements wird ignoriert.
- Alle in dem Element verschachtelten Elemente werden ignoriert.

Beispiel für eine SOAP-Nachricht

Sehen Sie sich das folgende XML-Beispielschema an, das Teil eines WSDL-Dokuments sein könnte:

```
<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>
```

Im Folgenden sehen Sie ein Beispiel für eine partielle SOAP-Nachricht, die diesem Schema entspricht:

```

<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <num xsi:nil="true"/>
  <num>15</num>
  <num xsi:nil="true"/>
</root>

```

In COBOL wird diese SOAP-Nachricht den folgenden Elementen zugeordnet:

```

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)

```

Unterstützung für <xsd:any> und xsd:anyType

DFHWS2LS und DFHSC2LS unterstützen die Verwendung von <xsd:any> und xsd:anyType im XML-Schema. Sie können das XML-Schemaelement <xsd:any> verwenden, um einen Abschnitt eines XML-Dokuments mit nicht definiertem Inhalt zu beschreiben. xsd:anyType ist der Basisdatentyp, aus dem alle einfachen und komplexen Datentypen abgeleitet werden. Er kennt keine Einschränkungen für die Dateninhalte.

Bevor Sie <xsd:any> und xsd:anyType mit den CICS-Assistenten verwenden können, legen Sie die folgenden Parameter fest:

- Legen Sie den Parameter **MAPPING-LEVEL** auf 2.1 oder höher fest.
- Für eine Web-Service-Provider-Anwendung geben Sie für den Parameter **PGMINT CHANNEL** an.

<xsd:any> - Beispiel

In diesem Beispiel wird ein <xsd:any>-Element verwendet, um optionalen unstrukturierten XML-Inhalt nach dem Tag "Surname" im Tag "Customer" zu beschreiben:

```

<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>

```

Im Folgenden sehen Sie ein Beispiel für eine SOAP-Nachricht, die diesem XML-Schema entspricht:

```

<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">
          <!-- Dieser Tag 'ExtraInformation' ist dem optionalen 'xsd:any'-Element
               in dem XML-Schema zugeordnet.
               Er kann jede korrekt formatierte XML enthalten. -->
          <ExampleField1>one</ExampleField1>
          <ExampleField2>two</ExampleField2>
        </ExtraInformation>
      </Customer>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>

```



```

</ExtraInformation>
</Customer>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Wenn diese SOAP-Nachricht an CICS gesendet wird, füllt CICS den Container Customer-xml-cont mit den folgenden XML-Daten:

```

<ExtraInformation
  xmlns="http://www.example.org/ExtraInformation">
  <!-- Dieser Tag 'ExtraInformation' ist dem optionalen 'xsd:any'-Element
in dem XML-Schema zugeordnet.
Er kann jede korrekt formatierte XML enthalten. -->
  <ExampleField1>one</ExampleField1>
  <ExampleField2>two</ExampleField2>
</ExtraInformation>

```

CICS füllt auch den Container Customer-xmlns-cont mit den folgenden gültigen XML-Namensbereichsdeklarationen. Diese Deklarationen sind durch ein Leerzeichen getrennt:

```

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://www.example.org/anyExample"

```

xsd:anyType - Beispiel

xsd:anyType ist der Basisdatentyp, aus dem alle einfachen und komplexen Datentypen abgeleitet werden. Er kennt keine Einschränkungen für Dateninhalte. Wenn Sie keinen Datentyp angeben, wird standardmäßig der Wert xsd:anyType angenommen. Die beiden folgenden XML-Fragmente sind beispielsweise äquivalent:

```

<xsd:element name="Name" type="xsd:anyType"/>
<xsd:element name="Name"/>

```

Generierte Sprachstrukturen

Die für <xsd:any> oder xsd:anyType generierten Sprachstrukturen haben das folgende Format in COBOL und ein äquivalentes Format für die anderen Sprachen:

elementName-xml-cont PIC X(16)

Der Name eines Containers, der die unformatierte XML enthält. Wenn CICS eine eingehende SOAP-Nachricht verarbeitet, stellt es das Subset der SOAP-Nachricht, das von <xsd:any> oder xsd:anyType definiert wird, in diesen Container. Die Anwendung kann die XML-Daten nur nativ verarbeiten. Die Anwendung muss die XML generieren, diesen Container mit Daten füllen und den Containernamen angeben.

Dieser Container muss im Textmodus gefüllt werden. Wenn CICS diesen Container mit Daten füllt, geschieht dies unter Verwendung derselben Variante von EBCDIC wie für den Web-Service definiert wurde. Zeichen, die in der EBCDIC-Zielcodepage nicht vorhanden sind, werden durch Substitutionszeichen ersetzt, auch wenn der Container von der Anwendung in UTF-8 gelesen wird.

elementName-xmlns-cont PIC X(16)

Der Name eines Containers, der alle gültigen Namensbereichspräfixdeklarationen enthält. Der Inhalt dieses Containers ist dem Inhalt des Containers DFHWS-XMLNS ähnlich, mit der Ausnahme, dass er alle gültigen und relevanten Namensbereichsdeklarationen enthält, und nicht nur das Subset des SOAP-Envelope-Tags.

Dieser Container muss im Textmodus gefüllt werden. Wenn CICS diesen Container mit Daten füllt, geschieht dies unter Verwendung derselben Variante von EBCDIC wie für den Web-Service definiert wurde. Zeichen, die in der EBCDIC-Zielcodepage nicht vorhanden sind, werden durch Substitutionszeichen ersetzt, auch wenn der Container von der Anwendung in UTF-8 gelesen wird.

Dieser Container wird nur verwendet, wenn SOAP-Nachrichten verarbeitet werden, die an CICS gesendet wurden. Falls die Anwendung versucht, einen Container mit Namensbereichsdeklarationen bereitzustellen, wenn eine SOAP-Ausgabenachricht generiert wird, werden der Container und sein Inhalt von CICS ignoriert. CICS erfordert, dass die von der Anwendung bereitgestellte XML im Hinblick auf Namensbereichsdeklarationen vollkommen eigenständig ist.

Der Name des XML-Elements, das das `<xsd:any>`-Element enthält, ist in den Variablennamen enthalten, die für das `<xsd:any>`-Element generiert werden. In dem Beispiel für `<xsd:any>`, ist das `<xsd:any>`-Element in dem Element `<xsd:element name="Customer">` verschachtelt und die Variablennamen, die für das `<xsd:any>`-Element generiert werden, lauten `Customer-xml-cont PIC X(16)` und `Customer-xmlns-cont PIC X(16)`.

Für einen `xsd:anyType`-Typ wird der direkte XML-Elementname verwendet. In dem obigen Beispiel für `xsd:anyType` lauten die Variablennamen `Name-xml-cont PIC X(16)` und `Name-xmlns-cont PIC X(16)`.

Unterstützung für `<xsd:choice>`

Ein Element `<xsd:choice>` gibt an, dass nur eine der Optionen im Element verwendet werden kann. Die CICS-Assistenten stellen in unterschiedlichem Umfang Unterstützung für `<xsd:choice>`-Elemente auf den verschiedenen Zuordnungsebenen bereit.

Unterstützung für `<xsd:choice>` auf Zuordnungsebene 2.2 und höher

Auf Zuordnungsebene 2.2 und höher stellen DFHWS2LS und DFHSC2LS eine verbesserte Unterstützung für `<xsd:choice>`-Elemente bereit. Die Assistenten generieren einen neuen Container, der den Wert speichert, der dem `<xsd:choice>`-Element zugeordnet ist. Die Assistenten generieren Sprachstrukturen, die den Namen eines neuen Containers und ein zusätzliches Feld enthalten:

fieldname -enum

Das differenzierende Feld zur Angabe, welche der Optionen von dem `<xsd:choice>`-Element verwendet werden.

fieldname -cont

Der Name des Containers, in dem die zu verwendende Option gespeichert wird. Es wird eine weitere Sprachstruktur generiert, um den Optionswert zuzuordnen.

Das folgende XML-Schemafragment enthält ein `<xsd:choice>`-Element:

```
<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>
```

Wenn dieses XML-Schemafragment auf Zuordnungsebene 2.2 oder höher verarbeitet wird, generiert der Assistent die folgenden COBOL-Sprachstrukturen:

```
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.
```

Einschränkungen für <xsd:choice> auf Zuordnungsebene 2.2 und höher

DFHSC2LS und DFHWS2LS unterstützen keine verschachtelten <xsd:choice>-Elemente. Beispielsweise wird die folgende XML nicht unterstützt:

```
<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 und DFHWS2LS unterstützen keine wiederkehrenden <xsd:choice>-Elemente. Beispielsweise wird die folgende XML nicht unterstützt:

```
<xsd:choice maxOccurs="2">  
<xsd:element name="name1" type="string"/>  
</xsd:choice>
```

DFHSC2LS und DFHWS2LS unterstützen höchstens 255 Optionen in einem <xsd:choice>-Element.

Unterstützung für <xsd:choice> auf Zuordnungsebene 2.1 und niedriger

Auf Zuordnungsebene 2.1 und niedriger stellt DFHWS2LS eingeschränkte Unterstützung für <xsd:choice>-Elemente bereit. DFHWS2LS behandelt jede einzelne Option in einem <xsd:choice>-Element, als ob es sich um ein <xsd:sequence>-Element handelte, das höchstens einmal vorkommen kann.

Nur eine der Optionen in einem <xsd:choice>-Element kann verwendet werden, deshalb müssen Sie beim Implementieren einer Anwendung mit dem <xsd:choice>-Element darauf achten, dass Sie nur gültige Kombinationen von Optionen generieren. Jedes der Elemente hat ein eigenes Feld count in der generierten Sprachstruktur. Von diesen muss genau eines den Wert 1 haben, die anderen müssen alle den Wert 0 haben. Alle anderen Kombinationen von Werten sind falsch, außer wenn <xsd:choice> selbst optional ist. In diesem Fall können alle Felder den Wert 0 haben.

Unterstützung für <xsd:sequence>

Die Verwendung von <xsd:sequence>-Elementen mit den CICS-Assistenten wird unterstützt, allerdings gibt es eine Reihe von Einschränkungen.

Die Verwendung der Attribute minOccurs und maxOccurs wird für das <xsd:sequence>-Element nicht unterstützt. Eine Ausnahme dieser Regel gilt, wenn minOccurs="0" und maxOccurs="1" oder minOccurs="1" und maxOccurs="1".

Die CICS-Assistenten lassen nicht zu, dass zwei Elemente mit demselben Namen in demselben <xsd:sequence>-Element verwendet werden. Beispiel: Die Sequenz {a, b, c, a} wird von den CICS-Assistenten zurückgewiesen. Vermeiden Sie diese Einschränkung, indem Sie die Sequenz durch {a maxOccurs="2", b, c} ersetzen.

Die CICS-Assistenten lassen nur ein <xsd:any>-Element oder ein Element, das als <xsd:any> behandelt wird, in demselben <xsd:sequence>-Element zu. Elemente, die als <xsd:any>-Elemente behandelt werden, sind unter anderem abstrakte Elemente, für die keine Substitutionsgruppe definiert ist.

Beim Parsen von XML behandelt CICS <xsd:sequence>-Elemente wie <xsd:all>-Elemente. Das heißt, dass CICS nicht prüft, ob die Reihenfolge der Elemente in der Sequenz korrekt ist. Beispiel: Wenn das Schema die Sequenz {a, b, c} definiert, toleriert CICS XML, das a, b und c in beliebiger Reihenfolge enthält. Wenn CICS jedoch XML generiert, wird die Reihenfolge von Elementen in einer <xsd:sequence> immer beibehalten.

CICS erkennt nicht automatisch fehlende XML-Daten. Beispiel: Wenn ein Element in der <xsd:sequence> als obligatorisch definiert ist (minOccurs="1" maxOccurs="1"), aber im XML-Dokument nicht vorkommt, berichtet CICS dieses Problem nur, wenn die Laufzeitvalidierung aktiviert ist.

Unterstützung für Substitutionsgruppen

Mit einer Substitutionsgruppe können Sie eine Gruppe von austauschbaren XML-Elementen definieren. Die CICS-Assistenten stellen Unterstützung für Substitutionsgruppen auf Zuordnungsebene 2.2 und höher bereit.

Auf Zuordnungsebene 2.2 und höher unterstützen DFHSC2LS und DFHWS2LS Substitutionsgruppen mithilfe ähnlicher Zuordnungen wie denjenigen für <xsd:choice>-Elemente. Der Assistent generiert ein Aufzählungsfeld und einen neuen Containernamen in der Sprachstruktur.

Das folgende XML-Schemafragment umfasst ein Array von zwei subGroupParent-Elementen, von denen jedes durch replacementOption1 oder replacementOption2: ersetzt werden kann.

```
<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" />
```

Die Verarbeitung dieses XML-Fragments mit dem Assistenten generiert die folgenden COBOL-Sprachstrukturen:

```
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).
```

Weitere Informationen zu Substitutionsgruppen finden Sie unter XML Schema Part 1: Structures Second Edition.

Unterstützung für abstrakte Elemente und abstrakte Datentypen

Die CICS-Assistenten stellen Unterstützung für abstrakte Elemente und abstrakte Datentypen auf Zuordnungsebene 2.2 und höher bereit. Die CICS-Assistenten ordnen abstrakte Elemente und abstrakte Datentypen in ähnlicher Weise zu wie Substitutionsgruppen.

Unterstützung für abstrakte Elemente auf Zuordnungsebene 2.2 und höher

Auf Zuordnungsebene 2.2 und höher behandeln DFHSC2LS und DFHWS2LS abstrakte Elemente fast genauso wie Substitutionsgruppen, mit der Ausnahme, dass das abstrakte Element kein gültiges Mitglied der Gruppe ist. Wenn keine Ersetzungselemente vorhanden sind, wird das abstrakte Element als `<xsd:any>`-Element behandelt und verwendet dieselben Zuordnungen wie ein `<xsd:any>`-Element auf Zuordnungsebene 2.1.

Das folgende XML-Schemafragment gibt zwei Optionen an, die anstelle des abstrakten Elements verwendet werden können. Das abstrakte Element selbst ist keine gültige Option:

```
<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" />
```

Die Verarbeitung dieses XML-Fragments mit dem Assistenten generiert die folgenden COBOL-Sprachstrukturen:

```
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.
```

Weitere Informationen zu abstrakten Elementen finden Sie unter XML Schema Part 0: Primer Second Edition.

Unterstützung für abstrakte Datentypen auf Zuordnungsebene 2.2 und höher

Auf Zuordnungsebene 2.2 und höher behandeln DFHSC2LS und DFHWS2LS abstrakte Datentypen als Substitutionsgruppen. Der Assistent generiert ein Aufzählungsfeld und einen neuen Containernamen in der Sprachstruktur.

Das folgende XML-Schemafragment gibt zwei Alternativen an, die anstelle des abstrakten Typs verwendet werden können:

```
<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>
```

Die Verarbeitung dieses XML-Fragments mit dem Assistenten generiert die folgenden COBOL-Sprachstrukturen:

```
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).
```

Die Sprachstrukturen werden in separaten Copybooks generiert. Die Sprachstruktur, die für option1 generiert wird, wird in einem Copybook generiert:

03 option1 PIC X(5).

Die Sprachstruktur für option2 wird in einem anderen Copybook generiert:

03 option2 PIC X(10).

Weitere Informationen zu abstrakten Datentypen finden Sie unter XML Schema Part 0: Primer Second Edition.

Informationen zur Verarbeitung variabel wiederkehrender Inhalte in COBOL

In COBOL können Sie variabel wiederkehrende Inhalte nicht mithilfe von Verweisarithmetik verarbeiten, um die einzelnen Dateninstanzen zu adressieren. Andere Programmiersprachen haben diese Einschränkungen nicht. Im folgenden Beispiel wird gezeigt, wie variabel wiederkehrende Inhalte in COBOL für eine Web-Service-Anwendung verarbeitet werden.

Dieses Verfahren gilt auch für die Umsetzung von XML in Anwendungsdaten unter Verwendung der **TRANSFORM**-API-Befehle. Das folgende WSDL-Beispieldokument stellt einen Web-Service mit Anwendungsdaten in Form einer 8-stelligen Zeichenfolge dar, die unregelmäßig wiederkehrt:

```
<?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>

```

Die Verarbeitung dieses WSDL-Dokuments durch DFHWS2LS generiert die folgenden COBOL-Sprachstrukturen:

```
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).
```

Beachten Sie, dass das acht Zeichen lange Feld component in einer separaten Struktur namens DFHWS-component definiert ist. Die Hauptdatenstruktur heißt applicationData und enthält zwei Felder: component-num und component-cont. Das Feld component-num gibt an, wie viele Instanzen der component-Daten vorhanden sind, und das Feld component-cont gibt den Namen eines Containers an, der die verkettete Liste von component-Feldern enthält.

Der folgende COBOL-Code stellt einen Weg zur Verarbeitung der Liste von variabel wiederkehrenden Daten dar. Er setzt ein Verknüpfungsabschnittsarray ein, um nachfolgende Instanzen der Daten zu adressieren, wobei jede Instanz mithilfe der Anweisung DISPLAY angezeigt wird:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EXVARY.
```

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
```

```
* Arbeitsspeichervariablen
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.
```

```
* umfangreiches Verknüpfungsabschnittsarray
01 BIG-ARRAY PIC X(659999).

* von DFHWS2LS erstellte Anwendungsdatenstrukturen
* wird üblicherweise mit einer COPY-Anweisung referenziert
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.
```

```
* Abrufen des Containers DFHWS-DATA
```



```

EXEC CICS GET CONTAINER('DFHWS-DATA')
SET(APP-DATA-PTR)
FLENGTH(APP-DATA-LENGTH)
END-EXEC
SET ADDRESS OF DFHWS2LS-data TO APP-DATA-PTR

* Abrufen der wiederkehrenden Komponentendaten
EXEC CICS GET CONTAINER(component-cont)
SET(COMPONENT-PTR)
FLENGTH(COMPONENT-DATA-LENGTH)
END-EXEC

* Verweisen der Komponentenstruktur auf die erste Instanz der Daten
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

* Speichern der Länge einer einzelnen Komponente
MOVE LENGTH OF DFHWS-component TO COMPONENT-LENGTH

* Verarbeiten aller Instanzen von Komponentendaten nacheinander
PERFORM WITH TEST AFTER
UNTIL COMPONENT-COUNT = component-num

* Anzeigen der aktuellen Instanz der Daten
DISPLAY 'component value is: ' component

* Adressieren der nächsten Instanz der Komponentendaten
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

* Beenden der Schleife
END-PERFORM.

* Verweisen der Komponentenstruktur zurück zur ersten Instanz der
* Daten, für alle weiteren Verarbeitungen, die durchgeführt werden sollen
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

* Zurückkehren zu CICS

EXEC CICS
RETURN
END-EXEC

GOBACK.

```

Der obige Code liefert eine generische Lösung für die Behandlung von variabel wiederkehrenden Inhalten. Das Array, BIG-ARRAY, bewegt sich der Reihe nach zum Anfang jeder einzelnen Komponente und verbleibt nicht fix am Anfang der Daten. Die Komponentendatenstruktur wird dann bewegt, um auf das erste Byte der nächsten Komponente zu verweisen. COMPONENT-PTR kann bei Bedarf verwendet werden, um die Anfangsposition der Komponentendaten wiederherzustellen.

Im Folgenden sehen Sie ein Beispiel einer SOAP-Nachricht, die dem WSDL-Dokument entspricht:

```

<?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>
    </applicationData>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

```

<component xmlns="">VALUE3</component>
</applicationData>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Dies ist die Ausgabe, die von dem COBOL-Programm erzeugt wird, wenn es die SOAP-Nachricht verarbeitet:

```

CPIH 20080115103151 component value is: VALUE1
CPIH 20080115103151 component value is: VALUE2
CPIH 20080115103151 component value is: VALUE3

```

Unterstützung für Werte variabler Länge und Leerzeichen

Sie können anpassen, wie Werte variabler Länge und Leerzeichen verarbeitet werden, indem Sie Einstellungen in den CICS-Assistenten nutzen und indem Sie Facetten direkt im XML-Schema hinzufügen.

Typischerweise ordnen der CICS-XML-Assistent und der CICS-Web-Service-Assistent Datenzeichenfolgen Zeichenarrays fester Länge zu. Diese Arrays müssen mit Leerzeichen oder Nullen gefüllt werden. Das Zuordnen von Werten variabler Länge zu Datenarrays fester Länge kann ineffizient sein und sinnlos Speicherplatz belegen. Wenn die Länge Ihrer Daten variabel ist, wird empfohlen, die Art und Weise anzupassen, in der diese Zuordnungen gehandhabt werden.

Bei einer WSDL-Dokumentkonvertierung aus einer Sprachstruktur in ein XML-Schema oder WSDL-Dokument wird empfohlen, die Facetten `whiteSpace` und `maxLength` in Ihrem XML-Schema anzupassen und den Parameter **CHAR-VARYING-LIMIT** in den Assistenten anzugeben.

Bei einer Konvertierung aus einem XML-Schema oder WSDL-Dokument in eine Sprachstruktur wird empfohlen, einen entsprechenden Wert für den Parameter **CHAR-VARYING** in den Assistenten festzulegen.

Anmerkung: Nullzeichen ('x00') sind in XML-Dokumenten nicht gültig. Alle Nullzeichen aus Anwendungsdaten, die von CICS geparkt werden, werden als Ende einer Zeichenfolge interpretiert und der Wert wird abgeschnitten. Wenn CICS Anwendungsdaten generiert, tut es dies entsprechend des Werts des Parameters **CHAR-VARYING**. Beispiel: Wenn die Option **CHAR-VARYING=NULL** angegeben ist, enden Zeichenfolgen variabler Länge, die von CICS generiert werden, auf ein Nullzeichen.

Werte variabler Länge aus XML zu Sprachstrukturen zuordnen

Verwenden Sie Facetten im XML-Schema oder geben Sie bestimmte Parameter in den CICS-Assistenten an, um anzupassen, wie Zuordnungen zwischen Ihrem XML-Schema oder WSDL-Dokument und der Sprachstruktur gehandhabt werden.

XML-Datentypen können mithilfe von Facetten eingeschränkt werden. Verwenden Sie die Längenfacetten (`length`, `maxLength` und `minLength`) und die Facette `whiteSpace`, um anzupassen, wie Daten variabler Länge in Ihrer XML gehandhabt werden.

length Gibt an, dass die Daten eine feste Länge haben.

maxLength

Gibt die maximale Länge für den Datentyp an. Wenn dieser Wert für einen zeichenfolgebasierten Datentyp nicht festgelegt wird, gibt es keine maximale Länge.

minLength

Gibt die minimale Länge für den Datentyp an. Wenn dieser Wert für einen zeichenfolgebasierten Datentyp nicht festgelegt wird, ist die minimale Länge '0'.

whiteSpace

Gibt an, wie Leerraum um einen Datenwert herum gehandhabt wird. Als Leerraum werden Leerzeichen, Tabulatoren und Zeilenumbrüche bezeichnet. Die Facette whiteSpace kann auf preserve, replace oder collapse festgelegt werden:

- Der Wert preserve behält alle Leerräume in dem Datenwert bei.
- Bei dem Wert replace werden alle Tabulatoren oder Zeilenumbrüche durch die entsprechende Anzahl von Leerzeichen ersetzt.
- Bei dem Wert collapse werden führende, nachfolgende und eingebettete Leerräume entfernt und alle Tabulatoren, Zeilenumbrüche und aufeinanderfolgenden Leerzeichen durch einzelne Leerzeichen ersetzt.

Wenn die Facette whiteSpace nicht festgelegt ist, wird der Leerraum beibehalten.

Weitere Informationen zu XML-Schemafacetten finden Sie unter XML Schema Part 2: Datatypes Second Edition.

Die folgenden Parameter in den CICS-Assistenten DFHSC2LS und DFHWS2LS können verwendet werden, um die Art und Weise zu ändern, in der Daten variabler Länge aus dem XML-Schema der Sprachstruktur zugeordnet werden. Diese Parameter sind auf Zuordnungsebene 1.2 oder höher verfügbar.

DEFAULT-CHAR-MAXLENGTH

Gibt die Standardarraylänge von Zeichendaten in Zeichen für Zuordnungen an, in deren XML-Schema oder WSDL-Dokument keine Länge eingeschlossen ist. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 1 bis 2.147.483.647 sein.

Es wird jedoch empfohlen, die maximale Zeichenlänge, die DFHSC2LS oder DFHWS2LS verwenden sollen, direkt in Ihrem XML-Schema oder WSDL-Dokument mithilfe der Facette maxLength anzugeben. Indem Sie die maximale Länge direkt in dem XML-Schema oder WSDL-Dokument angeben, vermeiden Sie Probleme, die auftreten können, wenn eine globale Standardeinstellung auf alle zeichenfolgebasierten Datentypen angewendet wird.

CHAR-VARYING-LIMIT

Gibt die maximale Größe von Zeichendaten variabler Länge an, die der Sprachstruktur zugeordnet werden. Wenn die Zeichendaten größer sind als der in diesem Parameter angegebene Wert, werden sie einem Container zugeordnet, und der Containername wird in der generierten Sprachstruktur verwendet. Der Wert kann zwischen 0 und dem Standardwert 32.767 Bytes liegen.

CHAR-VARYING

Gibt an, wie die Zeichendaten variabler Länge zugeordnet werden. Wenn Sie diesen Parameter nicht angeben, hängt die Standardzuordnung von der angegebenen Sprache ab. Sie haben folgende Optionen:

- **CHAR-VARYING=NO** gibt an, dass Zeichendaten variabler Länge als Zeichenfolgen mit fester Länge zugeordnet werden.
- **CHAR-VARYING=NULL** gibt an, dass Zeichendaten variabler Länge auf null endenden Zeichenfolgen zugeordnet werden.

- **CHAR-VARYING=YES** gibt an, dass Zeichendaten variabler Länge in PL/I einem CHAR VARYING-Datentyp zugeordnet werden. In den COBOL-, C- und C++-Sprachen werden Zeichendaten variabler Länge einer äquivalenten Darstellung zugeordnet, die zwei verwandte Elemente umfasst: die Datenlänge und die Daten.

CHAR-VARYING=YES führt in der Regel zu einer optimalen Leistung.

Werte variabler Länge aus Sprachstrukturen zu XML zuordnen

Sie können anpassen, wie Zuordnungen zwischen Ihrer Sprachstruktur und dem XML-Schema oder WSDL-Dokument gehandhabt werden. Legen Sie den Parameter **CHAR-VARYING** in DFHLS2SC oder DFHLS2WS auf COLLAPSE oder NULL fest, um zu ändern, wie diese Zeichenarrays generiert werden.

Das Festlegen der Option **CHAR-VARYING=NULL** weist CICS an, beim Generieren von XML ein Nullzeichen am Ende jedes Zeichenarrays hinzuzufügen.

Das Festlegen der Option **CHAR-VARYING=COLLAPSE** weist CICS an, beim Generieren von XML alle abschließenden Leerzeichen von Zeichenarrays automatisch zu entfernen. Diese Option ist nur auf Zuordnungsebene 2.1 oder höher verfügbar und **CHAR-VARYING=COLLAPSE** ist der Standardwert auf Zuordnungsebene 2.1 oder höher für alle Sprachen außer C und C++. Wenn die XML geparkt wird, werden alle führenden, abschließenden und eingebetteten Leerräume entfernt.

Unterstützung für UTF-16 in Anwendungsdaten

CICS-Web-Services unterstützen die Konvertierung von Anwendungsdaten, die in UTF-16 codiert sind, in XML oder JSON und umgekehrt. Verwenden Sie UTF-16, wenn Sie Daten in mehreren Sprachen speichern und verarbeiten müssen.

SOAP- und JSON-WebServices von CICS unterstützen die Konvertierung von Anwendungsdaten, die in UTF-16 codiert sind, in XML oder JSON und umgekehrt. Unicode ist ein Schema für Codeumsetzung mit variabler Breite, das es Systemen ermöglicht, Daten effizient zu verarbeiten.

UTF-16 ist eine Unicode-Codierung mit variabler Breite, bei der jedes Zeichen durch zwei oder vier Bytes dargestellt wird. CICS-Web-Services unterstützen CC-SID 1200 für Anwendungsdaten im Format UTF-16 BE (Big Endian) mit IBM Private Use Area. Dieses Verhalten ist konsistent mit der UTF-16-Unterstützung in allen unterstützten Sprachen.

UTF-16 wird auf Zuordnungsebene 4.0 und höher unterstützt. Sie können mithilfe von Zuordnungseinstellungen in den Assistenten anpassen, wie Anwendungsdaten konvertiert werden. Weitere Informationen zu XML-Zuordnungsebenen finden Sie unter Mapping levels for the CICS assistants. Weitere Informationen zu JSON-Zuordnungsebenen finden Sie unter Mapping levels for the CICS JSON assistants.

Anmerkung: UTF-16 erfordert mehr Verarbeitungszeit und ist weniger speichereffizient als EBCDIC-Codierungen. Hinzu kommt, dass gemischte Codierungstypen zu zusätzlicher Laufzeitverarbeitung führen.

UTF-16 aus einem XML- oder JSON-Schema Sprachstrukturen zuordnen

Die Unterstützung für UTF-16 hängt davon ab, wie Sie den Web-Service erstellen. Die Zuordnung von XML- oder JSON-Schemas zu Sprachstrukturen, auch bekannt als Top-down-Zuordnung, weist die folgenden Merkmale auf. Wenn UTF-16 akti-

viert ist, werden alle Textfelder UTF-16-Feldern zugeordnet, während numerische Anzeigedatentypen in COBOL als EBCDIC zugeordnet werden. Für die Verwendung von UTF-16 setzen Sie den CCSID-Parameter von DFHJS2LS, DFHSC2LS oder DFHWS2LS auf 1200.

Beispiel: Wenn das folgende XML-Schemafragment in der WSDL vorhanden ist:

```
<xsd:element name="myString" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="20"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Dann generiert der DFHWS2LS-Assistent das folgende Feld in einer COBOL-Sprachstruktur:

```
myString PIC N(
20
) USAGE NATIONAL
```

Mit dem Parameter CHAR-MULTIPLIER der Web-Service-Assistenten kann die Länge eines Felds angegeben werden, das die Assistenten generieren.

CHAR-MULTIPLIER

Wenn Sie UTF-16 verwenden, sind die einzig gültigen Werte für den Parameter **CHAR-MULTIPLIER** 2 oder 4 , wobei 2 der Standardwert ist.

CHAR-MULTIPLIER = 2 , wobei das Schema die Zeichenfolge `maxlength x` beschreibt, generiert `PIC N(x)` . Das Festlegen von **CHAR-MULTIPLIER** = 2 schließt nicht die Verwendung von Ersatzzeichenpaaren in einer UTF-16-Zeichenfolge aus, wirkt sich aber auf die Anzahl von Zeichen aus, die in das Feld passen.

CHAR-MULTIPLIER = 4 generiert `PIC N(2x)` . Wenn **CHAR-MULTIPLIER** = 4 , wird der Wert zur Laufzeit aufgefüllt, wenn die Zeichenfolge Zeichen umfasst, die in einer einzelnen Codierungseinheit ausgedrückt werden können.

UTF-16 aus Sprachstrukturen XML- oder JSON-Schemas zuordnen

Die Zuordnung aus einer Sprachstruktur zu einem XML- oder JSON-Schema, auch als Bottom-up-Zuordnung bezeichnet, wird auf andere Weise verwaltet als die Top-down-Zuordnung. Wenn eine UTF-16-Zeichenfolge in der Sprachstruktur deklariert wird, werden die Daten von CICS als UTF-16-codiert interpretiert, andernfalls wird angenommen, dass die Daten EBCDIC-codiert sind. Der CCSID-Parameter für DFHLS2JS, DFHLS2SC oder DFHLS2WS gibt die Codierung eines beliebigen EBCDIC-Texts innerhalb der Anwendungsdaten an. Er darf nicht UTF-16 angeben.

Die Datentypen, die als UTF-16-Zeichen interpretiert werden, sind folgende: `PIC N(n)` in COBOL, `WIDECHAR(n)` in PL/I und `char16_t[n]` in C und C++.

Der Parameter CHAR-USAGE des Web-Service-Assistenten kann verwendet werden, um Datentypen anzugeben.

CHAR-USAGE

In COBOL kann der nationale Datentyp, `PIC N`, für UTF-16- oder DBCS-Daten verwendet werden. Diese Einstellung wird durch die Compileroption `NSYMBOL` gesteuert. Sie müssen den Parameter **CHAR-USAGE** im Assistenten

auf denselben Wert setzen wie die Compileroption NSYMBOL, um sicherzustellen, dass die Daten richtig verarbeitet werden. Dieser Wert wird in der Regel auf CHAR-USAGE=NATIONAL gesetzt, wenn Sie UTF-16 verwenden.

Wenn Sie nationale Datentypen, die UTF-16- und DBCS-Daten in demselben Copybook enthalten, mischen möchten, können Sie die Qualifikationsmerkmale USAGE NATIONAL oder USAGE DISPLAY-1 in einzelnen Feldern verwenden.

Anmerkung: DFHLS2WS, DFHLS2SC und DFHLS2JS bieten keine Unterstützung für die COBOL-Klausel GROUP USAGE NATIONAL.

XML aus einer Anwendung abfragen

Sie können ein Anwendungsprogramm schreiben, um ein XML-Fragment abzufragen, bevor Sie es in Anwendungsdaten umsetzen.

About this task

Wenn Ihre Anwendung viele verschiedene Typen von XML verarbeiten soll, können Sie die XML abfragen, um zu bestimmen, welche XMLTRANSFORM-Ressource für die Umsetzung in Anwendungsdaten verwendet werden soll. Dieser Befehl kann auch nützlich sein, wenn Ihre XML-Datei <xsd:any>-Elemente enthält.

Procedure

1. Verwenden Sie in Ihrem Anwendungsprogramm den API-Befehl **TRANSFORM XMLTODATA**, um die XML abzufragen:

```
EXEC CICS TRANSFORM XMLTODATA  
CHANNEL('mein_kanalname')  
XMLCONTAINER('quellencontainername')  
ELEMNAME(elementname) ELEMNAMELEN(länge_des_elementnamens)
```

Sie müssen den Namen des Kanals und den Namen des Containers angeben, die die XML enthalten. Sie müssen keine XMLTRANSFORM-Ressource angeben, um die XML abzufragen. Das obige Beispiel fragt den Namen des ersten XML-Elements und die Länge des XML-Elements ab. Sie können auch den Typ des ersten XML-Elements, die Länge des Typs und den Namensbereich des Typs abfragen.

2. Optional: Wenn die Anwendung den Namensbereich des XML-Elements erfordert, stellen Sie einen Datenbereich bereit, in den CICS den ELEMNS-Wert schreiben kann.
3. Optional: Nach der Abfrage der XML können Sie Anwendungslogik schreiben, um zu bestimmen, welche XMLTRANSFORM-Ressource zum Umsetzen der XML in Anwendungsdaten verwendet werden soll.
4. Installieren Sie das Anwendungsprogramm in CICS.

Results

CICS liest den angegebenen Container und gibt die Informationen zum XML-Element an das Anwendungsprogramm zurück.

XML nach Datentyp verarbeiten

Wenn das XML-Schema globale Datentypen enthält, von denen mindestens einer in der XML referenziert ist, können Sie Metadaten zur Unterstützung dieser globalen Datentypen generieren und anschließend die XML parsen oder in ein Anwendungsprogramm umsetzen.

Before you begin

Um die richtigen Metadaten zu generieren, müssen Sie DFHSC2LS ausführen, wobei Sie den Parameter TYPES=ALL angeben.

About this task

Procedure

1. Um Anwendungsdaten nach Datentyp in XML umzusetzen, verwenden Sie den **TRANSFORM DATATOXML**-Befehl:

```
EXEC CICS TRANSFORM DATATOXML
XMLTRANSFORM('
mein_xml-umsetzungsname
')
CHANNEL('
mein_kanalname
')
DATCONTAINER('
quellencontainername
')
XMLCONTAINER('
zielcontainername
')
ELEMNAME(
elementname
) ELEMNAMELEN(
länge_des_elementnamens
)
ELEMNS(
elementnamensbereich
) ELEMNSLEN(
länge_des_elementnamensbereichs
)
TYPENAME(
typname
) TYPENAMELEN(
länge_des_typnamens
)
TYPENS(
typnamensbereich
) TYPENSLEN(
länge_des_typnamensbereichs
)
```

mein_xml-umsetzungsname ist der 32 Zeichen lange Name der XMLTRANSFORM-Ressource, die die XML-Bindung und das XML-Schema angibt. *mein_kanalname* ist der 16 Zeichen lange Name des Kanals, der die Eingabe- und Ausgabecontainer enthält. *quellencontainername* ist der 16 Zeichen lange Name des Eingabecontainers, der die Anwendungsdaten enthält. Und *zielcontainername* ist der 16 Zeichen lange Name des Ausgabecontainers, den CICS mit XML füllt. Sie müssen außerdem den XML-Elementnamen, den Namensbereich, den Datentyp und den Namensbereich des Datentyps für die Umsetzung angeben.

2. Um die XML nach Datentyp zu parsen, verwenden Sie den **TRANSFORM XMLTO-DATA**-Befehl. Welche Optionen Sie im Befehl angeben, hängt davon ab, ob die XML-Datei ein Attribut vom Typ 'xsi:type' aufweist.

- Wenn die XML das Attribut 'xsi:type' verwendet, geben Sie den folgenden Befehl in Ihrem Anwendungsprogramm an:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
mein_xml-umsetzungsname
')
CHANNEL('
```

```

mein_kanalname
')
XMLCONTAINER('
quellencontainername
')
DATCONTAINER('
zielcontainername
')
ELEMNAME(elementname) ELEMNAMELEN(länge_des_elementnamens)
TYPENAME(typname) TYPENAMELEN(länge_des_typnamens)

```

Wenn die Anwendung auch den Namensbereich des Datentyps erfordert, fügen Sie die Option TYPENS zum API-Befehl hinzu. CICS gibt den Datentyp aus dem Attribut 'xsi:type' in TYPENAME zurück.

- Wenn die XML das Attribut 'xsi:type' nicht verwendet, kann das Anwendungsprogramm den lokalen Namen und den Namensbereich des globalen Datentyps angeben. Verwenden Sie den folgenden Befehl in Ihrem Anwendungsprogramm:

```

EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
mein_xml-umsetzungsname
')
CHANNEL('
mein_kanalname
')
XMLCONTAINER('
quellencontainername
')
DATCONTAINER('
zielcontainername
')
TYPENAME(
typname
) TYPENAMELEN(
länge_des_typnamens
)
TYPENS(
typnamensbereich
) TYPENSLEN(
länge_des_typnamensbereichs
)

```

Das Anwendungsprogramm muss den Typnamen und den Typnamensbereich, aber nicht den Namen oder Namensbereich angeben. Dieser Befehl gibt an, dass die Anwendung CICS mitteilt, welcher Datentyp verwendet werden soll. Beim Generieren von XML fügt CICS immer das Attribut 'xsi:type' hinzu.

3. Installieren Sie das Anwendungsprogramm in CICS.

What to do next

Testen Sie, ob das Anwendungsprogramm die XML wie erwartet generiert und parst.

<xsd:any>-Datentypen verarbeiten

Wenn Sie mit einem XML-Schema arbeiten, das mindestens einen <xsd:any>-Datentyp enthält, können die XML-Assistenten den Datentyp einem Paar von CICS-Containern zuordnen. Sie können ein Anwendungsprogramm schreiben, um die XML in den Containern zu parsen.

Before you begin

Sie müssen das XML-Schema mit DFHSC2LS oder DFHWS2LS unter Verwendung der Zuordnungsebene 2.1 oder höher zuordnen.

About this task

Wenn CICS die Daten in XML umsetzt, platziert es die dem <xsd:any>-Datentyp zugeordnete XML in dem ersten Container und die gültigen Namensbereichsdeklarationen im zweiten Container.

Procedure

1. Verwenden Sie zum Parsen der XML-Daten den Befehl **TRANSFORM XMLTODATA** in Ihrem Anwendungsprogramm:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
mein_xml-umsetzungsname
')
CHANNEL('
mein_kanalname
')
XMLCONTAINER('
quellencontainername
')
DATCONTAINER('
zielcontainername
')
NSCONTAINER('
namensbereichscontainername
')
ELEMNAME(elementname) ELEMNAMELEN(länge_des_elementnamens)
```

mein_xml-umsetzungsname ist der 32 Zeichen lange Name der XMLTRANSFORM-Ressource, die die XML-Bindung und das Schema angibt. *mein_kanalname* ist der 16 Zeichen lange Name des Kanals mit den Eingabe- und Ausgabecontainern. *quellencontainername* ist der 16 Zeichen lange Name des Eingabecontainers mit der XML und *zielcontainername* ist der 16 Zeichen lange Name des Ausgabecontainers, den CICS mit Anwendungsdaten füllt. *namensbereichscontainername* ist der 16 Zeichen lange Name des Containers, den CICS mit den Namensbereichspräfixdeklarationen füllt. Stellen Sie Anfangswerte für die Optionen ELEMNAME und ELEMNAMELEN bereit. CICS gibt das XML-Element und seine Länge in den Optionen ELEMNAME und ELEMNAMELEN zurück.

2. Installieren Sie das Anwendungsprogramm in CICS.

What to do next

Testen Sie, dass das Anwendungsprogramm die XML korrekt parst.

XML-Umsetzungen überprüfen

Wenn Sie den CICS-XML-Assistenten verwenden, um Anwendungsdaten zu XML zuzuordnen, können Sie angeben, dass die Umsetzungen, die zur Laufzeit stattfinden, überprüft werden, um sicherzustellen, dass sie dem Schema folgen, das in der XML-Bindung enthalten ist. Sie können die Umsetzung von XML in Binärdaten oder von Binärdaten in XML überprüfen.

Before you begin

Während der Entwicklung und des Tests Ihrer CICS-Anwendung unterstützt eine vollständige Überprüfung die Erkennung von Problemen in der XML. Allerdings führt die vollständige Überprüfung der XML zu einer deutlich höheren Auslastung und es wird nicht empfohlen, XML in einer umfassend getesteten Produktionsanwendung zu überprüfen.

CICS verwendet ein Java-Programm, um die XML gegen ein Schema zu überprüfen. Deshalb muss Java-Unterstützung in Ihrer CICS-Region aktiviert sein.

Procedure

1. Richten Sie einen JVM-Server in der CICS-Region ein. Die XML-Validatorklasse kann in einem OSGi-Framework oder in Axis2 ausgeführt werden, aber nicht in einem Liberty-Profil. CICS stellt Beispiele für die schnelle Einrichtung eines JVM-Servers, der ein OSGi-Framework nutzt, bereit.
 - a. Installieren Sie den JVM-Beispielservers DFHJVMS in der Gruppe DFH\$OSGI oder erstellen Sie Ihren eigenen JVM-Server. Weitere Informationen finden Sie unter *Setting up a JVM server*.
 - b. Wenn Sie einen eigenen JVM-Server erstellt haben, ändern Sie die Programmdefinition DFHPIVAL in der Gruppe DFHPIVAL, um auf den Namen der JVMSERVER-Ressource zu verweisen. Die DFHPIVAL-Definition ist nicht gesperrt und kann bearbeitet werden. Standardmäßig referenziert die Definition DFHJVMS.
2. Stellen Sie sicher, dass sich die XML-Bindung und das Schema an derselben Position unter z/OS UNIX befinden. Die XMLTRANSFORM-Ressource definiert diese Dateien in CICS. Sie können den Befehl **INQUIRE XMLTRANSFORM** verwenden, um die Position der einzelnen Dateien zu überprüfen.
3. Aktivieren Sie die Überprüfung für die Anwendung. Öffnen Sie in CICS Explorer die XMLTRANSFORM-Ressource und bearbeiten Sie das Feld mit dem Validierungsstatus in der Liste von Attributen. Alternativ können Sie CEMT oder die SPI verwenden.

Results

Überprüfen Sie das Systemprotokoll, um zu sehen, ob die XML-Umsetzung gültig ist. Die Nachricht DFHML0508 gibt an, dass die XML erfolgreich überprüft wurde, und die Nachricht DFHML0507 gibt an, dass die Überprüfung fehlgeschlagen ist.

What to do next

Wenn Sie die XML-Überprüfung für die Anwendung nicht mehr benötigen, aktualisieren Sie die XMLTRANSFORM-Ressource, um sie zu inaktivieren.

Zuordnungen aus Sprachstrukturen generieren

Um XML aus Anwendungsdaten oder umgekehrt zu erstellen, müssen Sie die Zuordnungen erstellen, die beschreiben, wie CICS die Daten und den XML-Code während der Ausführung umsetzt. Sie können mit einem Anwendungsdatensatz anfangen, beispielsweise mit einem Kommunikationsbereich, einer VSAM-Datei, einer Warteschlange für temporären Speicher oder einem Db2-Datensatz.

Before you begin

Bevor Sie die Zuordnungen erstellen, müssen Sie sicherstellen, dass diese Vorbedingungen erfüllt sind:

- Sie müssen über eine Sprachstruktur verfügen, die den Anwendungsdatensatz in einer partitionierten Datei beschreibt. Die Sprachstruktur kann in allen höheren Programmiersprachen geschrieben werden, die vom CICS-XML-Assistenten unterstützt werden: COBOL, PL/I, C und C++. Wenn diese Zuordnung für einen Atom-Feed verwendet wird und Sie eines der Felder in Ihrem Anwendungsdatensatz verwenden, um Metadaten für die Atom-Einträge bereitzustellen (z. B. den Namen eines Autors), stellen Sie sicher, dass diese Felder nicht in Ihrer Sprachstruktur verschachtelt sind. Strukturen verschachtelter Felder können in einem Feld enthalten sein, das den Inhalt für einen Atom-Eintrag bereitstellt.
- Sie müssen die Benutzer-ID, unter der DFHLS2SC ausgeführt wird, für die Verwendung von z/OS UNIX konfigurieren.
- Die Benutzer-ID muss über Leseberechtigung verfügen, um auf die Sprachstruktur zugreifen zu können, und über Schreibberechtigung, um die Ausgabe in die entsprechenden Verzeichnisse unter z/OS UNIX zu stellen.
- Sie müssen der Benutzer-ID ausreichend Speicherplatz zuordnen, damit Java ausgeführt werden kann. Sie können alle unterstützten Versionen von Java verwenden. Standardmäßig verwendet DFHLS2SC die Java-Version, die im Parameter **JAVADIR** angegeben ist.

About this task

Verwenden Sie den CICS-XML-Assistenten, um die Datenzuordnungen für den Anwendungsdatensatz zu erstellen. Für jede höhere Programmiersprache, die von dem CICS-XML-Assistenten unterstützt wird, werden einige Datentypen nur eingeschränkt oder gar nicht unterstützt. Der CICS-XML-Assistent gibt Fehlermeldungen für alle nicht unterstützten Elemente aus, die er in Ihrer Sprachstruktur erkennt. Die Referenzinformationen für den CICS-XML-Assistenten listen die Einschränkungen auf, die für jede höhere Programmiersprache gelten.

Procedure

1. Führen Sie den DFHLS2SC-Stapeljob aus. DFHLS2SC hat optionale Parameter, die Sie auswählen, um Ihre Anforderungen zu erfüllen, z. B. die Auswahl einer bestimmten Codepage oder eines Namensbereichs. Verwenden Sie mindestens die folgenden Parameter:
 - a. Geben Sie die höhere Programmiersprache Ihrer Sprachstruktur im Parameter **LANG** an.
 - b. Wenn Sie die Datenzuordnungen in einem Bundle implementieren, geben Sie den Namen der Bundleressource im Parameter **BUNDLE** an. Wenn Sie eine XML-Bindung für einen Atom-Feed erstellen, geben Sie diesen Parameter nicht an.
 - c. Geben Sie die Zuordnungsebene im Parameter **MAPPING-LEVEL** an. Wenn Sie eine XML-Bindung für einen Atom-Feed erstellen, müssen Sie die Zuordnungsebene 3.0 oder höher verwenden. Verwenden Sie in anderen Fällen die aktuelle Zuordnungsebene, auch wenn Sie jede Zuordnungsebene verwenden können, um auf die erweiterten Zuordnungsoptionen zugreifen zu können.
 - d. Optional: Wenn Sie eine XML-Bindung für einen Atom-Feed erstellen und Ihr Anwendungsdatensatz Zeitmarken im CICS ABSTIME-Format enthält, geben Sie den optionalen Parameter **DATETIME=PACKED15** an, um diese Felder als Zeitmarken zuzuordnen.

- e. Geben Sie die Position und die Codepage der Sprachstrukturen an, die den Anwendungsdatensatz in den Parametern **PDSMEM** und **PDSCP** beschreiben.
- f. Geben Sie den Namen und die Position der Schemadatei im Parameter **SCHEMA** an. Die Dateierweiterung ist **.xsd**. Wenn Sie ein Bundle erstellen, geben Sie keine Position an. DFHLS2SC erstellt das XML-Schema, aber nicht die Verzeichnisstruktur, falls die Datei nicht bereits vorhanden ist.
- g. Geben Sie den Namen und die Position der XML-Bindung im Parameter **XSDBIND** an. Die Dateierweiterung ist **.xsdbind**. Wenn Sie ein Bundle erstellen, geben Sie keine Position an. DFHLS2SC erstellt die XML-Bindung, aber nicht die Verzeichnisstruktur, falls die Datei nicht bereits vorhanden ist.

Tip: Platzieren Sie die XML-Bindung und das Schema in derselben Verzeichnisstruktur, um die Validierung zu aktivieren. Eine Validierung kann nützlich sein, wenn Sie Ihre Anwendung in einer Entwicklungs- oder Testumgebung testen. Wenn Sie ein Bundle erstellen, platziert CICS die Dateien in demselben Verzeichnis.

Wenn Sie den Parameter **BUNDLE** angeben, erstellt der Stapeljob eine Bundleverzeichnisstruktur unter z/OS UNIX. Das Bundleverzeichnis hat ein Unterverzeichnis **META-INF**, das das Bundlemanifest enthält. Der Stapeljob erstellt außerdem ein XML-Schema und eine XML-Bindung in dem Bundleverzeichnis, wobei er die Dateinamen verwendet, die Sie für die Parameter **SCHEMA** und **XSDBIND** angegeben haben. Wenn Sie den Parameter **BUNDLE** nicht angeben, erstellt der Stapeljob das XML-Schema und die XML-Bindung nur an der angegebenen Position.

- 2. Installieren Sie die **BUNDLE**-Ressource oder eine **ATOMSERVICE**-Ressource, die diese XML-Bindung angibt. Die **BUNDLE**- und **ATOMSERVICE**-Ressourcen erstellen dynamisch eine **XMLTRANSFORM**-Ressource, die die Position des XML-Schemas und der Bindungsdatei definiert.

Results

Wenn Sie Zuordnungen aus Sprachstrukturen generieren, ist nur eine (1) XML-Umsetzung möglich.

Example

Das folgende Beispiel zeigt DFHLS2SC mit dem Minimalsatz von angegebenen Parametern an.

```
//LS2SC JOB 'abrechnungsdaten',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
/*
```

What to do next

Schreiben Sie ein Anwendungsprogramm, um die Anwendungsdaten in XML umzusetzen und umgekehrt. Sie können dieselben Zuordnungen für beide Umsetzun-

gen verwenden. Wenn Sie eine XML-Bindung für einen Atom-Feed erstellt haben, fahren Sie mit den Schritten zum Festlegen des Atom-Feeds fort.

Zuordnungen aus einem XML-Schema erstellen

Um Anwendungsdaten aus XML oder XML aus Anwendungsdaten zu erstellen, die einem vorhandenen XML-Schema entsprechen, erstellen Sie die Zuordnungen, die beschreiben, wie CICS die Daten zur Laufzeit umsetzen soll. Sie können mit einem XML-Schema oder einem WSDL-Dokument starten.

Before you begin

Sie müssen über ein gültiges XML-Schema oder ein gültiges WSDL-Dokument verfügen. Bevor Sie die Zuordnungen erstellen, müssen Sie sicherstellen, dass diese Vorbedingungen erfüllt sind:

- Sie müssen über ein gültiges XML-Schema oder ein gültiges WSDL-Dokument verfügen.
- Sie müssen die Benutzer-ID konfigurieren, unter der DFHSC2LS ausgeführt wird, um UNIX System Services zu verwenden.
- Die Benutzer-ID muss über Leseberechtigung verfügen, um auf das XML-Schema oder das WSDL-Dokument zugreifen zu können, und über Schreibberechtigung, um die Ausgabe in die entsprechenden Verzeichnisse unter z/OS UNIX zu stellen.
- Der Benutzer-ID muss ausreichend Speicher zugeordnet sein, um Java auszuführen. Sie können alle unterstützten Versionen von Java verwenden. Standardmäßig verwendet DFHWS2LS die Java-Version, die im Parameter **JAVADIR** angegeben ist.

About this task

Verwenden Sie den CICS XML-Assistenten, um die Datenzuordnungen für das XML-Schema zu erstellen.

Procedure

1. Führen Sie den DFHSC2LS-Stapeljob aus. DFHSC2LS hat optionale Parameter, die Sie auswählen können, um Ihre Anforderungen zu erfüllen, z. B. die Auswahl einer bestimmten Codepage oder die Angabe der Vorgehensweise zur Verarbeitung von Zeichendaten variabler Länge. Verwenden Sie mindestens die folgenden Parameter:
 - a. Geben Sie die Position Ihrer Eingabedatei im Parameter **WSDL** oder **SCHEMA** an. Sie können entweder ein WSDL-Dokument oder ein XML-Schema verwenden. Wenn Ihre Eingabedatei Referenzen auf andere Schemas oder Dokumente im Internet enthält und das System einen Proxy-Server verwendet, geben Sie den Domänennamen oder die IP-Adresse und die Portnummer des Proxy-Servers an.
 - b. Geben Sie die höhere Programmiersprache, die Sie generieren möchten, im Parameter **LANG** an. Der XML-Assistent unterstützt COBOL, C, C++ und PL/I.
 - c. Wenn Sie die Datenzuordnungen in einem Bundle implementieren, geben Sie den Namen und die Position des Bundles im Parameter **BUNDLE** an.

Der XML-Assistent erstellt eine Bibliothek der unterstützten Umsetzungen in der XML-Bindung. Für jedes globale Element in der Eingabedatei erstellt der Assistent eine eigene Umsetzung.

Wenn Sie den Parameter **BUNDLE** angeben, erstellt der Stapeljob eine Bundleverzeichnisstruktur unter z/OS UNIX. Das Bundleverzeichnis hat ein Unterverzeichnis META-INF, das das Bundlemanifest enthält. Der Stapeljob erstellt außerdem eine XML-Bindung im Bundleverzeichnis und platziert die Sprachstrukturen an der angegebenen Position. Der XML-Assistent stellt außerdem eine Kopie der Eingabedatei in das Bundleverzeichnis. Wenn Sie den Parameter **BUNDLE** nicht angeben, erstellt der Stapeljob die Sprachstrukturen und die XML-Bindung nur an der angegebenen Position.

2. Installieren Sie die BUNDLE-Ressource. Die BUNDLE-Ressource erstellt dynamisch eine XMLTRANSFORM-Ressource, die die Position des XML-Schemas oder des WSDL-Dokuments, der XML-Bindung und der Sprachstrukturen definiert.

Results

Wenn Sie Zuordnungen aus einem XML-Schema generieren, generiert CICS eine Sprachstruktur für jedes globale Element, das in dem Schema vorhanden ist.

Example

Das folgende Beispiel zeigt DFHSC2LS mit dem Mindestsatz von angegebenen Parametern an.

```
//SC2LS JOB 'abrechnungsdaten',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
/*
```

What to do next

Schreiben Sie ein Anwendungsprogramm, um die Anwendungsdaten in XML umzusetzen oder XML in Anwendungsdaten. Sie können dieselben Zuordnungen für beide Umsetzungen verwenden.

Anwendungsdaten in XML umsetzen

Sie können ein Anwendungsprogramm schreiben, um Anwendungsdaten in XML umsetzen.

Before you begin

Sie müssen über eine aktivierte XMLTRANSFORM-Ressource verfügen, die die XML-Bindung und das XML-Schema definiert.

About this task

Der XML-Assistent generiert die Zuordnungen in der XML-Bindung. Wenn Sie mit einer Sprachstruktur begonnen und DFHLS2SC verwendet haben, ist nur eine (1) Umsetzung in XML möglich. Wenn Sie mit einem XML-Schema begonnen haben,

sind viele Umsetzungen von XML in Sprachstrukturen möglich, deshalb muss Ihre Anwendung auswählen, welches XML-Element generiert werden soll.

Procedure

1. Das Anwendungsprogramm muss einen Kanal erstellen und die Daten, die der Sprachstruktur zugeordnet sind, in einen Bitmoduscontainer in diesem Kanal platzieren.
2. Verwenden Sie den API-Befehl **TRANSFORM DATATOXML**, um die Daten in XML umzusetzen:

```
EXEC CICS TRANSFORM DATATOXML
XMLTRANSFORM('
  mein_xml-umsetzungsname
')
CHANNEL('
  mein_kanalname
')
DATCONTAINER('
  quellencontainername
')
XMLCONTAINER('
  zielcontainername
')
```

Wenn die Ressource XMLTRANSFORM nur eine einzige Umsetzung aus der Sprachstruktur unterstützt, müssen Sie den Konvertierungstyp nicht in dem Befehl angeben. Wenn viele Umsetzungen möglich sind, fügen Sie Ihrem Anwendungsprogramm die folgenden Optionen hinzu:

```
ELEMNAME(elementname) ELEMNAMELEN(länge_des_elementnamens)
```

Diese zusätzlichen Optionen geben das XML-Element an, in das die Anwendungsdaten umgesetzt und im Ausgabecontainer platziert werden.

3. Installieren Sie das Anwendungsprogramm.

Results

Wenn die Anwendung den Befehl **TRANSFORM DATATOXML** ausführt, prüft CICS die XMLTRANSFORM-Ressource, um die Zuordnungen in der XML-Bindung zu finden, und setzt die Anwendungsbinärdaten unter Verwendung der Container im Kanal in XML um. Die XML wird in dem Container platziert, den die Anwendung in der Option XMLCONTAINER angegeben hat. Die XML entspricht dem XML-Schema, das in der Ressource XMLTRANSFORM definiert ist.

What to do next

Sie können dieselben Zuordnungen auch verwenden, um XML in Anwendungsdaten umzusetzen. Details finden Sie unter „XML in Anwendungsdaten umsetzen“.

XML in Anwendungsdaten umsetzen

Sie können ein Anwendungsprogramm schreiben, um XML in Anwendungsdaten umsetzen. Sie können die XML auch abfragen, bevor sie umgesetzt wird.

Before you begin

Sie müssen über eine aktivierte XMLTRANSFORM-Ressource verfügen, die die XML-Bindung und das XML-Schema definiert.

About this task

Der CICS-XML-Assistent generiert die Zuordnungen in der XML-Bindung. Wenn Sie mit einer Sprachstruktur begonnen und DFHLS2SC verwendet haben, ist nur eine (1) Umsetzung aus XML möglich. Wenn Sie mit einem XML-Schema begonnen haben, kann es zu vielen Umsetzungen von XML in Sprachstrukturen kommen, deshalb muss Ihre Anwendung auswählen, welche Sprachstruktur als Ausgabe verwendet werden soll.

Procedure

1. Erstellen Sie einen Kanal und platzieren Sie die XML in einem Textmoduscontainer in diesem Kanal. Wenn die Anwendung die XML in einem BIT-Modus-Container platziert, versucht CICS, die Codierung der Textdaten zu bestimmen, aber es kann länger dauern, den Container zu verarbeiten, und die Codierung ist unter Umständen nicht korrekt.
2. Verwenden Sie den API-Befehl **TRANSFORM XMLTODATA**. Wenn nur eine Umsetzung für die XML möglich ist, können Sie den folgenden Befehl verwenden:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
  mein_xml-umsetzungsname
')
CHANNEL('
  mein_kanalname
')
XMLCONTAINER('
  quellencontainername
')
DATCONTAINER('
  zielcontainername
')
```

Wenn mehr als eine Umsetzung möglich ist, verwenden Sie die folgenden zusätzlichen Optionen:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
  mein_xml-umsetzungsname
')
CHANNEL('
  mein_kanalname
')
XMLCONTAINER('
  quellencontainername
')
DATCONTAINER('
  zielcontainername
')
ELEMNAME(
  elementname
) ELEMNAMELEN(
  länge_des_elementnamens
)
```

Im zweiten Beispiel gibt CICS den Namen des Elements zurück, das in der Option ELEMNAME gefunden wurde. Anschließend kann die Anwendung den Namen des Elements verwenden, um zu bestimmen, welche Sprachstrukturen zur Interpretation der Inhalte des Zielcontainers verwendet werden sollen.

3. Installieren Sie das Anwendungsprogramm.

Results

Wenn die Anwendung den Befehl **TRANSFORM XMLTODATA** ausführt, verwendet CICS die Details in der XMLTRANSFORM-Ressource zur Umsetzung der XML in Anwendungsbinärdaten mithilfe der Container im Kanal.

Example

What to do next

Sie können dieselben Zuordnungen auch verwenden, um Anwendungsdaten in XML umzusetzen. Details finden Sie unter „Anwendungsdaten in XML umsetzen“ auf Seite 430.

Anwendungsdaten und JSON zuordnen und umsetzen

Sie können Anwendungsprogramme schreiben, um Binärdaten von Anwendungen in JavaScript Object Notation (JSON) umzusetzen und umgekehrt. CICS unterstützt eine Reihe höherer Programmiersprachen und stellt einen JSON-Assistenten für die Zuordnung der Daten zur Umsetzung während der Verarbeitung bereit. CICS verwendet im Rahmen der Web-Service-Unterstützung dieselbe Technologie zum Zuordnen der Anwendungsdaten zu JSON-Nachrichten.

Before you begin

Für die Ausführung des JSON-Assistenten muss Java installiert sein. Umsetzungen können entweder intern mit CICS oder mithilfe eines JVM-Servers ausgeführt werden. Wenn Sie Java für die Umsetzungen verwenden, müssen Sie einen Axis2-JVM-Server installiert haben, um die Anwendungsdaten und JSON umsetzen zu können. Weitere Informationen finden Sie unter *Configuring a JVM server for Axis2*.

About this task

Der Vorteil dieser Vorgehensweise zur Umsetzung von Anwendungsdaten in und aus JSON besteht darin, dass CICS mehr kann als ein JSON-Parser. CICS kann die JSON interpretieren und datensatzbasierte Konvertierungen der Anwendungsdaten durchführen. Deshalb können Sie mit dieser Methode einfacher und schneller Anwendungen erstellen, die mit JSON arbeiten.

Der CICS JSON-Assistent ist ein bereitgestelltes Dienstprogramm, das Sie beim Erstellen der erforderlichen Zuordnungsartefakte zum Umsetzen von Anwendungsbinärdaten in JSON oder von JSON in Anwendungsbinärdaten unterstützt. Der JSON-Assistent erstellt die Artefakte in einem Bundleverzeichnis.

Procedure

1. Erstellen Sie ein Bundle mithilfe des JSON-Assistenten. Dieses Bundle enthält die erforderlichen Zuordnungsartefakte für Datenumsetzungen.
2. Installieren Sie das Bundle in CICS, um die Zuordnungen verfügbar zu machen.
3. Erstellen oder aktualisieren Sie ein Anwendungsprogramm zur Verarbeitung von Datenumsetzungen. Sie haben zwei Optionen:
 - Verwenden Sie die API-Befehle **TRANSFORM DATATOJSON** und **TRANSFORM JSONTODATA** im Anwendungsprogramm. Dies ist die empfohlene Vorgehensweise.

- Verwenden Sie den API-Befehl **LINK PROGRAM**, um die von CICS bereitgestellte verknüpfbare Schnittstelle DFHJSON zu verknüpfen.

Die Anwendung muss eine kanalbasierte Schnittstelle verwenden.

4. Führen Sie die Anwendung aus, um zu testen, ob die Umsetzung wie gewünscht funktioniert.

Results

Ihre Anwendungsdaten werden in JSON umgesetzt oder Ihre JSON-Daten in Anwendungsdaten.

What to do next

Weitere Informationen zu den Schritten 1 auf Seite 433 bis 4 finden Sie in den folgenden Themen.

CICS JSON-Assistent

Der CICS JSON-Assistent ist ein Satz Stapeldienstprogramme, die Sie beim Generieren von Zuordnungen zwischen Strukturen einer höheren Programmiersprache und JSON-Schemas unterstützen, um Umsetzungen zwischen JSON und Anwendungsdaten durchzuführen. Der Assistent unterstützt die schnelle Implementierung von Anwendungen, die JSON mit minimalem Programmieraufwand verarbeiten.

Indem Sie den JSON-Assistenten für CICS verwenden, reduzieren Sie die Menge an Code, den Sie schreiben müssen, um JSON zu parsen oder zu erstellen; CICS setzt Daten zwischen JSON-Fragmenten und der Datenstruktur eines Anwendungsprogramms um.

Der JSON-Assistent kann ein JSON-Schema aus einer einfachen Sprachstruktur bzw. eine Sprachstruktur aus einem vorhandenen XML-Schema erstellen und er unterstützt COBOL, C/C++ und PL/I. Er generiert außerdem Metadaten, die CICS zur Laufzeit verwendet, um XML-Daten automatisch in binäre Anwendungsdaten zu konvertieren (oder umgekehrt). Die Metadaten sind in einer XML-Bindung definiert und unter z/OS UNIX gespeichert. Das Schema für die XML-Bindung befindet sich im Verzeichnis `/usr/lpp/cicsts/cicsts52/schemas/xmltransform/` unter z/OS UNIX.

Die Dienstprogramme DFHJS2LS und DFHLS2JS werden gesammelt als CICS JSON-Assistent bezeichnet:

DFHLS2JS

DFHLS2JS ordnet Strukturen höherer Programmiersprachen JSON-Schemas zu.

DFHJS2LS

DFHJS2LS ordnet JSON-Schemas Strukturen höherer Programmiersprachen zu.

Die JCL-Prozeduren zum Ausführen beider Programme befinden sich in der Bibliothek `hlq.XDFHINST`, wobei `hlq` das übergeordnete Qualifikationsmerkmal Ihrer CICS-Installation ist.

Der relevante Verwendungsmodus für die Prozedur DFHLS2JS oder DFHJS2LS hängt von Ihren Anforderungen ab:

- DFHLS2JS: Konvertierung einer höheren Programmiersprache in ein JSON-Schema für eine verknüpfbare Schnittstelle
- DFHJS2LS: Konvertierung eines JSON-Schemas in eine höhere Programmiersprache für eine verknüpfbare Schnittstelle
- DFHLS2JS: Konvertierung einer höheren Programmiersprache in ein JSON-Schema für Anforderung/ Antwort-Services
- DFHJS2LS: Konvertierung eines JSON-Schemas in eine höhere Programmiersprache für Anforderung/ Antwort-Services
- DFHJS2LS: Konvertierung eines JSON-Schemas in eine höhere Programmiersprache für REST-konforme Services

Die beiden Zuordnungen sind nicht symmetrisch:

- Wenn Sie eine Sprachdatenstruktur mit DFHLS2JS verarbeiten und anschließend das resultierende JSON-Schema mit DFHJS2LS verarbeiten, stimmt die finale Datenstruktur wahrscheinlich nicht mit dem Original überein.
- Wenn Sie ein JSON-Schema mit DFHJS2LS verarbeiten und anschließend die resultierende Sprachstruktur mit DFHLS2JS verarbeiten, stimmt das finale JSON-Schema wahrscheinlich nicht mit dem Original überein.
- In manchen Fällen generiert DFHJS2LS Sprachstrukturen, die von DFHLS2JS nicht unterstützt werden.

Sie müssen Strukturen höherer Programmiersprachen, die von DFHLS2JS verarbeitet werden, entsprechend den Regeln der Sprache codieren, die in den von CICS unterstützten Sprachcompilern implementiert sind.

DFHLS2JS: Konvertierung einer höheren Programmiersprache in ein JSON-Schema für eine verknüpfbare Schnittstelle

Die katalogisierte Prozedur DFHLS2JS generiert aus einer Struktur einer höheren Programmiersprache ein JSON-Schema und eine JSON-Bindungsdatei. Verwenden Sie DFHLS2JS, wenn Sie ein CICS-Programm erstellen wollen, das JSON parsen oder erstellen kann.

Die JCL-Prozedur DFHLS2JS wird in der Datei *HLQ.XDFHINST* installiert, wobei *HLQ* das übergeordnete Qualifikationsmerkmal für die Installationsposition von CICS ist.

Jobsteueranweisungen für DFHLS2JS

JOB Startet den Job.

EXEC Gibt den Prozedurnamen an (DFHLS2JS).

INPUT.SYSUT1 DD

Gibt die Eingabe an. Die Eingabeparameter werden im Eingabedatenstrom angegeben. Sie können jedoch auch in einer Datei oder in einem Member einer partitionierten Datei definiert werden.

Symbolische Parameter

Die folgenden symbolischen Parameter sind in DFHLS2JS definiert:

JAVADIR = *pfad*

Gibt den Namen des Java-Verzeichnisses an, das von DFHLS2JS verwendet wird. Der Wert dieses Parameters wird an */usr/lpp/* angehängt, wodurch sich der vollständige Pfadname */usr/lpp/pfad* ergibt.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **JAVADIR** angegeben wurde.

PATHPREF = *präfix*

Gibt ein optionales Präfix an, das den z/OS UNIX-Verzeichnispfad erweitert, der in anderen Parametern verwendet wird. Der Standardwert ist eine leere Zeichenfolge.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **PATHPREF** angegeben wurde.

SERVICE = *wert*

Verwenden Sie diesen Parameter nur, wenn sie vom IBM Support dazu aufgefordert werden.

TMPDIR = *tmp-verzeichnis*

Gibt die Position eines Verzeichnisses in z/OS UNIX an, das DFHLS2JS als temporären Arbeitsbereich verwendet. Die Benutzer-ID, unter der der Job ausgeführt wird, muss über Lese- und Schreibberechtigungen für dieses Verzeichnis verfügen.

Der Standardwert ist /tmp.

TMPPFILE = *tmp-präfix*

Gibt ein Präfix an, das DFHLS2JS verwendet, um die Namen der temporären Arbeitsbereichsdateien zu erstellen.

Der Standardwert ist LS2JS.

USSDIR = *pfad*

Gibt den Namen des CICS TS-Verzeichnisses im z/OS UNIX-Dateisystem an. Der Wert dieses Parameters wird an /usr/lpp/cicsts/ angehängt, wodurch sich der vollständige Pfadname /usr/lpp/cicsts/*pfad* ergibt. Er muss als '.' (Punkt) angegeben werden, wenn der Standardwert verwendet wird.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **USSDIR** angegeben wurde.

Der temporäre Arbeitsbereich

DFHLS2JS erstellt die folgenden drei temporären Dateien zur Laufzeit:

tmp-verzeichnis/tmp-präfix.in

tmp-verzeichnis/tmp-präfix.out

tmp-verzeichnis/tmp-präfix.err

Dabei gilt Folgendes:

tmp-verzeichnis ist der Wert, der im Parameter **TMPDIR** angegeben ist.

tmp-präfix ist der Wert, der im Parameter **TMPPFILE** angegeben ist.

Die Standardnamen für die Dateien lauten wie folgt (wenn **TMPDIR** und **TMPPFILE** nicht angegeben sind):

/tmp/LS2JS.in

/tmp/LS2JS.out

/tmp/LS2JS.err

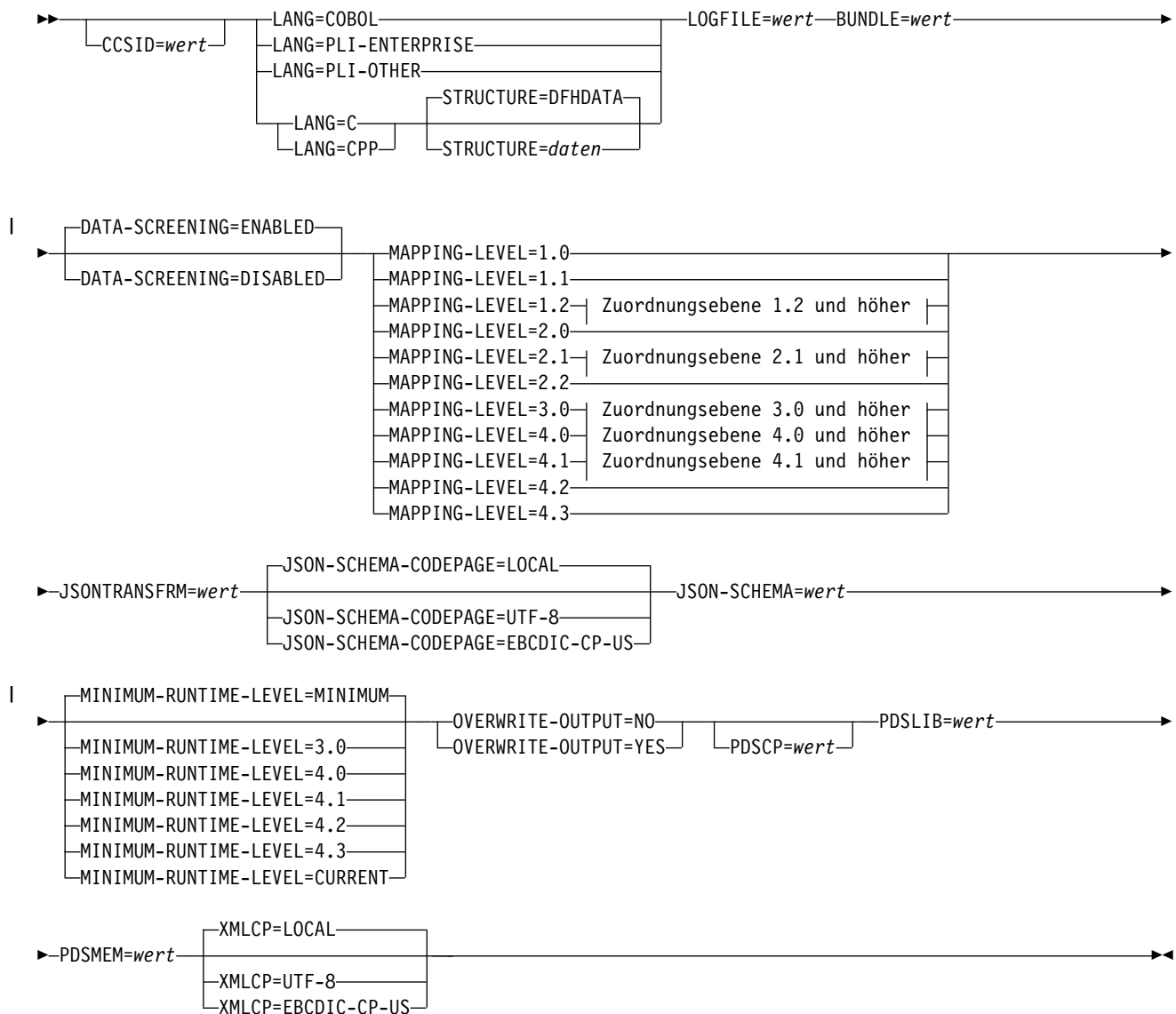
Important: DFHLS2JS sperrt nicht den Zugriff auf die z/OS UNIX-Dateien oder -Dateimember. Wenn folglich zwei oder mehr Instanzen von DFHLS2JS gleichzeitig ausgeführt werden und dieselben temporären Arbeitsbereichsdateien verwenden, kann nicht verhindert werden, dass ein Job die Arbeitsbereichsdateien überschreibt, während ein anderer Job sie verwendet, was zu unvorhersehbaren Fehlern führt.

Arbeiten Sie eine Namenskonvention und Betriebsprozeduren aus, die diese Situation verhindern. Sie können beispielsweise den symbolischen Systemparameter **SYSUID** verwenden, um Arbeitsbereichsdateinamen zu generieren, die für einen einzelnen Benutzer eindeutig sind.

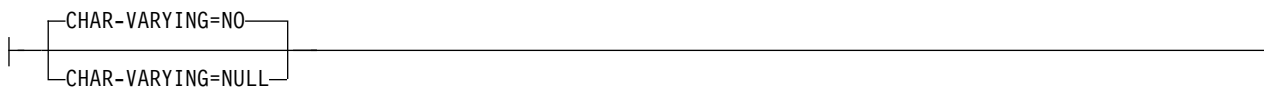
Diese temporären Dateien werden gelöscht, bevor der Job beendet wird.

Eingabeparameter für DFHLS2JS

Das folgende Syntaxdiagramm zeigt die verfügbaren Eingabeparameter:



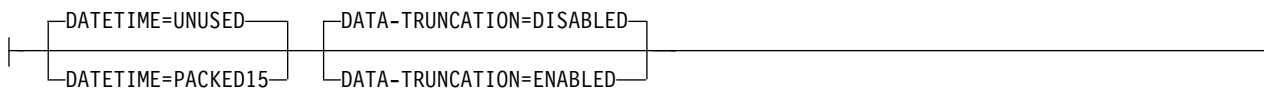
Zuordnungsebene 1.2 und höher:



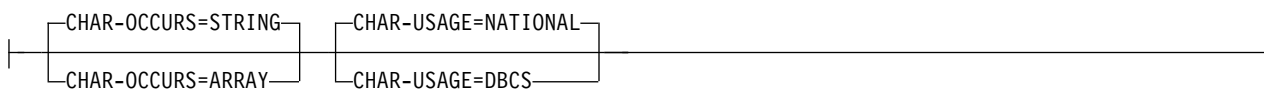
Zuordnungsebene 2.1 und höher:



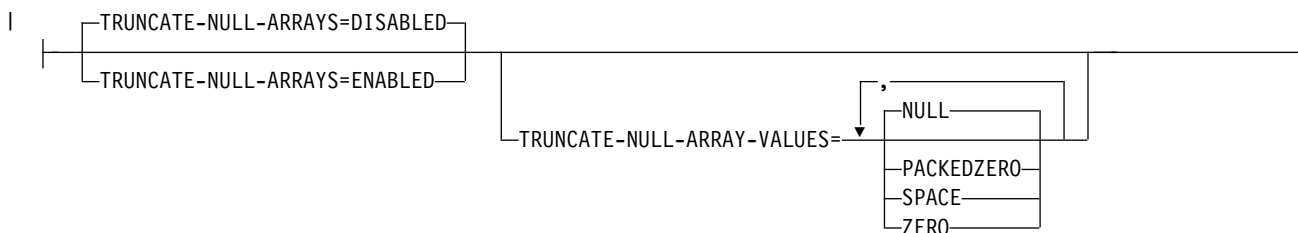
Zuordnungsebene 3.0 und höher:



Zuordnungsebene 4.0 und höher:



Zuordnungsebene 4.1 und höher:



Parameterverwendung

Parameter müssen die folgenden Regeln befolgen:

- Sie können die Eingabeparameter in beliebiger Reihenfolge angeben.
- Jeder Parameter muss in einer neuen Zeile eingegeben werden.
- Ein Parameter (und ggf. das Fortsetzungszeichen) darf nicht über Spalte 72 hinausgehen; die Spalten 73-80 müssen Leerzeichen enthalten.
- Wenn ein Parameter zu lang ist, um in eine einzelne Zeile zu passen, verwenden Sie am Ende der Zeile einen Stern (*), um anzugeben, dass der Parameter in der nächsten Zeile fortgesetzt wird. Alles (einschließlich der Leerzeichen) vor dem Stern wird als Teil des Parameters betrachtet.
- Ein Nummernzeichen (#) an der ersten Zeichenposition der Zeile ist ein Kommentarzeichen. Die Zeile wird ignoriert.

- Ein Komma an der letzten Zeichenposition der Zeile ist ein optionales Zeilentrennzeichen und wird ignoriert.

Parameterbeschreibungen

BUNDLE = *wert*

Gibt den Pfad und den Namen des Bundleverzeichnis unter z/OS UNIX an. Wenn Sie diesen Wert angeben, generiert der JSON-Assistent ein Bundle, das die JSON-Bindung enthält. Die Pfadinformationen für diesen Parameter überschreiben alle Pfadinformationen für den Parameter **JSONTRANSFRM**.

Sie können optional eine Archivdatei statt eines Verzeichnisnamens angeben. Der JSON-Assistent unterstützt .zip- und .jar-Archive. Sie müssen jedoch das Archiv dekomprimieren, bevor Sie versuchen, die BUNDLE-Ressource zu installieren.

CCSID = *wert*

Gibt die CCSID an, die zur Laufzeit verwendet wird, um Zeichendaten in der Anwendungsdatenstruktur zu codieren. Der Wert dieses Parameters überschreibt den Wert des Systeminitialisierungsparameters **LOCALCCSID**. *wert* muss eine EBCDIC-CCSID sein, die von Java- und z/OS-Konvertierungsservices unterstützt wird. Wenn Sie diesen Parameter nicht angeben, wird die Anwendungsdatenstruktur unter Verwendung der im Systeminitialisierungsparameter angegebenen CCSID codiert.

Sie können diesen Parameter mit jeder Zuordnungsebene verwenden.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Gibt an, wie Zeichenfelder in der Sprachstruktur auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Ein Zeichenfeld in COBOL ist eine Picture-Klausel vom Typ X, z. B. PIC(X) 10. Ein Zeichenfeld in C/C++ ist ein Zeichenarray. Dieser Parameter gilt nicht für Enterprise PL/I und andere PL/I-Sprachstrukturen. Sie haben folgende Optionen:

NO Zeichenfelder werden einer JSON-Zeichenfolge zugeordnet und als Felder mit fester Länge verarbeitet. Die maximale Länge der Daten ist gleich der Länge des Felds. NO ist der Standardwert für den Parameter **CHAR-VARYING** für COBOL und PL/I auf den Zuordnungsebenen 2.0 und niedriger.

NULL Zeichenfelder werden einer JSON-Zeichenfolge zugeordnet und als auf null endende Zeichenfolgen verarbeitet. CICS fügt bei der Umsetzung aus einem JSON-Schema ein abschließendes Nullzeichen (NULL) hinzu. Die maximale Länge der Zeichenfolge wird als ein Zeichen weniger als die in der Sprachstruktur angegebene Länge berechnet. NULL ist der Standardwert für den Parameter **CHAR-VARYING** für C/C++.

COLLAPSE

Zeichenfelder werden einer JSON-Zeichenfolge zugeordnet. Abschließende und eingebettete Leerzeichen in dem Feld werden nicht in die JSON-Nachricht eingeschlossen, z. B. wird
 <leerzeichen>AB<leerzeichen><leerzeichen><leerzeichen>C<leerzeichen>
 zu AB<leerzeichen>C. COLLAPSE ist der Standardwert für den Parameter **CHAR-VARYING** für COBOL und PL/I ab Zuordnungsebene 2.1.

BINARY

Zeichenfelder werden einer JSON-Zeichenfolge zugeordnet, die Base64-codierte Daten enthält, und als Felder mit fester Länge verarbeitet. Der BINARY-Wert im Parameter **CHAR-VARYING** ist nur ab Zuordnungsebene 2.1 verfügbar.

CHAR-OCCURS = { **STRING** | **ARRAY** }

Gibt an, wie Zeichenarrays in der Sprachstruktur auf Zuordnungsebene 4.0 oder höher zugeordnet werden. Beispiel: PIC X OCCURS 20. Dieser Parameter kann nur in der COBOL-Sprache verwendet werden.

ARRAY

Zeichenarrays werden einem JSON-Array zugeordnet. Das heißt, dass jedes Zeichen als einzelnes JSON-Element zugeordnet wird. Dies ist auch das Verhalten auf Zuordnungsebene 3.0 und niedriger.

STRING

Zeichenarrays werden einer JSON-Zeichenfolge zugeordnet. Das heißt, dass das gesamte COBOL-Array als einzelnes JSON-Element zugeordnet wird.

CHAR-USAGE = { **NATIONAL** | **DBCS** }

In COBOL kann der nationale Datentyp, PIC N, für UTF-16- oder DBCS-Daten verwendet werden. Diese Einstellung wird durch die Compileroption NSYMBOL gesteuert. Sie müssen den Parameter **CHAR-USAGE** im Assistenten auf denselben Wert setzen wie die Compileroption NSYMBOL, um sicherzustellen, dass die Daten richtig verarbeitet werden. Dieser Wert wird in der Regel auf CHAR-USAGE=NATIONAL gesetzt, wenn Sie UTF-16 verwenden.

DBCS Daten aus PIC(*n*)-Feldern werden als UTF-16-codierte Daten behandelt.

NATIONAL

Daten aus PIC(*n*)-Feldern werden als DBCS-codierte Daten behandelt.

DATA-SCREENING = { **ENABLED** | **DISABLED** }

Gibt an, ob die von Anwendungen bereitgestellten Daten auf Fehler überprüft werden.

ENABLED

Alle von Anwendungen bereitgestellten Laufzeitdaten, die nicht mit der Sprachstruktur konsistent sind, werden als Fehler behandelt, und es wird die Nachricht DFHPI1010 ausgegeben. Eine Fehlerantwort wird an die Anwendung zurückgegeben.

DISABLED

Werte in den von Anwendungen bereitgestellten Laufzeitdaten, die nicht konsistent mit der Sprachstruktur sind, werden durch Standardwerte ersetzt. Eine Null ersetzt beispielsweise einen fehlerhaften Wert in einem numerischen Feld. Die Nachricht DFHPI1010 wird nicht ausgegeben und es wird eine normale Antwort an die Anwendung zurückgegeben. Dieses Feature kann verwendet werden, um die Fehlerantworten INVALID_PACKED_DEC und INVALID_ZONED_DEC zu vermeiden, die von nicht initialisierten Ausgabefeldern generiert werden.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Gibt an, ob Daten mit variabler Länge in einer Feldstruktur mit fester Länge toleriert werden:

DISABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, weist CICS die abgeschnittenen Daten zurück und gibt eine Fehlermeldung aus.

ENABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, toleriert CICS die abgeschnittenen Daten und verarbeitet die fehlenden Daten als Nullwerte.

DATETIME = { **UNUSED** | **PACKED15** }

Gibt an, ob JSON-Datum/Zeit-Eigenschaften in der Struktur einer höheren Programmiersprache, einschließlich CICS ABSTIME-Werte, als Zeitmarken zugeordnet werden:

PACKED15

Alle JSON-Datum/Zeit-Eigenschaften werden als Zeitmarken zugeordnet.

UNUSED

Keine JSON-Datum/Zeit-Eigenschaften werden als Zeitmarken zugeordnet. Dies ist die Standardzuordnung.

Sie können diesen Parameter auf der Zuordnungsebene 3.0 festlegen.

JSON-SCHEMA = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die das JSON-Schema geschrieben wird.

JSON-SCHEMA-CODEPAGE = { **LOCAL** | **UTF-8** | **EBCDIC-CP-US** }

Gibt die Codepage an, die zum Generieren der JSON-Schemadokumente verwendet wird.

LOCAL

Gibt an, dass die JSON-Schemas unter Verwendung der Standardcodepage für das Dateisystem generiert werden.

UTF-8 Gibt an, dass die JSON-Schemas unter Verwendung der UTF-8-Codepage generiert werden.

EBCDIC-CP-US

Gibt an, dass die JSON-Schemas unter Verwendung der US EBCDIC-Codepage generiert werden.

JSONTRANSFRM = *wert*

Dieser Parameter ist für den LINKable-Modus obligatorisch, aber ungültig für den Anforderung/Antwort-Modus. Er gibt den Namen an, der für die **JSON-TRANSFRM**-Bundleressource in CICS verwendet wird.

LANG = **COBOL**|**PLI-ENTERPRISE**|**PLI-OTHER**|**C**|**CPP**

Gibt die Datenstruktur der höheren Programmiersprache an:

COBOL

COBOL

PLI-ENTERPRISE

Enterprise PL/I

PLI-OTHER

Andere PL/I-Ebene als Enterprise PL/I

C C

CPP C++

LOGFILE = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die DFHLS2JS das

Aktivitätenprotokoll und die Traceinformationen schreibt. Bei Bedarf erstellt DFHLS2JS die Datei, aber nicht die Verzeichnisstruktur.

Unter Umständen werden Sie von der IBM Serviceorganisation aufgefordert, diesen Parameter zu verwenden, wenn Sie Probleme mit DFHLS2JS feststellen.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 }

Gibt die Zuordnungsebene für den Assistenten an, der beim Generieren der JSON-Bindung und der Sprachstrukturen verwendet werden soll. Sie müssen Zuordnungsebene 3.0 oder höher verwenden.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 | **CURRENT** }

Gibt die älteste CICS-Laufzeitumgebung an, in der die JSON-Bindung implementiert werden kann. Wenn Sie eine Version auswählen, die nicht mit den anderen Parametern übereinstimmt, die Sie angegeben haben, erhalten Sie eine Fehlermeldung. Sie können die folgenden Optionen auswählen:

MINIMUM

Die niedrigste mögliche Laufzeitebene von CICS wird anhand der von Ihnen ausgewählten Parameter automatisch zugewiesen.

3.0 Geben Sie die Laufzeitebene 3.0 oder höher an, wenn Sie den CICS JSON-Assistenten verwenden und von fortschrittlichen Datenzuordnungen profitieren möchten.

4.0 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.0 oder niedriger verwenden. Sie können alle optionalen Parameter auf dieser Ebene verwenden.

4.1 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.1 oder niedriger verwenden.

4.2 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS V5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.2 oder niedriger verwenden.

4.3 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.3 oder niedriger verwenden.

CURRENT

Verwenden Sie diese Laufzeitebene, um die generierte Bindungsdatei in einer CICS-Region zu implementieren, die dieselbe Laufzeitumgebung wie die Region hat, die zum Generieren der Bindungsdatei verwendet wurde.

OVERWRITE-OUTPUT = { NO | YES }

Steuert, ob vorhandene CICS-BUNDLES im Dateisystem überschrieben werden können.

NO Vorhandene BUNDLES werden nicht ersetzt. Wenn ein vorhandenes BUNDLE gefunden wird, gibt DFHLS2JS die Fehlermeldung DFHPI9689E aus und wird beendet.

YES Vorhandene BUNDLES werden ersetzt. Wenn ein vorhandenes BUND-

LE gefunden wird, wird die Nachricht DFHPI9683W ausgegeben, um Sie darüber zu informieren, dass die Datei ersetzt wurde.

PDSCP = *wert*

Gibt die Codepage an, die in den Members der partitionierten Datei verwendet wird, wobei *wert* eine CCSID-Nummer oder eine Java-Codepagenummer ist. Wenn dieser Parameter nicht angegeben wird, wird die z/OS UNIX System Services-Codepage verwendet. Beispielsweise können Sie PDSCP=037 angeben.

PDSLIB = *wert*

Gibt den Namen der partitionierten Datei an, die die zu verarbeitenden Datenstrukturen der höheren Programmiersprachen enthält.

Restriction: Die Datensätze in der partitionierten Datei müssen eine feste Länge von 80 Bytes haben.

PDSMEM = *wert*

Gibt den Namen des Members der partitionierten Datei an, die die Strukturen der höheren Programmiersprache enthält, die Sie verarbeiten möchten.

STRUCTURE = { **DFHDATA** | *daten* }

Der Name der übergeordneten Datenstruktur in C und C++. Der Standardwert ist DFHDATA.

TRUNCATE-NULL-ARRAYS = { **DISABLED** | **ENABLED** }

Gibt an, wie strukturierte Arrays auf Zuordnungsebene 4.1 oder höher verarbeitet werden. Wenn diese Option aktiviert ist, versucht CICS, leere Datensätze innerhalb eines Arrays zu erkennen (weitere Informationen zur Identifizierung leerer Datensätze finden Sie unter TRUNCATE-NULL-ARRAY-VALUES). Wenn fünf aufeinanderfolgende leere Arraydatensätze erkannt werden, wird das Array bei der Generierung von XML/JSON beim ersten solchen Datensatz abgeschnitten. Diese Abschneidefunktion ist nur für Arrays mit strukturiertem Inhalt aktiviert, Arrays aus einfachen primitiven Feldern werden nicht abgeschnitten. Das Abschneiden von Arrays kann zu einer kürzeren Darstellung der Daten in JSON/XML führen, dies ist jedoch nicht ohne Risiko. Wenn fünf aufeinanderfolgende Datensätze fehlerhaft als nicht initialisierter Speicher identifiziert werden (beispielsweise weil sie legitim niedrige Werte enthalten), kann es zu Datenverlusten kommen. Wenn TRUNCATE-NULL-ARRAYS aktiviert und TRUNCATE-NULL-ARRAY-VALUES nicht festgelegt ist, wird der Standardwert für TRUNCATE-NULL-ARRAY-VALUES verwendet.

TRUNCATE-NULL-ARRAY-VALUES = { **NULL** | **PACKEDZERO** | **SPACE** | **ZERO** }

Gibt an, welche Werte für die Verarbeitung von TRUNCATE-NULL-ARRAYS auf Zuordnungsebene 4.1 oder höher als leer behandelt werden. Standardmäßig wird der Nullwert ('0x00' oder 'low-values') als leer behandelt. Wenn alle Bytes des Speichers in einem Datensatz eines strukturierten Arrays Nullen enthalten, wird der gesamte Datensatz als leer betrachtet. Ein Wert oder mehrere Werte vom Typ NULL, PACKEDZERO, SPACE und ZERO können in einer durch Kommas getrennten Liste angegeben werden.

NULL Impliziert ein Nullzeichen (0x00).

PACKEDZERO

Impliziert eine gepackte Dezimalnull mit positivem Vorzeichen (0x0C), eine gepackte Dezimalnull mit negativem Vorzeichen (0x0D) oder eine gepackte Dezimalnull ohne Vorzeichen (0x0F).

SPACE

Impliziert einen SBCS EBCDIC-Bereich (0x40).

ZERO Impliziert eine gezonte Dezimalnull ohne Vorzeichen (0xF0).

Jede passende Kombination der ausgewählten Bytes innerhalb eines strukturierten Arraydatensatzes führt dazu, dass der gesamte Datensatz als leer erkannt wird.

Wenn für TRUNCATE-NULL-ARRAY-VALUES ein Wert definiert ist, muss TRUNCATE-NULL-ARRAYS aktiviert sein.

```
//LS2JS JOB '  
abrechnungsdaten  
,  
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: Konvertierung eines JSON-Schemas in eine höhere Programmiersprache für eine verknüpfbare Schnittstelle

Die katalogisierte Prozedur DFHJS2LS generiert aus einem JSON-Schema eine Datenstruktur einer höheren Programmiersprache und eine JSON-Bindungsdatei. Verwenden Sie DFHJS2LS, wenn Sie ein CICS-Programm erstellen wollen, das JSON parsen oder erstellen kann. In diesem Thema werden die Jobsteueranweisungen, symbolischen Parameter, Eingabeparameter und ihre Beschreibungen für DFHJS2LS aufgelistet.

Die JCL-Prozedur DFHJS2LS wird in der Datei *HLQ.XDFHINST* installiert, wobei *HLQ* das übergeordnete Qualifikationsmerkmal für die Installationsposition von CICS ist.

Jobsteueranweisungen für DFHJS2LS

JOB Startet den Job.

EXEC Gibt den Prozedurnamen an (DFHJS2LS).

INPUT.SYSUT1 DD

Gibt die Eingabe an. Die Eingabeparameter werden im Eingabedatenstrom angegeben. Sie können auch in einer Datei oder in einem Member einer partitionierten Datei definiert werden.

Symbolische Parameter

Die folgenden symbolischen Parameter sind in DFHJS2LS definiert:

JAVADIR = *pfad*

Gibt den Namen des Java-Verzeichnisses an, das von DFHJS2LS verwendet wird. Der Wert dieses Parameters wird an */usr/lpp/* angehängt, wodurch sich der vollständige Pfadname */usr/lpp/pfad* ergibt.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **JAVADIR** angegeben wurde.

PATHPREF = *präfix*

Gibt ein optionales Präfix an, das den z/OS UNIX-Verzeichnispfad erweitert, der in anderen Parametern verwendet wird. Der Standardwert ist eine leere Zeichenfolge.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **PATHPREF** angegeben wurde.

SERVICE = *wert*

Verwenden Sie diesen Parameter nur, wenn sie vom IBM Support dazu aufgefordert werden.

TMPDIR = *tmp-verzeichnis*

Gibt die Position eines Verzeichnisses in z/OS UNIX an, das DFHJS2LS als temporären Arbeitsbereich verwendet. Die Benutzer-ID, unter der der Job ausgeführt wird, muss über Lese- und Schreibberechtigungen für dieses Verzeichnis verfügen.

Der Standardwert ist /tmp.

TMPPFILE = *tmp-präfix*

Gibt ein Präfix an, das DFHJS2LS verwendet, um die Namen der temporären Arbeitsbereichsdateien zu erstellen.

Der Standardwert ist JS2LS.

USSDIR = *pfad*

Gibt den Namen des CICS TS-Verzeichnisses im UNIX System Services-Dateisystem an. Der Wert dieses Parameters wird an /usr/lpp/cicsts/ angehängt, um den vollständigen Pfadnamen /usr/lpp/cicsts/*pfad* zu erstellen. Er muss als '.' (Punkt) angegeben werden, wenn der Standardwert verwendet wird.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **USSDIR** angegeben wurde.

Der temporäre Arbeitsbereich

DFHJS2LS erstellt die folgenden drei temporären Dateien zur Laufzeit:

tmp-verzeichnis/tmp-präfix.in
tmp-verzeichnis/tmp-präfix.out
tmp-verzeichnis/tmp-präfix.err

Dabei gilt Folgendes:

tmp-verzeichnis ist der Wert, der im Parameter **TMPDIR** angegeben ist.
tmp-präfix ist der Wert, der im Parameter **TMPPFILE** angegeben ist.

Die Standardnamen für die Dateien lauten wie folgt (wenn **TMPDIR** und **TMPPFILE** nicht angegeben sind):

/tmp/JS2LS.in
/tmp/JS2LS.out
/tmp/JS2LS.err

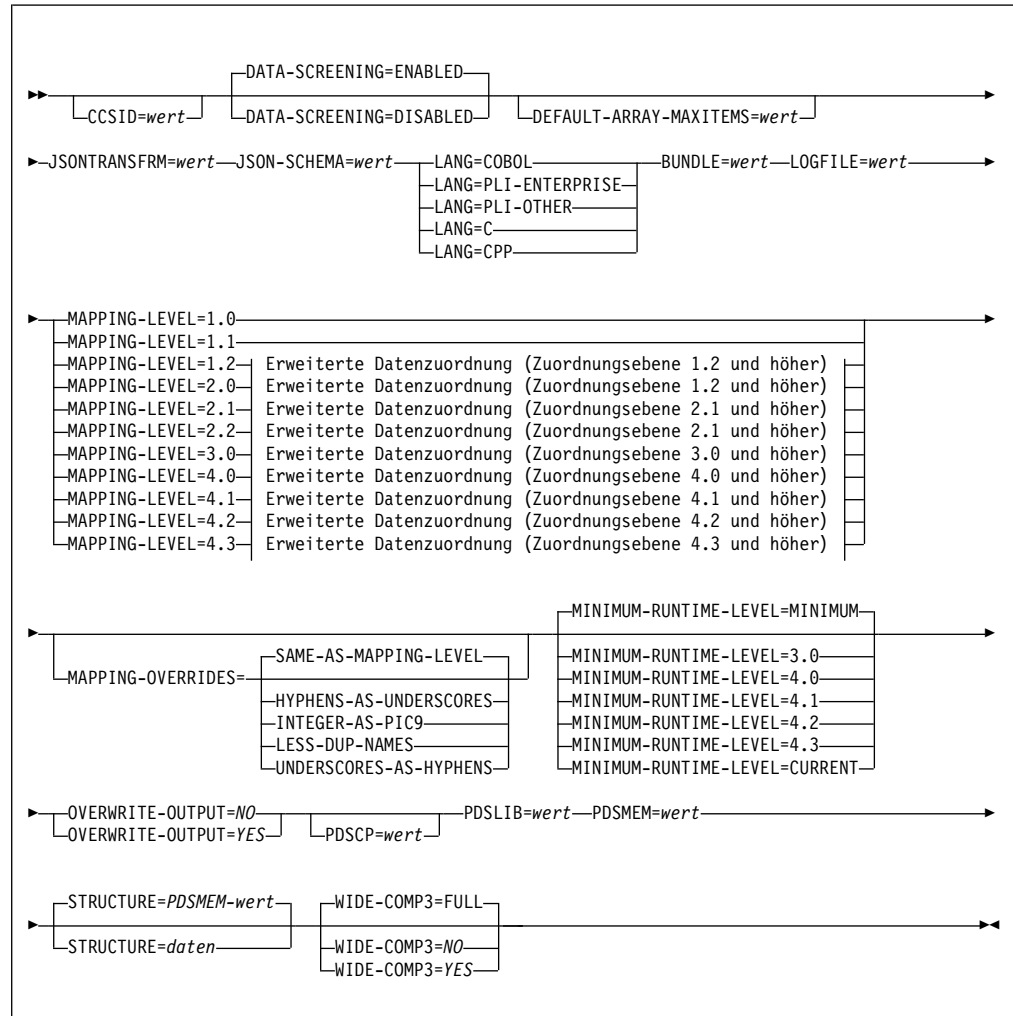
Important: DFHJS2LS sperrt nicht den Zugriff auf die z/OS UNIX-Dateien oder -Dateimember. Wenn folglich zwei oder mehr Instanzen von DFHJS2LS gleichzeitig ausgeführt werden und dieselben temporären Arbeitsbereichsdateien verwenden,

kann nicht verhindert werden, dass ein Job die Arbeitsbereichsdateien überschreibt, während ein anderer Job sie verwendet, was zu unvorhersehbaren Fehlern führt.

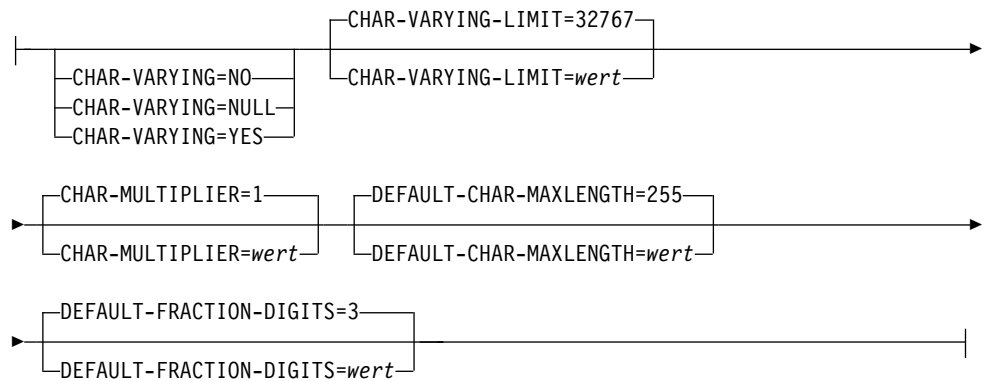
Deshalb wird empfohlen, eine Namenskonvention und Betriebsprozeduren auszu-
arbeiten, die diese Situation verhindern. Sie können beispielsweise den symboli-
schen Systemparameter **SYSUID** verwenden, um Arbeitsbereichsdateinamen zu ge-
nerieren, die für einen einzelnen Benutzer eindeutig sind.

Diese temporären Dateien werden gelöscht, bevor der Job beendet wird.

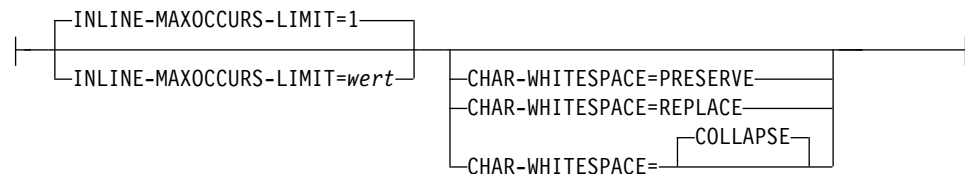
Eingabeparameter für DFHJS2LS



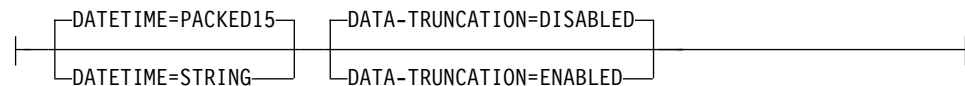
Erweiterte Datenzuordnung (Zuordnungsebene 1.2 und höher):



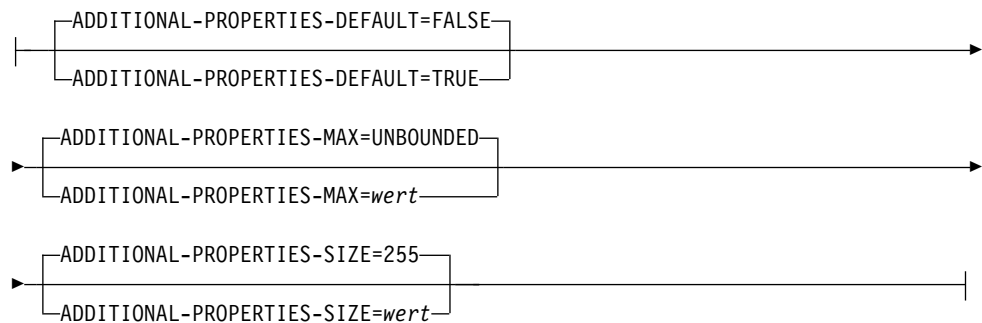
Erweiterte Datenzuordnung (Zuordnungsebene 2.1 und höher):



Erweiterte Datenzuordnung (Zuordnungsebene 3.0 und höher):



Erweiterte Datenzuordnung (Zuordnungsebene 4.2 und höher):



Parameterverwendung

- Sie können die Eingabeparameter in beliebiger Reihenfolge angeben.
- Jeder Parameter muss in einer neuen Zeile eingegeben werden.
- Ein Parameter (und ggf. das Fortsetzungszeichen) darf nicht über Spalte 72 hinausgehen; die Spalten 73-80 müssen Leerzeichen enthalten.
- Wenn ein Parameter zu lang ist, um in eine einzelne Zeile zu passen, verwenden Sie am Ende der Zeile einen Stern (*), um anzugeben, dass der Parameter in der

nächsten Zeile fortgesetzt wird. Alles (einschließlich der Leerzeichen) vor dem Stern wird als Teil des Parameters betrachtet.

- Ein #-Zeichen an der ersten Zeichenposition der Zeile ist ein Kommentarzeichen. Die Zeile wird ignoriert.
- Ein Komma an der letzten Zeichenposition der Zeile ist ein optionales Zeilen-trennzeichen und wird ignoriert.

Parameterbeschreibungen

ADDITIONAL-PROPERTIES-DEFAULT = { **true** | **false** }

Gibt an, ob JSON-Schemaobjekte, die zusätzliche Eigenschaften nicht explizit unterstützen, als unterstützende Elemente interpretiert werden oder nicht. Zusätzliche JSON-Eigenschaften sind alle Eigenschaften in einem JSON-Objekt, die im JSON-Schema nicht vordefiniert sind. Diese Eigenschaften werden in der Regel vom Datenumsetzungsmechanismus als unerwartete Zusatzdaten zurückgewiesen. Wenn **ADDITIONAL-PROPERTIES-DEFAULT** auf TRUE gesetzt ist, oder wenn das JSON-Schema explizit `additionalProperties:true` für ein Objekt festlegt, wird ein Bereich für solche Werte in den generierten Copybooks reserviert. Anwendungen können mit diesen Werten interagieren, indem sie die zugeordneten Felder in den Copybooks verwenden.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

ADDITIONAL-PROPERTIES-MAX = { **0-20** | **UNBOUNDED** }

Gibt an, wie viele zusätzliche Eigenschaften für ein entsprechendes JSON-Objekt unterstützt werden. Siehe **ADDITIONAL-PROPERTIES-DEFAULT**. Die generierten Copybooks enthalten Strukturen, die sich für die Adressierung aller zusätzlichen Eigenschaften eignen. Standardmäßig gibt es keine maximale Einschränkung für die Anzahl der unterstützten Eigenschaften. Die Copybooks werden in ähnlicher Weise wie Arrays ohne Einschränkungen generiert und verwenden Container. Mit diesem Parameter kann eine maximale Einschränkung angewendet werden, die in Kombination mit dem Parameter **INLINE-MAXOCCURS-LIMIT** dafür sorgt, dass ein Array mit fester Länge für die maximale Anzahl von Eigenschaften zugeordnet werden kann, wodurch keine Container erforderlich sind.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

ADDITIONAL-PROPERTIES-SIZE = { **16-32767** | **255** }

Gibt die maximale Größe für jede der zusätzlichen JSON-Eigenschaften an. Wenn ein JSON-Objekt zusätzliche Eigenschaften unterstützt, wie in **ADDITIONAL-PROPERTIES-DEFAULT** definiert, dann haben die generierten Copybooks Bindungen, um Eigenschaften bis zu der von **ADDITIONAL-PROPERTIES-MAX** angegebenen Anzahl zu unterstützen. Standardmäßig beträgt der für jede zusätzliche Eigenschaft unterstützte Maximalwert 255 Zeichen. Ein Feld dieser Größe wird in den erstellten Copybooks generiert. Diese Größe kann durch Festlegen des Parameters **ADDITIONAL-PROPERTIES-SIZE** angepasst werden. So wird beispielsweise ein JSON-Objekt verarbeitet, das folgende Eigenschaft enthält:

```
"example": { "notes": "this extra property was not defined in the JSON Schema"
}
```

Wenn die Copybooks generiert wurden, um zusätzliche Eigenschaften zu unterstützen, wird der gesamte Wert zur Verarbeitung an die Anwendung übergeben. Der Wert beginnt mit dem führenden Anführungszeichen vor dem Eigenschaftsschlüssel und endet mit der abschließenden rechten geschweiften Klammer im Eigenschaftswert. In diesem Beispiel sind es etwa 100 Zeichen. Der Wert für **ADDITIONAL-PROPERTIES-SIZE** muss groß genug sein, um den

größtmöglichen Wert enthalten zu können. Wenn der zugeordnete Puffer für den verarbeiteten Wert zu klein ist, wird eine Fehlerantwort generiert.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

BUNDLE = *wert*

Gibt den Pfad und den Namen des Bundleverzeichnis unter z/OS UNIX an. Wenn Sie diesen Wert angeben, generiert der JSON-Assistent die JSON-Bindung im Bundle-Verzeichnis und erstellt ein Bundlemanifest für Sie. Die Pfadinformationen für diesen Parameter überschreiben alle Pfadinformationen für den Parameter **JSONTRANSFRM**.

CCSID = *wert*

Gibt die CCSID an, die zur Laufzeit verwendet wird, um Zeichendaten in der Anwendungsdatenstruktur zu codieren. Der Wert dieses Parameters überschreibt den Wert des Systeminitialisierungsparameters **LOCALCCSID**. *wert* muss eine EBCDIC-CCSID sein, die von Java- und z/OS-Konvertierungsservices unterstützt wird. Wenn Sie diesen Parameter nicht angeben, wird die Anwendungsdatenstruktur unter Verwendung der im Systeminitialisierungsparameter angegebenen CCSID codiert.

Sie können diesen Parameter mit jeder Zuordnungsebene verwenden.

CHAR-MULTIPLIER = { 1 | *wert* }

Gibt die Anzahl von Bytes an, die für die einzelnen Zeichen auf Zuordnungsebene 1.2 oder höher zulässig sein sollen. Der Wert (*wert*) dieses Parameters kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein. Alle nicht numerischen, zeichenbasierten Zuordnungen unterliegen diesem Multiplikator. Binäre, numerische, in Zonen eingeteilte und gepackte Dezimalfelder unterliegen diesem Multiplikator nicht.

Dieser Parameter kann nützlich sein, wenn Sie beispielsweise planen, DBCS-Zeichen zu verwenden, bei denen Sie sich für einen Multiplikator von 3 entscheiden können, um zur Laufzeit Platz zu lassen für mögliche SO- und SI-Zeichen (shift-out = Umschalttaste nicht gedrückt, shift-in = Umschalttaste gedrückt) um jedes Doppelbytezeichen herum.

Wenn Sie **CCSID=1200** (gibt UTF-16 an) festlegen, sind die einzigen gültigen Werte für **CHAR-MULTIPLIER** 2 oder 4. Wenn Sie UTF-16 verwenden, ist der Standardwert 2. Verwenden Sie **CHAR-MULTIPLIER=2**, wenn Sie erwarten, dass Anwendungsdaten Zeichen enthalten, die eine UTF-16-Codierungseinheit erfordern. Verwenden Sie **CHAR-MULTIPLIER=4**, wenn Sie erwarten, dass Anwendungsdaten Zeichen enthalten, die zwei UTF-16-Codierungseinheiten erfordern.

Anmerkung: Wenn Sie **CHAR-MULTIPLIER** auf 1 festlegen, schließt dies nicht aus, dass DBCS-Zeichen verwendet werden können. Und wenn Sie den Parameter auf 2 festlegen, schließt dies nicht aus, dass UTF-16-Ersatzzeichenpaare verwendet werden können. Wenn jedoch regelmäßig Breitzzeichen verwendet werden, werden einige gültige Werte nicht in das zugeordnete Feld passen. Wenn ein größerer Wert für **CHAR-MULTIPLIER** verwendet wird, ist es möglich, mehr Zeichen in dem zugeordneten Feld zu speichern als in der XML gültig sind. Achten Sie darauf, die passenden Bereichseinschränkungen einzuhalten.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Gibt an, wie Zeichendaten mit variabler Länge auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Binärdatentypen mit variabler Länge werden immer entweder einem Container oder einer variierenden Struktur zugeordnet. Wenn Sie diesen Parameter nicht angeben, hängt die Standardzuordnung von der angegebenen Sprache ab. Sie haben folgende Optionen:

- NO** Zeichendaten mit variabler Länge werden als Zeichenfolgen mit fester Länge zugeordnet.
- NULL** Zeichendaten mit variabler Länge werden auf null endenden Zeichenfolgen zugeordnet.
- YES** Zeichendaten mit variabler Länge werden in PL/I einem CHAR VARYING-Datentyp zugeordnet. In den COBOL-, C- und C++-Sprachen werden Zeichendaten mit variabler Länge einer äquivalenten Darstellung zugeordnet, die aus zwei verwandten Elementen besteht: der Datenlänge und den Daten.

CHAR-VARYING-LIMIT = { 32767 | *wert* }

Gibt die maximale Größe von Binärdaten und Zeichendaten mit variabler Länge an, die der Sprachstruktur auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Wenn die Zeichen oder Binärdaten größer sind als der in diesem Parameter angegebene Wert, werden sie einem Container zugeordnet, und der Containername wird in der generierten Sprachstruktur verwendet. Der Wert (*wert*) kann zwischen 0 und dem Standardwert 32.767 Bytes liegen.

CHAR-WHITESPACE = COLLAPSE | **REPLACE** | **PRESERVE**

Gibt an, wie Leerzeichen in Werten vom Typ Zeichenfolge von CICS verarbeitet werden.

COLLAPSE

Führende, abschließende und eingebettete Leerzeichen werden entfernt und alle Tabulatoren, neuen Zeilen und aufeinanderfolgenden Leerzeichen werden durch einzelne Leerzeichen ersetzt.

REPLACE

Alle Tabulatoren oder neuen Zeilen werden durch die entsprechende Anzahl von Leerzeichen ersetzt.

PRESERVE

Alle Leerzeichen in dem Datenwert werden beibehalten.

Wenn der Parameter **CHAR-WHITESPACE** nicht festgelegt ist, werden Leerzeichen komprimiert.

Anmerkung: Dieser Parameter gilt nicht für Felder mit dem Format date-time, uri, base64Binary oder hexBinary, bei denen Leerzeichen immer komprimiert werden.

DATA-SCREENING = { ENABLED | **DISABLED** }

Gibt an, ob die von Anwendungen bereitgestellten Daten auf Fehler überprüft werden.

ENABLED

Alle von Anwendungen bereitgestellten Laufzeitdaten, die nicht mit der Sprachstruktur konsistent sind, werden als Fehler behandelt, und es wird die Nachricht DFHPI1010 ausgegeben. Eine Fehlerantwort wird an die Anwendung zurückgegeben.

DISABLED

Werte in den von Anwendungen bereitgestellten Laufzeitdaten, die nicht konsistent mit der Sprachstruktur sind, werden durch Standardwerte ersetzt. Eine Null ersetzt beispielsweise einen fehlerhaften Wert in einem numerischen Feld. Die Nachricht DFHPI1010 wird nicht ausgegeben und es wird eine normale Antwort an die Anwendung zurückgegeben. Dieses Feature kann verwendet werden, um die Feh-

lerantworten `INVALID_PACKED_DEC` und `INVALID_ZONED_DEC` zu vermeiden, die von nicht initialisierten Ausgabefeldern generiert werden.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Gibt an, ob Daten mit variabler Länge in einer Feldstruktur mit fester Länge toleriert werden:

DISABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, weist CICS die abgeschnittenen Daten zurück und gibt eine Fehlermeldung aus.

ENABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, toleriert CICS die abgeschnittenen Daten und verarbeitet die fehlenden Daten als Nullwerte.

DATETIME = { **PACKED15** | **STRING** }

Gibt an, dass JSON-Datum/Zeit-Eigenschaften dem Datenformat CICS ABSTIME oder Text zugeordnet werden:

PACKED15

JSON-Datum/Zeit-Eigenschaften werden dem Format CICS ABSTIME zugeordnet.

STRING

JSON-Datum/Zeit-Eigenschaften werden Text zugeordnet. Die Zuordnung ist identisch mit allen vorherigen Zuordnungsebenen.

Sie können diesen Parameter auf der Zuordnungsebene 3.0 verwenden.

DEFAULT-ARRAY-MAXITEMS = *wert*

Gibt die maximale Arraygrenze an, die angewendet werden soll, wenn im JSON-Schema keine Angabe zur maximalen Anzahl von Vorkommen (`maxItems`) eingeschlossen ist. Wenn dieser Parameter nicht festgelegt ist, wird kein Maximalwert angewendet. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein. Dieser Parameter kann mit dem Parameter **INLINE-MAXOCCURS-LIMIT** kombiniert werden, um die Zuordnung von JSON-Arrays in die Sprachstrukturen zu beeinflussen.

DEFAULT-CHAR-MAXLENGTH = { **255** | *wert* }

Gibt die Standardarraylänge von Zeichendaten in Zeichen für Zuordnungen an, in deren JSON-Schemadokument keine Länge eingeschlossen ist, wenn die Zuordnungsebene 1.2 oder höher ist. Der Wert (*wert*) dieses Parameters kann eine positive ganze Zahl im Bereich von 1 bis 2.147.483.647 sein.

DEFAULT-FRACTION-DIGITS = { **3** | *wert* }

Gibt die Standardanzahl an Bruchteilstellen für einen JSON-Dezimalschematyp an. Der Standardwert ist 3. Für COBOL liegt der gültige Bereich zwischen 0 und 17, wenn der Parameter **WIDE-COMP3** verwendet wird. Für C oder PLI liegt der gültige Bereich zwischen 0 und 30.

INLINE-MAXOCCURS-LIMIT = { **1** | *wert* }

Gibt auf Basis des Attributs `maxItems` der JSON-Schemaschlüsselwörter an, ob variabel wiederkehrende Inline-Inhalte verwendet werden.

Der Parameter **INLINE-MAXOCCURS-LIMIT** ist nur ab Zuordnungsebene 2.1 verfügbar. Der Wert (*wert*) von **INLINE-MAXOCCURS-LIMIT** kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein. Der Wert 0 gibt an, dass die Inline-Zuordnung nicht verwendet wird. Der Wert 1 gibt an, dass optionale Elemente inline zugeordnet werden. Wenn die Variable *wert* des Attributs `maxItems` grö-

ßer als die Variable *wert* von **INLINE-MAXOCCURS-LIMIT** ist, wird eine containerbasierte Zuordnung verwendet. Andernfalls wird eine Inline-Zuordnung verwendet.

Wenn Sie entscheiden, dass variabel wiederkehrende Listen inline zugeordnet werden sollen, ziehen Sie die Länge der einzelnen Elemente von wiederkehrenden Daten in Betracht. Wenn es wenige lange Instanzen gibt, ist die containerbasierte Zuordnung die bevorzugte Lösung. Wenn es viele kurze Instanzen gibt, ist die Inline-Zuordnung geeigneter.

JSONTRANSFRM = *wert*

Dieser Parameter ist für den LINKable-Modus obligatorisch, aber ungültig für Anforderung/Antwort- und REST-konforme Modi. Er gibt den Namen an, der für die **JSONTRANSFRM**-Bundleressource in CICS verwendet wird.

JSON-SCHEMA = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, aus der das JSON-Schema gelesen wird. Bei Bedarf erstellt DFHJS2LS die Datei, aber nicht die Verzeichnisstruktur.

LANG = **COBOL**

Gibt an, dass die Datenstruktur der höheren Programmiersprache COBOL ist.

LANG = **PLI-ENTERPRISE**

Gibt an, dass die Datenstruktur der höheren Programmiersprache Enterprise PL/I ist.

LANG = **PLI-OTHER**

Gibt an, dass die Datenstruktur der höheren Programmiersprache eine andere Version von PL/I ist als Enterprise PL/I.

LANG = **C**

Gibt an, dass die Datenstruktur der höheren Programmiersprache C ist.

LANG = **CPP**

Gibt an, dass die Datenstruktur der höheren Programmiersprache C++ ist.

LOGFILE = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die DFHJS2LS das Aktivitätenprotokoll und die Traceinformationen schreibt. Bei Bedarf erstellt DFHJS2LS die Datei, aber nicht die Verzeichnisstruktur.

In der Regel verwenden Sie diese Datei nicht, aber dies kann von der IBM Serviceorganisation angefordert werden, wenn Sie Probleme mit DFHJS2LS feststellen.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 }

Gibt die Zuordnungsebene für den Assistenten an, der beim Generieren der JSON-Bindung und der Sprachstrukturen verwendet werden soll. Sie müssen die aktuellste Zuordnungsebene verwenden, die verfügbar ist. Für DFHJS2LS müssen Sie die Zuordnungsebene 3.0 oder höher verwenden.

3.0 Dies ist die minimale Zuordnungsebene, die Sie mit DFHJS2LS verwenden können.

4.0 Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher, wenn Sie UTF-16 verwenden möchten.

4.1 Zur Unterstützung abschneidbarer Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher.

- 4.2 Für zusätzliche Eigenschaften verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.
- 4.3 Zur Unterstützung mehrdimensionaler Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERSCORES | INTEGER-AS-PIC9 | LESS-DUP-NAMES | UNDERSCORES-AS-HYPHENS | NO-ARRAY-NAME-INDEXING }

Gibt an, ob das Standardverhalten für die angegebene Zuordnungsebene beim Generieren von Sprachstrukturen überschrieben wird.

SAME-AS-MAPPING-LEVEL

Dieser Parameter generiert Sprachstrukturen auf dieselbe Weise wie die Zuordnungsebene. Dies ist die Standardeinstellung.

HYPHENS-AS-UNDERSCORES

Nur für PL/I. Dieser Parameter konvertiert alle Bindestriche im JSON-Schema in Unterstriche und nicht in das Zeichen X, um die Lesbarkeit der generierten PL/I-Sprachstrukturen zu verbessern. Weitere Informationen finden Sie unter JSON schema to PL/I mapping. Diese Option ist auf Zuordnungsebene 4.2 automatisch aktiviert.

INTEGER-AS-PIC9

Nur für COBOL und DFHJS2LS. Dieser Parameter generiert Sprachstrukturen, die ganzzahlige Werte aus dem JSON-Schema in Form von Ziffern und nicht von alphanumerischen Zeichen enthalten. Diese Option ist auf Zuordnungsebene 4.0 automatisch aktiviert.

LESS-DUP-NAMES

Dieser Parameter generiert nicht strukturelle Strukturfeldnamen mit `_value` am Ende des Namens, um das direkte Referenzieren des Felds zu aktivieren. Beispiel: In der folgenden PL/I-Sprachstruktur wird, wenn `MAPPING-OVERRIDES=LESS-DUP-NAMES` angegeben ist, an das Ebene-12-Feld 'streetName' das Element `_value` angehängt:

```
09 streetName,
12 streetName CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

Die resultierende Struktur sieht wie folgt aus:

```
09 streetName,
12 streetName_value CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

Diese Option ist auf Zuordnungsebene 4.2 automatisch aktiviert.

UNDERSCORES-AS-HYPHENS

Nur für COBOL. Dieser Parameter konvertiert alle Unterstriche im JSON-Schema in Bindestriche und nicht in das Zeichen X, um die Lesbarkeit der generierten COBOL-Sprachstrukturen zu verbessern. Im Fall von Feldnamenskollisionen werden die Felder nummeriert, um sicherzustellen, dass sie eindeutig sind. Weitere Informationen finden Sie unter JSON schema to COBOL mapping.

Diese Option ist auf Zuordnungsebene 4.0 automatisch aktiviert.

NO-ARRAY-NAME-INDEXING

Nur für COBOL und Enterprise PL/I. Stellt sicher, dass die Feldnamen in einem Array nur im Rahmen der übergeordneten Struktur eindeutig sind.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 | CURRENT }

Gibt die älteste CICS-Laufzeitumgebung an, in der die JSON-Bindung implementiert werden kann. Wenn Sie eine Version auswählen, die nicht mit den anderen Parametern übereinstimmt, die Sie angegeben haben, erhalten Sie eine Fehlermeldung. Sie können die folgenden Optionen auswählen:

MINIMUM

Die niedrigste mögliche Laufzeitebene von CICS wird anhand der von Ihnen ausgewählten Parameter automatisch zugewiesen.

- 3.0** Geben Sie die Laufzeitebene 3.0 oder höher an, wenn Sie den CICS JSON-Assistenten verwenden und von fortschrittlichen Datenzuordnungen profitieren möchten.
- 4.0** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.0 oder niedriger verwenden. Sie können alle optionalen Parameter auf dieser Ebene verwenden.
- 4.1** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.1 oder niedriger verwenden.
- 4.2** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS V5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.2 oder niedriger verwenden.
- 4.3** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.3 oder niedriger verwenden.

CURRENT

Verwenden Sie diese Laufzeitebene, um die generierte Bindungsdatei in einer CICS-Region zu implementieren, die dieselbe Laufzeitumgebung wie die Region hat, die zum Generieren der Bindungsdatei verwendet wurde.

OVERWRITE-OUTPUT = { NO | YES }

Steuert, ob vorhandene CICS-BUNDLES im Dateisystem überschrieben werden können.

- NO** Vorhandene BUNDLES werden nicht ersetzt. Wenn ein vorhandenes BUNDLE gefunden wird, gibt DFHJS2LS die Fehlermeldung DFHPI9689E aus und wird beendet.
- YES** Vorhandene BUNDLES werden ersetzt. Wenn ein vorhandenes BUNDLE gefunden wird, wird die Nachricht DFHPI9683W ausgegeben, um Sie darüber zu informieren, dass die Datei ersetzt wurde.

PDSCP = *wert*

Gibt die Codepage an, die in den Mitgliedern der partitionierten Datei verwendet wird, wobei *wert* eine CCSID-Nummer oder eine Java-Codepagenummer

ist. Wenn dieser Parameter nicht angegeben wird, wird die z/OS UNIX System Services-Codepage verwendet. Beispielsweise können Sie PDSCP=037 angeben.

PDSLIB = *wert*

Gibt den Namen der partitionierten Datei an, die die generierte höhere Programmiersprache enthält.

PDSMEM = *wert*

Gibt das ein bis sechs Zeichen lange Präfix an, das DFHJS2LS verwendet, um den Namen des Members der partitionierten Datei zu generieren, die die Strukturen der höheren Programmiersprachen enthalten wird.

DFHJS2LS generiert ein Member einer partitionierten Datei pro Operation. Der Membername wird durch das Anhängen einer zweistelligen Zahl an das Präfix generiert.

STRUCTURE = { *PDSMEM-wert* | *daten* }

Der Name der übergeordneten Datenstruktur in C und C++. Der Standardwert ist der Wert des **PDSMEM**-Parameters.

WIDE-COMP3 = { **FULL** | **NO** | **YES** }

Steuert die maximale Länge der gepackten Dezimalvariablen in der generierten COBOL- oder PL/I-Sprachstruktur.

FULL Für COBOL und PL/I generiert DFHJS2LS ein gepacktes Dezimalfeld, das groß genug ist, um alle gültigen Werte aufzunehmen. Die maximale Länge sind 31 Ziffern. Dies ist die Standardeinstellung.

NO Nur für COBOL. DFHJS2LS begrenzt die Länge der gepackten Dezimalvariablen auf 18 Ziffern, wenn der COBOL-Sprachstrukturtyp COMP-3 generiert wird. Wenn die gepackte Dezimalzahl länger als 18 Ziffern ist, wird die Nachricht DFHPI9022W ausgegeben, um darauf hinzuweisen, dass der angegebene Typ auf höchstens 18 Ziffern begrenzt ist.

YES Nur für COBOL. DFHJS2LS unterstützt die maximale Länge von 31 Ziffern, wenn der COBOL-Sprachstrukturtyp COMP-3 generiert wird.

Anmerkung: Die Optionen NO und YES generieren Felder, die nicht alle gültigen Werte darstellen können. Mit der Option FULL wird dieses Problem behoben. Allerdings lässt die Option FULL auch die Darstellung einiger ungültiger Werte in dem gepackten Dezimalfeld zu. Wenn ein Schema beispielsweise angibt, dass es maximal fünf Ziffern und maximal zwei Nachkommastellen gibt, generiert die Option FULL ein gepacktes Dezimalfeld, das sieben Ziffern zulässt, wodurch Platz für gültige Werte wie 25000 und 999,99 verfügbar ist, aber auch genug Platz für ungültige Werte wie 9999,99. Wenn Sie die Option FULL verwenden, achten Sie darauf, keine ungültigen Werte in Anwendungsdaten zu generieren.

Beispiel

```
//JS2LS JOB '  
abrechnungsdaten  
,  
name,MSGCLASS=A  
// SET QT='''  
//JAVAPROG EXEC DFHJS2LS,  
// TMPFILE=&QT.&SYSUID.&QT  
/INPUT.SYSUT1 DD *  
PDSLIB=//CICSHLQ.SDFHSAMP  
PDSMEM=CPYBK2  
JSON-SCHEMA=example.json  
LANG=COBOL
```

```
LOGFILE=/u/exampleapp/example.log
MAPPING-LEVEL=4.0
CHAR-VARYING=NULL
JSONTRANSFORM=EXAMPLE
BUNDLE=/u/exampleapp/bundles/exampleBundle
/*
```

Zuordnungsebenen für den CICS JSON-Assistenten

Eine Zuordnung besteht aus einer Anzahl von Regeln, die angeben, wie Informationen zwischen Sprachstrukturen und JSON-Schemas konvertiert werden. Damit Sie von den ausgereiftesten verfügbaren Zuordnungen profitieren, wird empfohlen, für den Parameter **MAPPING-LEVEL** in dem CICS-Assistenten die neueste Ebene festzulegen.

Jede Zuordnungsebene übernimmt die Funktionalität der vorherigen Zuordnung, wobei die höchste Zuordnungsebene die beste Auswahl an Funktionen bietet. Die höchste Zuordnungsebene bietet mehr Kontrolle über die Datenkonvertierung zur Laufzeit und entfernt Einschränkungen bei der Unterstützung für bestimmte Datentypen und JSON-Eigenschaften.

Sie können den Parameter **MAPPING-LEVEL** auf eine niedrigere Ebene festlegen, wenn Sie Anwendungen erneut implementieren möchten, die bereits auf dieser Ebene implementiert wurden.

Einschränkungen auf allen Zuordnungsebenen

- Im JSON-Schema verwendete Datentypen müssen explizit deklariert werden.
- JSON-Objektreferenzen auf externe Dokumente unter Verwendung von \$ref werden im JSON-Schema nicht unterstützt.

Zuordnungsebene 4.3

Zuordnungsebene 4.3 ist mit CICS TS V5.4 und höher kompatibel.

Zuordnungsebene 4.3 dient in erster Linie der Verwendung mit DFHJS2LS, ist aber auch in den CICS-Web-Service-Assistenten, den XML-Assistenten und den JSON-Assistenten enthalten. Diese Zuordnungsebene implementiert Unterstützung für mehrdimensionale Arrays in JSON.

Zuordnungsebene 4.2

Zuordnungsebene 4.2 ist mit CICS TS V5.4 und höher kompatibel.

Zuordnungsebene 4.2 dient in erster Linie der Verwendung mit DFHJS2LS, ist aber auch in den CICS-Web-Service-Assistenten, den XML-Assistenten und den JSON-Assistenten enthalten. Diese Zuordnungsebene implementiert Unterstützung für zusätzliche Eigenschaften in JSON und führt die folgenden drei Parameter in DFHJS2LS ein: **ADDITIONAL-PROPERTIES-DEFAULT**, **ADDITIONAL-PROPERTIES-MAX** und **ADDITIONAL-PROPERTIES-SIZE**.

Zuordnungsebene 4.1

Zuordnungsebene 4.1 ist mit einer CICS TS 5.3-Region und höher kompatibel.

Zuordnungsebene 4.1 wird den CICS-Web-Service-Assistenten, XML-Assistenten und JSON-Assistenten hinzugefügt. Diese Zuordnungsebene implementiert verbesserte Zuordnungen für einfache Arrays, die von unten nach oben aus vorhandenen

Copybooks generiert werden. Sie fügt außerdem die Möglichkeit hinzu, in CICS nicht initialisierten abschließenden Speicher in Arrays automatisch zu erkennen und diese Datensätze aus dem generierten XML/JSON-Formular auszuschließen.

DFHLS2WS, DFHWS2LS, DFHLS2SC, DFHSC2LS, DFHJS2LS und DFHLS2JS unterstützen die Parameter **TRUNCATE-NULL-ARRAYS** und **TRUNCATE-NULL-ARRAY-VALUES**.

Wenn Sie einen beliebigen Wert für **TRUNCATE-NULL-ARRAY-VALUES** angeben, müssen Sie auch **TRUNCATE-NULL-ARRAYS=ENABLED** angeben.

Zuordnungsebene 4.0

Diese Zuordnungsebene stellt die folgende Unterstützung bereit:

Auf Zuordnungsebene 4.0 und höher unterstützt DFHLS2JS die COBOL OCCURS DEPENDING ON-Klausel und die Zuordnung von COBOL-Zeichenarrays in JSON-Schemas. Sie können dieses Verhalten mithilfe des Parameters **CHAR-OCCURS** im CICS JSON-Assistenten festlegen.

- Sie müssen den Parameter **DATA-TRUNCATION=ENABLED** angeben.
- Eine komplexe Klausel OCCURS DEPENDING ON wird nicht unterstützt. Diese Einschränkung bedeutet, dass OCCURS DEPENDING ON nur für das letzte Feld einer Struktur verwendet wird.
- CICS unterstützt keine qualifizierten Namen (unter Verwendung des Schlüsselworts 'OF') als Ziel einer OCCURS DEPENDING ON-Klausel, z. B. FIELD1 OF STRUCTURE1.
- CICS unterstützt nicht das Schlüsselwort UNBOUNDED. Sie müssen eine ganze Zahl angeben, die die maximale Größe der Tabelle festlegt, die von der Anwendung erwartet wird.

Auf Zuordnungsebene 4.0 und höher unterstützen JSON-Web-Services die Konvertierung von Anwendungsdaten, die mit UTF-16 Unicode codiert sind.

- Wenn Sie LS2JS verwenden, können Sie dieses Verhalten aktivieren, indem Sie sprachspezifische Datentypen für UTF-16 verwenden.
- Wenn Sie JS2LS verwenden, können Sie dieses Verhalten aktivieren, indem Sie CCSID=1200 festlegen.
- CICS unterstützt nur eine einzelne Unicode-Codepage, „ UTF-16BE mit IBM Private Use Area “ (CCSID 1200).
- Die Konvertierung von Anwendungsdaten, die mit UTF-8 codiert sind, wird nicht unterstützt.

Anmerkung: DFHLS2JS bietet keine Unterstützung für die COBOL-Klausel GROUP USAGE NATIONAL.

Zuordnungsebene 3.0 und höher

Diese Zuordnungsebene stellt die folgende Unterstützung bereit:

- DFHJS2LS ordnet date-time-Datentypen dem CICS ASKTIME-Format zu.
- DFHLS2JS kann ein JSON-Schema und eine Web-Service-Bindung aus einer Anwendung erstellen, die nicht nur einen, sondern viele Container verwendet.
- Tolerieren von abgeschnittenen Daten, die durch eine Datenstruktur fester Länge beschrieben werden. Sie können dieses Verhalten festlegen, indem Sie den Parameter **DATA-TRUNCATION** im CICS-Assistenten verwenden.

Zuordnungsebene 2.2 und höher

Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

Zuordnungsebene 2.1 und höher

Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

Diese Zuordnungsebene umfasst eine bessere Kontrolle über die Verarbeitung von Variableninhalten mit dem neuen Parameter **INLINE-MAXOCCURS-LIMIT** und den neuen Werten des Parameters **CHAR-VARYING**.

Auf Zuordnungsebene 2.1 und höher bietet DFHJS2LS die folgende neue und verbesserte Unterstützung für Arrays:

- Parameter **INLINE-MAXOCCURS-LIMIT**
- Eigenschaft `minItems`

Der Parameter **INLINE-MAXOCCURS-LIMIT** gibt an, ob variabel wiederkehrende Listen inline zugeordnet werden. Weitere Informationen zur Inline-Zuordnung von variabel wiederkehrenden Inhalten finden Sie unter „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.

Auf Zuordnungsebene 2.1 und höher unterstützt DFHLS2JS die folgenden JSON-Zuordnungen:

- FILLER-Felder in COBOL und PL/I werden ignoriert.
- Der Wert **COLLAPSE** für den Parameter **CHAR-VARYING**
- Der Wert **BINARY** für den Parameter **CHAR-VARYING**

FILLER-Felder in COBOL und PL/I werden ignoriert. Sie werden nicht im generierten JSON-Schema angezeigt und eine entsprechende Lücke wird zur Laufzeit in den Datenstrukturen beibehalten.

COLLAPSE sorgt dafür, dass CICS nachgestellte Leerzeichen in Textfeldern ignoriert.

BINARY bietet Unterstützung für Binärfelder. Dieser Wert ist bei der Konvertierung von COBOL in ein JSON-Schema nützlich. Diese Option ist nur in SBCS-Zeichenarrays verfügbar und ermöglicht das Zuordnen des Arrays zu einer JSON-Zeichenfolge fester Länge, die statt einer normalen Zeichenfolge Base64-codierte Daten enthält.

Zuordnungsebene 1.2 und höher

Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

Mehr Kontrolle über die Umsetzung von Zeichen- und Binärdaten zur Laufzeit ist mit diesen zusätzlichen Parametern in den Stapelverarbeitungstools verfügbar:

- **CHAR-VARYING**
- **CHAR-VARYING-LIMIT**
- **CHAR-MULTIPLIER**
- **DEFAULT-CHAR-MAXLENGTH**

Wenn Sie entscheiden, den Parameter **CHAR-MULTIPLIER** in DFHJS2LS zu verwenden, gelten die folgenden Regeln, nachdem der Wert dieses Parameters zur Berechnung des Platzes verwendet wurde, der für Zeichendaten erforderlich ist.

- DFHJS2LS stellt die folgenden Zuordnungen bereit:
 - Zeichendatentypen variabler Länge, die eine maximale Länge von mehr als 32.767 Bytes haben, werden einem Container zugeordnet. Mit dem Parameter **CHAR-VARYING-LIMIT** können Sie ein niedrigeres Limit festlegen. In der Sprachstruktur wird ein 16-Byte-Feld erstellt, in dem der Name des Containers gespeichert wird. Zur Laufzeit werden die Zeichendaten in einem Container gespeichert, und der Containername wird in die Sprachstruktur gestellt.
 - Zeichendatentypen variabler Länge, die eine maximale Länge von weniger als 32.768 Bytes haben, werden einer VARYING-Struktur für alle Sprachen außer C/C++ und Enterprise PL/I zugeordnet. In C/C++ werden diese Datentypen auf null endenden Zeichenfolgen zugeordnet und in Enterprise PL/I werden diese Datentypen VARYINGZ-Strukturen zugeordnet. Sie können den Parameter **CHAR-VARYING** verwenden, um die Art der Zuordnung von Zeichendaten variabler Länge auszuwählen.
 - Binärdaten variabler Länge, die eine maximale Länge von weniger als 32.768 Bytes haben, werden einer VARYING-Struktur für alle Sprachen zugeordnet. Wenn die maximale Länge größer-gleich 32.768 Bytes ist, werden die Daten einem Container zugeordnet. In der Sprachstruktur wird ein 16-Byte-Feld erstellt, in dem der Name des Containers gespeichert wird. Zur Laufzeit werden die Binärdaten in einem Container gespeichert, und der Containername wird in die Sprachstruktur gestellt.

Wenn in Ihrem JSON-Schema Zeichendatentypen enthalten sind, denen keine Länge zugeordnet wurde, können Sie mithilfe des Parameters **DEFAULT-CHAR-MAXLENGTH** in DFHJS2LS eine Standardlänge zuweisen.

DFHLS2JS stellt die folgenden Zuordnungen bereit:

- Zeichenfelder werden einem string-Datentyp zugeordnet und können als Felder fester Länge oder als auf null endende Zeichenfolgen zur Laufzeit verarbeitet werden. Mit dem Parameter **CHAR-VARYING** können Sie die Art der Verarbeitung von Zeichendaten variabler Länge zur Laufzeit für alle Sprachen außer PL/I auswählen.
- Base64Binary-Datentypen werden auch einem Container zugeordnet, wenn die maximale Länge der Daten größer ist als 32.767 Bytes oder wenn die Länge nicht definiert ist. Wenn die Länge der Daten 32.767 Bytes oder weniger ist, wird der base64Binary-Datentyp einer VARYING-Struktur für alle Sprachen zugeordnet.

Zuordnungsebene 1.1 und höher

Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

Diese Zuordnungsebene bietet eine verbesserte Zuordnung von JSON-Zeichen- und Binärdatentypen, insbesondere bei der Zuordnung von Daten variabler Länge, deren Eigenschaften `maxLength` und `minLength` mit verschiedenen Werten im JSON-Schema definiert sind. Die Daten werden wie folgt gehandhabt:

- Zeichen- und Binärdatentypen mit einer festen Länge größer als 16 MB werden einem Container für alle Sprachen außer PL/I zugeordnet. In PL/I werden Zeichen- und Binärdatentypen mit einer festen Länge größer als 32.767 Bytes einem Container zugeordnet. In der Sprachstruktur wird ein 16-Byte-Feld erstellt, in

dem der Name des Containers gespeichert wird. Zur Laufzeit werden die Daten fester Länge in einem Container gespeichert, und der Containername wird in die Sprachstruktur gestellt.

Da Container eine variable Länge haben können, werden Daten fester Länge, die einem Container zugeordnet werden, nicht um Leerzeichen oder Nullen ergänzt bzw. abgeschnitten, um der festen Länge zu entsprechen, die im JSON-Schema angegeben ist. Wenn die Länge der Daten signifikant ist, können Sie entweder Ihre Anwendung so schreiben, dass sie diese prüft oder Sie können die Validierung in der CICS-Region aktivieren. Die Validierung wirkt sich deutlich auf die Leistung aus.

Nur Zuordnungsebene 1.1

Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

Diese Zuordnungsebene bietet eine verbesserte Zuordnung von JSON-Zeichen- und Binärdatentypen, insbesondere bei der Zuordnung von Daten variabler Länge, deren Eigenschaften `maxLength` und `minLength` mit verschiedenen Werten im JSON-Schema definiert sind. Daten werden auf die folgenden Arten verarbeitet:

- Binärdatentypen variabler Länge werden einem Container zugeordnet. In der Sprachstruktur wird ein 16-Byte-Feld erstellt, in dem der Name des Containers gespeichert wird. Zur Laufzeit werden die Binärdaten in einem Container gespeichert, und der Containername wird in die Sprachstruktur gestellt.
- Zeichendatentypen variabler Länge, die eine maximale Länge von mehr als 32.767 Bytes haben, werden einem Container zugeordnet. In der Sprachstruktur wird ein 16-Byte-Feld erstellt, in dem der Name des Containers gespeichert wird. Zur Laufzeit werden die Zeichendaten in einem Container gespeichert, und der Containername wird in die Sprachstruktur gestellt.
- Zeichen- und Binärdatentypen mit einer festen Länge kleiner als 16 MB werden Feldern fester Länge für alle Sprachen außer PL/I zugeordnet. In PL/I werden Zeichen- und Binärdatentypen mit einer festen Länge von 32.767 Bytes oder weniger Feldern fester Länge zugeordnet.
- CICS codiert und decodiert Daten im `hexBinary`-Format, aber nicht im `base64Binary`-Format. `Base64Binary`-Datentypen im JSON-Schema werden einem Feld in der Sprachstruktur zugeordnet. Die Größe des Felds wird mithilfe dieser Formel berechnet: $4 \times (\text{ceil}(z / 3))$. Dabei gilt Folgendes:
 - z ist die Länge des Datentyps im JSON-Schema.
 - $\text{ceil}(x)$ ist die kleinste ganze Zahl größer als oder gleich x .

Wenn die Länge von z größer als 24.566 Bytes ist, wird die resultierende Sprachstruktur nicht kompiliert. Wenn Sie `base64Binary`-Daten größer als 24.566 Bytes haben, empfehlen wir, die Zuordnungsebene 1.2 zu verwenden. Auf Zuordnungsebene 1.2 können Sie die `base64Binary`-Daten einem Container zuordnen, statt ein Feld in der Sprachstruktur zu nutzen.

Nur Zuordnungsebene 1.0

Diese Option wird aus Gründen der Kompatibilität mit SOAP-Web-Services beibehalten. Es wird nicht empfohlen, sie mit JSON zu verwenden.

Beachten Sie die folgenden Einschränkungen, die auf neueren Zuordnungsebenen geändert wurden:

- DFHJS2LS ordnen Zeichen- und Binärdatentypen im XML-Schema Feldern fester Länge in der Sprachstruktur zu. Sehen Sie sich dieses partielle JSON-Schema an:

```
"example":{
  "type":"string",
  "maxLength":33000
```

Das partielle JSON-Schema wird in einer COBOL-Sprachstruktur wie folgt dargestellt:

```
15 example PIC X(33000)
```

- CICS codiert und decodiert Daten im hexBinary-Format, aber nicht im base64Binary-Format. DFHJS2LS ordnet Base64Binary-Daten einem Zeichenfeld fester Länge zu, dessen Inhalte von dem Anwendungsprogramm codiert bzw. decodiert werden müssen.
- DFHLS2JS interpretiert Zeichen- und Binärfelder in der Sprachstruktur als Felder fester Länge und ordnet diese Felder JSON-Zeichenfolgen zu, die eine Eigenschaft maxLength aufweisen. Zur Laufzeit werden die Felder in der Sprachstruktur mit Leerzeichen oder Nullen gefüllt, wenn nicht ausreichend Daten verfügbar sind.

Zuordnung einer höheren Programmiersprache zu einem JSON-Schema

Mit dem CICS JSON-Assistenten können Sie Zuordnungen zwischen Strukturen einer höheren Programmiersprache und JSON-Schemas generieren. Außerdem generiert der CICS JSON-Assistent JSON-Schemas aus Datenstrukturen einer höheren Programmiersprache oder umgekehrt.

Die Dienstprogramme DFHJS2LS und DFHLS2JS werden gesammelt als CICS JSON-Assistent bezeichnet.

- DFHLS2JS ordnet Strukturen höherer Programmiersprachen JSON-Schemas zu.
- DFHJS2LS ordnet JSON-Schemas Strukturen höherer Programmiersprachen zu.

Die beiden Zuordnungen sind nicht symmetrisch:

- Wenn Sie eine Sprachdatenstruktur mit DFHLS2JS verarbeiten und anschließend das resultierende JSON-Schema mit DFHJS2LS verarbeiten, stimmt die finale Datenstruktur wahrscheinlich nicht mit dem Original überein.
- Wenn Sie ein JSON-Schema mit DFHJS2LS verarbeiten und anschließend die resultierende Sprachstruktur mit DFHLS2JS verarbeiten, stimmt das finale JSON-Schema wahrscheinlich nicht mit dem Original überein.
- In manchen Fällen generiert DFHJS2LS Sprachstrukturen, die von DFHLS2JS nicht unterstützt werden.

Sie müssen Strukturen höherer Programmiersprachen, die von DFHLS2JS verarbeitet werden, entsprechend den Regeln der Sprache codieren, die in den von CICS unterstützten Sprachcompilern implementiert sind.

Zuordnung von COBOL zu JSON-Schema:

Das Dienstprogramm DFHLS2JS unterstützt Zuordnungen zwischen COBOL-Datenstrukturen und JSON-Schemadefinitionen.

COBOL-Namen werden entsprechend den folgenden Regeln in JSON-Namen konvertiert:

- Doppelte Namen werden durch Hinzufügen einer oder mehrerer Ziffern eindeutig gemacht.
Zwei Instanzen von year werden zu year und year1.
- Bindestriche werden durch Unterstriche ersetzt. Aufeinanderfolgende Bindestriche werden durch aufeinanderfolgende Unterstriche ersetzt.
Beispiel: current-user--id wird zu current_user__id.
- Segmente von Namen, die durch Bindestriche begrenzt sind und nur Großbuchstaben enthalten, werden in Kleinbuchstaben konvertiert.
Beispiel: CA-REQUEST-ID wird zu ca_request_id.
- Ein Unterstrich wird vor Namen hinzugefügt, die mit einer Ziffer beginnen.
Beispiel: 9A-REQUEST-ID wird zu _9a_request_id.

CICS ordnet COBOL-Datenbeschreibungselemente entsprechend der folgenden Tabelle Schemaelementen zu. COBOL-Datenbeschreibungselemente, die in der Tabelle nicht enthalten sind, werden von DFHLS2JS nicht unterstützt. Es gelten außerdem die folgenden Einschränkungen:

- Datenbeschreibungselemente mit den Ebenennummern 66 und 77 werden nicht unterstützt. Datenbeschreibungselemente mit der Ebenennummer 88 werden ignoriert.
- Die folgenden Klauseln in Datenbeschreibungseinträgen werden nicht unterstützt:
 - REDEFINES
 - RENAMES; d. h. Ebene 66
 - DATE FORMAT
- Die folgenden Klauseln in Datenbeschreibungselementen werden ignoriert:
 - BLANK WHEN ZERO
 - JUSTIFIED
 - VALUE
- Die SIGN-Klausel SIGN TRAILING wird unterstützt. Die SIGN-Klausel SIGN LEADING wird nur unterstützt, wenn in DFHLS2JS eine Zuordnungsebene 1.2 oder höher angegeben ist.
- SEPARATE CHARACTER wird sowohl für SIGN TRAILING- als auch für SIGN LEADING-Klauseln auf Zuordnungsebene 1.2 oder höher unterstützt.
- Die folgenden Phrasen in der USAGE-Klausel werden nicht unterstützt:
 - OBJECT REFERENCE
 - POINTER
 - FUNCTION-POINTER
 - PROCEDURE-POINTER
- Die folgenden Phrasen in der USAGE-Klausel werden auf Zuordnungsebene 1.2 oder höher unterstützt:
 - COMPUTATIONAL-1
 - COMPUTATIONAL-2
- Die einzigen PICTURE-Zeichen, die für DISPLAY- und COMPUTATIONAL-5-Datenbeschreibungselemente unterstützt werden, sind 9, S und Z.
- Die PICTURE-Zeichen, die für PACKED-DECIMAL-Datenbeschreibungselemente unterstützt werden, sind 9, S, V und Z.
- Die einzigen PICTURE-Zeichen, die für bearbeitete numerische Datenbeschreibungselemente unterstützt werden, sind 9 und Z.

- Wenn der Parameter MAPPING-LEVEL auf 1.2 oder höher und der Parameter CHAR-VARYING auf NULL festgelegt ist, werden Zeichenarrays einem string-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet.
- Wenn der Parameter MAPPING-LEVEL auf 1.2 oder höher und der Parameter CHAR-VARYING auf BINARY festgelegt ist, werden Zeichenarrays einem string-Element zugeordnet und als binäre Daten verarbeitet.
- Wenn der Parameter MAPPING-LEVEL auf 1.2 oder höher und der Parameter CHAR-VARYING auf COLLAPSE festgelegt ist, werden Leerzeichen am Ende von Zeichenfolgen ignoriert.
- Die Klausel OCCURS DEPENDING ON wird auf Zuordnungsebene 4.0 oder höher unterstützt. Eine komplexe Klausel OCCURS DEPENDING ON wird nicht unterstützt. Dies bedeutet, dass OCCURS DEPENDING ON nur für das letzte Feld einer Struktur unterstützt wird.
- Die Klausel OCCURS INDEXED BY wird auf jeder Zuordnungsebene unterstützt.

COBOL-Datenbeschreibung	JSON-Schemadefinition
<pre> PIC X(n) PIC A(n) PIC G(n) DISPLAY-1 PIC N(n) </pre>	<pre> "type": "string", "maxLength": n </pre>

COBOL-Datenbeschreibung	JSON-Schemadefinition
<pre> PIC S9 DISPLAY PIC S99 DISPLAY PIC S999 DISPLAY PIC S9999 DISPLAY PIC S9(<i>n</i>) DISPLAY PIC S9(<i>n</i>) COMP PIC S9(<i>n</i>) COMP-4 PIC S9(<i>n</i>) COMP-5 PIC S9(<i>n</i>) BINARY </pre>	<pre> "type":"integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i> </pre> <p>Dabei ist <i>n</i> der Höchstwert, der durch ein Muster von '9'-Zeichen dargestellt werden kann.</p>
<pre> PIC 9 DISPLAY PIC 99 DISPLAY PIC 999 DISPLAY PIC 9999 DISPLAY PIC 9(<i>n</i>) DISPLAY PIC 9(<i>n</i>) COMP PIC 9(<i>n</i>) COMP-4 PIC 9(<i>n</i>) COMP-5 PIC 9(<i>n</i>) BINARY </pre>	<pre> "type":"integer", "minimum":0, "maximum": <i>n</i> </pre> <p>Dabei ist <i>n</i> der Höchstwert, der durch ein Muster von '9'-Zeichen dargestellt werden kann.</p>

COBOL-Datenbeschreibung	JSON-Schemadefinition
PIC S9(<i>m</i>)V9(<i>n</i>) COMP-3	<pre>"type": "number", "format": "decimal", "minimum": x , "maximum": y , "multipleOf": z</pre> <p>Dabei gilt Folgendes:</p> <p><i>x</i> ist der Mindestwert, der durch das Muster von '9'-Zeichen dargestellt werden kann.</p> <p><i>y</i> ist der Höchstwert, der durch das Muster von '9'-Zeichen dargestellt werden kann.</p> <p><i>z</i> ist die kleinste verfügbare Einheit = 1/10 ⁿ.</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>Dabei gilt Folgendes:</p> <p><i>y</i> ist der Höchstwert, der durch das Muster von '9'-Zeichen dargestellt werden kann.</p> <p><i>z</i> ist die kleinste verfügbare Einheit = 1/10 ⁿ.</p>
PIC S9(15) COMP-3 Unterstützt auf Zuordnungsebene 3.0, wenn DATETIME=PACKED15	<pre>"type": "string", "format": "date-time"</pre> <p>Das Format der Zeitmarke wird durch RFC3339 definiert.</p>
PIC S9(<i>m</i>)V9(<i>n</i>) DISPLAY Unterstützt auf Zuordnungsebene 1.2 und höher	<pre>"type": "number", "format": "decimal", "minimum": x , "maximum": y , "multipleOf": z</pre> <p>Dabei gilt Folgendes:</p> <p><i>x</i> ist der Mindestwert, der durch das Muster von '9'-Zeichen dargestellt werden kann.</p> <p><i>y</i> ist der Höchstwert, der durch das Muster von '9'-Zeichen dargestellt werden kann.</p> <p><i>z</i> ist die kleinste verfügbare Einheit = 1/10 ⁿ.</p>

COBOL-Datenbeschreibung	JSON-Schemadefinition
<p>COMP-1</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert.</p> <p>Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-1-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre>"type": "number", "format": "float"</pre>
<p>COMP-2</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert.</p> <p>Einige extrem große oder kleine Werte sind für doppelte Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-2-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden. .</p>	<pre>"type": "number", "format": "double"</pre>

COBOL-Datenbeschreibung	JSON-Schemadefinition
<p><i>datenbeschreibung</i> OCCURS <i>n</i> TIMES</p>	<p>Auf Zuordnungsebene 4.0 und höher</p> <p>Für Basiselemente:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { "type": "object", "properties": { name : { datenbeschreibung_JSON } } "required": [name] }</pre> <p>Für Strukturen:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { datenbeschreibung_JSON }</pre> <p>Dabei ist <i>datenbeschreibung_JSON</i> die JSON-Schemadarstellung der <i>datenbeschreibung</i> in COBOL und <i>name</i> der Name der <i>datenbeschreibung</i> in COBOL.</p> <p>Auf Zuordnungsebene 4.1 und höher</p> <p>TRUNCATE-NULL-ARRAYS = DISABLED</p> <p>Sowohl strukturierte als auch Basisarrays werden wie folgt zugeordnet.</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { datenbeschreibung_JSON }</pre> <p>TRUNCATE-NULL-ARRAYS = ENABLED</p> <p>Basisarrays werden wie oben gezeigt zugeordnet, und strukturierte Arrays wie folgt.</p> <pre> "type": "array" "maxItems": n "minItems": 0 "items": { datenbeschreibung_JSON }</pre>

COBOL-Datenbeschreibung	JSON-Schemadefinition
<i>datenbeschreibung</i> OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDENT ON <i>t</i> Unterstützt auf Zuordnungsebene 4.0	<i>"field-name"</i> : { "array": "array" , "maxItems": <i>m</i> "minItems": <i>n</i> "items":{ ... } } Der Inhalt des Arrayelements hängt vom verwendeten Datentyp ab.
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	Wenn CHAR-OCCURS =STRING: <i>"field-name"</i> : { "string": "string" , "maxLength": <i>n</i> } Dies ist eine Zeichenfolge.

COBOL-Datenbeschreibung	JSON-Schemadefinition
	<p>Wenn CHAR-OCCURS =ARRAY:</p> <p>Auf Zuordnungsebene 4.0 und höher</p> <pre> "field-name" :{ "maxItems": m , "minItems": n , "items":{ "type": "object" , "properties":{ "field-name":{ "type": "string" , "maxLength":1 } }, "required":["field-name"] } } </pre> <p>Dies ist ein Array einzelner Zeichen.</p> <p>Auf Zuordnungsebene 4.1 und höher</p> <p>TRUNCATE-NULL-ARRAYS = <i>DISABLED</i></p> <p>Sowohl strukturierte als auch Basisarrays werden wie folgt zugeordnet.</p> <pre> "type":"array" "maxItems": n "minItems": n "items":{ datenbeschreibung_JSON } </pre> <p>TRUNCATE-NULL-ARRAYS = <i>ENABLED</i></p> <p>Basisarrays werden wie oben gezeigt zugeordnet, und strukturierte Arrays wie folgt.</p> <pre> "type":"array" "maxItems": n "minItems": 0 "items":{ datenbeschreibung_JSON } </pre>

COBOL-Datenbeschreibung	JSON-Schemadefinition
<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>Wenn CHAR-OCCURS =STRING:</p> <pre> "field-name" :{ "type": "string" , "maxLength": m "minLength": n } </pre>
<pre> PIC N(n) USAGE NATIONAL </pre> <p>Wenn CHAR-USAGE =NATIONAL: PIC N(n)</p>	<p>Auf Zuordnungsebene 4.0 und höher:</p> <pre> "type":"string", "maxLength": n </pre> <p>Zur Laufzeit füllt CICS das Anwendungsdatenstrukturfeld mit UTF-16-Daten.</p>

Zuordnung von JSON-Schema zu COBOL:

Das Dienstprogramm DFHJS2LS unterstützt Zuordnungen zwischen JSON-Schemas und COBOL-Datenstrukturen.

Die CICS-Assistenten generieren eindeutige, gültige Feldnamen für COBOL-Variablen aus den Schemaelementnamen mithilfe der folgenden Regeln:

1. Reservierten COBOL-Wörtern wird das Präfix X vorangestellt.
Beispiel: DISPLAY wird zu XDISPLAY.
2. Alle Zeichen außer A-Z, a-z, 0-9 oder Bindestrich werden durch X ersetzt.
Beispiel: monthly_total wird zu monthlyXtotal.
3. Wenn das letzte Zeichen ein Bindestrich ist, wird es durch X ersetzt.
Beispiel: ca-request- wird zu ca-requestX.

4. Doppelte Namen in demselben Bereich werden durch das Hinzufügen von ein oder zwei Ziffern zu der zweiten Instanz und zu allen nachfolgenden Instanzen des Namens eindeutig gemacht.

Beispiel: Drei Instanzen von year werden zur year , year1 und year2.

Sollte das oben beschriebene Verhalten unerwünscht sein, kann der Benutzer **MAPPING-OVERRIDES=NO-ARRAY-NAME-INDEXING** als Eingabe für das Dienstprogramm angeben, die das Hinzufügen einer oder mehrerer Ziffern zur zweiten Instanz und allen nachfolgenden Instanzen des Namens inaktiviert.

5. Ein JSON-Schema gibt an, dass eine Variable eine variierende Kardinalität hat, wenn das Element "type" den Wert "array" hat und wenn die Schlüsselwörter "minItems" und "maxItems" nicht angegeben sind oder abweichende Werte aufweisen. Wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat, werden Feldnamen mit den Suffixes "_cont" und "_num" erstellt.

Weitere Informationen finden Sie unter „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.

6. Ein JSON-Schema gibt an, dass eine Variable optional ist, wenn sie nicht in dem "required"-Schlüsselwortarray enthalten ist, das dem einschließenden JSON-Schema vom Typ "object" zugeordnet ist. Bei optionalen Feldern wird ein zusätzliches Feld mit dem Suffix _num zu dem Elementnamen hinzugefügt. Zur Laufzeit ist dies gleich null, um anzugeben, dass der Wert in den JSON-Daten gefehlt hat, und ungleich null, wenn der Wert in den JSON-Daten vorhanden war.
7. Feldnamen sind auf 28 Zeichen begrenzt. Wenn ein generierter Name, einschließlich des Präfix und Suffix, diese Länge überschreitet, wird der Elementname abgeschnitten.

DFHJS2LS ordnet Schematypen zu COBOL-Datenbeschreibungselementen mithilfe der angegebenen Zuordnungsebene in der folgenden Tabelle zu. Beachten Sie die folgenden Punkte:

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL gesetzt ist, werden Zeichendaten mit variabler Länge auf null endenden Zeichenfolgen zugeordnet, und ein zusätzliches Zeichen wird dem Nullabschlusszeichen zugeordnet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf YES gesetzt ist, werden Zeichendaten mit variabler Länge zwei verwandten Elementen zugeordnet: einem Längenfeld und einem Datenfeld.

Beispiel:

```
"textString": {  
  "type": "string",  
  "maxLength": 10000,  
  "minLength": 1  
}
```

Zuordnung zu:

```
15 textString-length PIC S9999 COMP-5 SYNC  
15 textString PIC X(10000)
```

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
Alle: "type": "null" "type": []	Nicht unterstützt

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
"\$schema": "http://json-schema.org/draft-04/schema#"	Dieses Schlüsselwort wird ignoriert, es wird jedoch davon ausgegangen, dass es mit der JSON-Schemaspezifikation 'draft-04' kompatibel ist.
"title": "same text" "description": "more text"	Diese Schlüsselwörter werden ignoriert.
"format": "<vordefinierte werte>"	Das Schlüsselwort "format" wird verwendet, um entweder die generierte Struktur oder den Laufzeitwert zu ändern. Lesen Sie die folgenden Informationen zur unterstützten Verwendung von "format".
"type": "array", "items": {<JSON-unterschema>}, "additionalItems": false, "maxItems": m , "minItems": n	Mehrdimensionale Arrays werden ab Zuordnungsebene 4.3 aufwärts unterstützt, und Arrays mit gemischtem Typ werden nicht unterstützt. "additionalItems" wird als falsch angenommen und es wird kein anderer Wert unterstützt. Wenn sowohl "minItems" als auch "maxItems" vorhanden und gleichwertig sind, wird das Array als feste Kardinalität behandelt. Ist dies nicht der Fall, wird es als variierende Kardinalität behandelt. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.
"type": "array", "uniqueItems": true	"uniqueItems" wird nicht mit JSON-Arrays unterstützt.
"type": "object", "additionalProperties": false, "properties": { ["<elementname>": {<JSON-unterschema> } [,]]* } "required": [["<elementname>" [,]]*]	Das einzige JSON-Objekt, das derzeit unterstützt wird, ist eine feste Gruppe benannter Elemente. Dieses generiert eine Struktur (oder Unterstruktur), die diese Elementnamen verwendet. Für "additionalProperties" wird false angenommen, wenn kein Wert im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist. Alle Elemente im Objekt "properties" werden als "optional" angenommen, wenn sie sich nicht im Array "required" befinden oder wenn kein Array "required" vorhanden ist. Ein optionales Element ("optional") erhält eine variable Ordinalität von null bis X. Dabei ist X entweder 1 oder die maximale Anzahl von Elementen im Array, vorausgesetzt, das Element ist als Array definiert. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
<pre>"type": "object", "additionalProperties": true</pre>	<p>Das Objekt wird wie im vorherigen Beispiel zugeordnet, mit zusätzlichen Feldern, die zusätzliche Eigenschaften unterstützen. Für die Eigenschaft <code>additionalProperties</code> wird 'false' angenommen, wenn sie nicht im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist.</p> <p>Ist diese Option aktiviert, werden bis zu dem im Parameter ADDITIONAL-PROPERTIES-MAX angegebenen Wert Leerzeichen zugeordnet. Die Anzahl der Zeichen in jedem Leerzeichen wird durch den Parameter ADDITIONAL-PROPERTIES-SIZE festgelegt. Jede Eigenschaft wird einem Feld PIC X(z) zugeordnet, wobei z durch den Parameter ADDITIONAL-PROPERTIES-SIZE definiert wird. Ist der Wert von ADDITIONAL-PROPERTIES-MAX größer als 0, werden die Eigenschaften in ähnlicher Weise zugeordnet wie in einem Array, in dem maxItems festgelegt ist.</p> <p>Anmerkung: Es gibt mehrere Möglichkeiten, JSON-Unterstützung in CICS zu konfigurieren, einschließlich der Verwendung von z/OS Connect for CICS. Wenn Sie die ältere CICS-Java-Pipeline-Technologie nutzen, werden Additional Properties nur unterstützt, falls die <code>com.ibm.cics.json.enableAxis2Handlers-JVM</code>-Systemeigenschaft nicht auf 'true' festgelegt ist.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>Keines dieser Schlüsselwörter wird mit JSON-Objekten unterstützt.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression", "minLength": l</pre>	<p>PIC X(z)</p> <p>Dabei basiert der Wert von z auf m , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p>m basiert auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>"pattern"- und "minLength"-Einschränkungen werden über die Sprachstruktur nur in Form eines Kommentars übergeben.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>PIC N(z) USAGE NATIONAL</p> <p>Dabei basiert der Wert von z auf m , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p>m basiert auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*": { "type": "string", "format": "date-time" }</pre>	<p>PIC S9(15) COMP-3</p> <p>Alle unterstützt, wenn DATETIME=PACKED15</p> <p>Beachten Sie, dass "maxLength" und "minLength" nicht für dieses Format unterstützt werden.</p>

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
<pre>"*name*":{ "type":"string", "format":"uri" }</pre>	<p>PIC X(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>PIC N(<i>m</i>) USAGE NATIONAL</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*":{ "type":"string", "format":"base64Binary" }</pre>	<p>PIC X(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"hexBinary" }</pre>	<p>PIC X(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"<vordefiniert>" }</pre>	<p>PIC X(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i> . Ein relevantes Muster ("pattern") wird verwendet und an den Kommentar übergeben.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>PIC N(<i>m</i>) USAGE NATIONAL</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i> . Ein relevantes Muster ("pattern") wird verwendet und an den Kommentar übergeben.</p>
<pre>"type":"boolean"</pre>	<p>PIC X DISPLAY</p> <p>Der Wert x'00' impliziert 'false', x'01' impliziert 'true'.</p>
<pre>"type": "integer", "exclusiveMaximum": true, "exclusiveMinimum": true, "multipleOf": n</pre>	<p>"exclusiveMaximum"- und "exclusiveMinimum"-Einschränkungen werden nur in Form eines Kommentars an die Sprachstruktur übergeben.</p> <p>"multipleOf" wird ignoriert.</p>

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
"type"="integer", minimum=0, maximum=255	Zuordnungsebene 3.0 und niedriger: PIC X DISPLAY Zuordnungsebene 4.0 und höher (oder wenn der Parameter INTEGER-AS-PIC9 angegeben wurde, unabhängig von der Zuordnungsebene): PIC 9(z) COMP-5 SYNC oder PIC 9(z) DISPLAY Dabei ist $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:-128, maximum:127	Zuordnungsebene 3.0 und niedriger: PIC X DISPLAY Zuordnungsebene 4.0 und höher (oder wenn der Parameter INTEGER-AS-PIC9 angegeben wurde, unabhängig von der Zuordnungsebene): PIC S9(z) COMP-5 SYNC oder PIC S9(z) DISPLAY Dabei ist $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:0, maximum; <i>m</i>	PIC 9(z) COMP-5 SYNC oder PIC 9(z) DISPLAY Dabei ist $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:- <i>m</i> , maximum: <i>m</i> -1	PIC S9(z) COMP-5 SYNC oder PIC S9(z) DISPLAY Dabei ist $10^{(z-1)} < m \leq 10^z$
"type": "number", "maximum": m, "minimum": n, "exclusiveMaximum": true, "exclusiveMinimum": true, "multipleOf": n	"maximum"-, "minimum"-, "exclusiveMaximum"- und "exclusiveMinimum"-Einschränkungen werden nur in Form eines Kommentars an die Sprachstruktur übergeben. "multipleOf" wird ignoriert.
"type": "number" "format": "decimal"	PIC 9(p)V9(n) COMP-3 Dabei sind <i>p</i> und <i>n</i> Standardwerte.

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
<pre>"type": "number" "format": "float"</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <ul style="list-style-type: none"> • PIC X(32) <p>Zuordnungsebene 1.2 und höher:</p> <ul style="list-style-type: none"> • COMP-1 <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert.</p> <p>Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-1-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>
<pre>"type": "number" "format": "double"</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <ul style="list-style-type: none"> • PIC X(32) <p>Zuordnungsebene 1.2 und höher:</p> <ul style="list-style-type: none"> • COMP-2 <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert.</p> <p>Einige extrem große oder kleine Werte sind für doppelte Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-2-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden. .</p>

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
<pre> oneOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] anyOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] allOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] </pre>	<p>Die logischen Pfade durch das Array von Optionen werden so zusammengeführt, als ob ein einzelnes Objekt vom Typ { "properties": { "A":..., "B":... } } definiert wäre. Jede dieser JSON-Eigenschaften wird als optional behandelt. Wenn die Unteroptionen widersprüchliche Definitionen für denselben Eigenschaftsnamen enthalten, wird eine Fehlermeldung ausgegeben. Wenn eine einzelne Eigenschaft beispielsweise sowohl als Zeichenfolge (in Pfad A) als auch als Objekt (in Pfad B) definiert ist, wird diese Definition nicht unterstützt und führt zu einer Fehlermeldung.</p> <p>Es ist möglich, dass ein JSON-Schema komplexe logische Strukturen definiert. Die in den komplexen logischen Strukturen implizierten Feinheiten können jedoch bei der Zuordnung zu einer Sprachstruktur verloren gehen. Der Umsetzungsprozess versucht nicht, die kombinatorischen Regeln aus dem Schema umzusetzen. Es interagiert nur mit den Sprachstrukturfeldern, die angeben, ob eine bestimmte JSON-Eigenschaft vorhanden oder nicht vorhanden ist.</p> <p>Wenn die Unteroptionen kompatible Definitionen für denselben Eigenschaftsnamen enthalten, versucht DFHJS2LS, das zugeordnete Muster von Einschränkungen zusammenzuführen, wobei allerdings einige Feinheiten verloren gehen können. Nehmen wir beispielsweise die folgende Definition an:</p> <pre> "A": { "oneOf": [{ "type": "string", "maxLength": 5 }, { "type": "string", "minLength": 7, "maxLength": 8 }] } </pre> <p>„A“ ist entweder als eine Zeichenfolge mit einer Länge von bis zu 5 Zeichen oder als eine Zeichenfolge mit einer Länge zwischen 7 und 8 Zeichen definiert. Die Zusammenführung kann zu Zuordnungen zu einer Zeichenfolge mit einer Länge von 0 bis 8 Zeichen führen, ohne dass erkannt wird, dass die Länge von 6 Zeichen ungültig ist.</p> <p>Die meisten Szenarios mit logischer Zusammensetzung werden einfachen Sprachstrukturen zugeordnet, aber bei einer komplizierten logischen Zusammensetzung werden während des Zuordnungsprozesses möglicherweise Kompromisse geschlossen. Die besten Ergebnisse erhalten Sie, wenn Sie die logische Zusammensetzung in einem JSON-Schema zum Definieren alternativer Deklarationen für dieselbe JSON-Eigenschaft vermeiden.</p>

Anmerkung: CICS kann keine ganzzahligen Werte größer als den maximalen Wert für 'signed long' ($2^{63} - 1$) umsetzen, es sei denn, sie sind in Anführungszeichen eingeschlossen.

Anmerkung: Minimale und maximale Werte, die in dem Schema für numerische Typen angegeben sind, werden nur zum Zuordnen zu einem COBOL-Datentyp verwendet. Die Daten werden während der Laufzeit nicht gegen diese Werte geprüft.

Einige der Schematypen, die in der Tabelle aufgeführt sind, werden dem COBOL-Format COMP-5 SYNC oder DISPLAY zugeordnet, abhängig von den Werten, die ggf. in den Schlüsselwörtern minimum und maximum angegeben sind:

- Für Typen mit Vorzeichen (short , int und long) wird DISPLAY verwendet, wenn Folgendes angegeben ist:

```
"maximum":  
a  
"minimum":  
-a
```

Dabei ist *a* eine aus Neunen bestehende Zeichenfolge.

- Für Typen ohne Vorzeichen (unsignedShort , unsignedInt und unsignedLong) wird DISPLAY verwendet, wenn Folgendes angegeben ist:

```
"maximum":  
a  
"minimum":0
```

Dabei ist *a* eine aus Neunen bestehende Zeichenfolge.

Ist ein anderer Wert oder kein Wert angegeben, wird COMP-5 SYNC verwendet.

Zuordnung von C und C++ zu JSON-Schema:

Das Dienstprogramm DFHLS2JS unterstützt Zuordnungen zwischen C- und C++-Datentypen mit JSON-Schemadefinitionen.

C- und C++-Namen werden entsprechend den folgenden Regeln in JSON-Namen konvertiert:

1. Zeichen, die in JSON-Eigenschaftsnamen nicht gültig sind, werden durch X ersetzt.

Beispiel: monthly-total wird zu monthlyXtotal.

2. Doppelte Namen werden durch Hinzufügen einer oder mehrerer Ziffern eindeutig gemacht.

Zwei Instanzen von year werden zu year und year1.

DFHLS2JS ordnet C- und C++-Datentypen Schemaelementen entsprechend der folgenden Tabelle zu. C- und C++-Typen, die in der Tabelle nicht enthalten sind, werden von DFHLS2JS nicht unterstützt. Das Qualifikationsmerkmal _Packed wird für Strukturen unterstützt. Folgende Einschränkungen gelten:

- Headerdateien müssen eine struct-Instanz der höchsten Ebene enthalten.
- Sie können keinen Strukturtyp deklarieren, der sich selbst als Mitglied enthält.
- Die folgenden C- und C++-Datentypen werden nicht unterstützt:

```
decimal  
long double  
wchar_t (nur C++)
```

- Folgendes wird ignoriert, wenn es in der Headerdatei vorhanden ist.

Speicherklassenkennungen:

```
auto  
register  
static  
extern  
mutable
```

Qualifikationsmerkmale

```
const
```

volatile
_Export (nur C++)
Funktionskennungen
inline (nur C++)
virtual (nur C++)

Anfangswerte

- Die Headerdatei darf die folgenden Werte nicht enthalten:
 - Unionen
 - Klassendeklarationen
 - Aufzählungsdatentypen
 - Zeigertypvariablen
 - Schablonendeklarationen
 - Vordefinierte Makros, d. h. Makros mit Namen, die mit zwei Unterstrichen beginnen und enden (__)
 - Die Zeilenfortsetzungssequenz (ein '\'-Symbol, auf das unmittelbar ein Zeilenvorschubzeichen folgt)
 - Deklaratoren für Prototypfunktionen
 - Vorprozessoranweisungen
 - Bitfelder
 - Das Schlüsselwort __cdecl (oder _cdecl) (nur C++)
- Der Anwendungsprogrammierer muss einen 32-Bit-Compiler verwenden, um sicherzustellen, dass einem int-Wert 4 Bytes zugeordnet werden.
- Die folgenden C++-reservierten Schlüsselwörter werden nicht unterstützt:
 - explicit
 - using
 - namespace
 - typename
 - typeid
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL festgelegt ist, werden Zeichenarrays einem string-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf BINARY festgelegt ist, werden Zeichenarrays einem xsd:base64Binary-Element zugeordnet und als binäre Daten verarbeitet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf COLLAPSE festgelegt ist, wird <xsd:whiteSpace value="collapse"/> für Zeichenfolgen generiert.

C- und C++-Datentyp	Schema 'simpleType'
char[z]	"type":"string" "maxlength": z
char16_t[n]	Auf Zuordnungsebene 4.0 und höher: "type":"string" "maxlength": n Zur Laufzeit füllt CICS das Anwendungsdatenstrukturfeld mit UTF-16-Daten.

C- und C++-Datentyp	Schema 'simpleType'
char[8] Unterstützt auf Zuordnungsebene 3.0 und höher, wenn DATETIME=PACKED15	"type":"string" "format":"date-time" Das Format der Zeitmarke wird durch RFC3339 definiert.
char short int long long long	"type":"integer", "minimum":- (n + 1), "maximum": n Dabei ist n der Maximalwert, der durch das Basiselement dargestellt werden kann.
unsigned char unsigned short unsigned int unsigned long unsigned long long	"type":"integer", "minimum":0, "maximum": n Dabei ist n der Maximalwert, der durch das Basiselement dargestellt werden kann.
bool (nur C++)	"type":"boolean"
float Unterstützt auf Zuordnungsebene 1.2 und höher.	"type":"number", "format":"float" Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt Gleitkommatentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.
double Unterstützt auf Zuordnungsebene 1.2 und höher.	"type":"number", "format":"double" Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für doppelte Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt doppelter Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.

C- und C++-Datentyp	Schema 'simpleType'
<i>typename</i> [<i>n</i>]	<p>Für Basiselemente:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { "type": "object", "properties": { name : { typ_JSON } } "required": [name] } </pre> <p>Für Structs:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { typ_JSON } </pre> <p>Dabei ist <i>typ_JSON</i> die JSON-Schemadarstellung des C- oder C++-Typs.</p>

Zuordnung von JSON-Schema zu C und C++:

Die DFHJS2LS-Dienstprogramme unterstützen Zuordnungen zwischen den JSON-Schemas und C- und C++-Datentypen.

Die CICS-Assistenten generieren eindeutige, gültige Feldnamen für C- und C++-Variablen aus den Schemaelementnamen mithilfe der folgenden Regeln:

1. Alle Zeichen außer A-Z, a-z, 0-9 oder _ werden durch X ersetzt.
Beispiel: `monthly-total` wird zu `monthlyXtotal`.
2. Wenn das erste Zeichen kein alphabetisches Zeichen ist, wird es durch ein führendes X ersetzt.
Beispiel: `_monthlysummary` wird zu `Xmonthlysummary`.
3. Doppelte Namen in demselben Bereich werden durch das Hinzufügen von ein oder zwei Ziffern zu der zweiten Instanz und zu allen nachfolgenden Instanzen des Namens eindeutig gemacht.
Beispiel: Drei Instanzen von `year` werden zu `year`, `year1` und `year2`.
4. Ein JSON-Schema gibt an, dass eine Variable eine variierende Kardinalität hat, wenn das Element "type" den Wert "array" hat und wenn die Schlüsselwörter "minItems" und "maxItems" nicht angegeben sind oder abweichende Werte aufweisen. Wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat, werden Feldnamen mit den Suffixes "_cont" und "_num" erstellt.
Weitere Informationen finden Sie unter „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.

5. Ein JSON-Schema gibt an, dass eine Variable optional ist, wenn sie nicht in dem "required"-Schlüsselwortarray enthalten ist, das dem einschließenden JSON-Schema vom Typ "object" zugeordnet ist. Bei optionalen Feldern wird ein zusätzliches Feld mit dem Suffix `_num` zu dem Elementnamen hinzugefügt. Zur Laufzeit ist dies gleich null, um anzugeben, dass der Wert in den JSON-Daten gefehlt hat, und ungleich null, wenn der Wert in den JSON-Daten vorhanden war.
6. Feldnamen sind auf 50 Zeichen begrenzt. Wenn ein generierter Name, einschließlich aller Präfixe und Suffixe, diese Länge überschreitet, wird der Elementname abgeschnitten.

DFHJS2LS ordnet JSON-Schematypwerte C- und C++-Datentypen gemäß der folgenden Tabelle zu. Es gelten außerdem die folgenden Regeln:

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL gesetzt ist, werden Zeichendaten mit variabler Länge auf null endenden Zeichenfolgen zugeordnet, und ein zusätzliches Zeichen wird dem Nullabschlusszeichen zugeordnet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf YES gesetzt ist, werden Zeichendaten mit variabler Länge zwei verwandten Elementen zugeordnet: einem Längenfeld und einem Datenfeld.

JSON-Schemaschlüsselwort	C- und C++-Datentyp
Alle: "type": "null" "type": []	Nicht unterstützt
"\$schema": "http://json-schema.org/draft-04/schema#"	Dieses Schlüsselwort wird ignoriert, es wird jedoch davon ausgegangen, dass es mit der JSON-Schemaspezifikation 'draft-04' kompatibel ist.
"title": "same text" "description": "more text"	Diese Schlüsselwörter werden ignoriert.
"format": "<vordefinierte werte>"	Das Schlüsselwort "format" wird verwendet, um entweder die generierte Struktur oder den Laufzeitwert zu ändern. Lesen Sie die folgenden Informationen zur unterstützten Verwendung von "format".
<pre> "type": "array", "items": {<JSON-unterschema>}, "additionalItems": false, "maxItems": m , "minItems": n </pre>	<p>Mehrdimensionale Arrays werden ab Zuordnungsebene 4.3 aufwärts unterstützt, und Arrays mit gemischtem Typ werden nicht unterstützt.</p> <p>"additionalItems" wird als 'false' angenommen und es wird kein anderer Wert unterstützt.</p> <p>Wenn sowohl "minItems" als auch "maxItems" vorhanden und gleichwertig sind, wird das Array als feste Kardinalität behandelt. Ist dies nicht der Fall, wird es als variierende Kardinalität behandelt. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.</p>

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre>"type": "array", "uniqueItems": true</pre>	<p>"uniqueItems" wird mit JSON-Arrays nicht unterstützt. Das <JSON-unterschema> muss einen unterstützten Typ ("type") definieren, aber bei diesem Typ kann es sich nicht um ein Array ("array") handeln. Dies ist eine Einschränkung für die generierte Sprachstruktur.</p>
<pre>"type": "object", "additionalProperties": false, "properties": { ["<elementname>": {<JSON-unterschema> } [,]]* } "required": [["<elementname>" [,]]*]</pre>	<p>Das einzige JSON-Objekt, das derzeit unterstützt wird, ist eine feste Gruppe benannter Elemente.</p> <p>Dieses generiert eine Struktur (oder Unterstruktur), die diese Elementnamen verwendet.</p> <p>Für "additionalProperties" wird false angenommen, wenn kein Wert im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist.</p> <p>Alle Elemente im Objekt "properties" werden als "optional" angenommen, wenn sie sich nicht im Array "required" befinden oder wenn kein Array "required" vorhanden ist. Ein optionales Element ("optional") erhält eine variable Ordinalität von null bis X. Dabei ist X entweder 1 oder die maximale Anzahl von Elementen im Array, vorausgesetzt, das Element ist als Array definiert. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.</p>

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre>"type": "object", "additionalProperties": true</pre>	<p>Das Objekt wird wie im vorherigen Beispiel zugeordnet, mit zusätzlichen Feldern, die zusätzliche Eigenschaften unterstützen. Für die Eigenschaft <code>additionalProperties</code> wird 'false' angenommen, wenn sie nicht im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist.</p> <p>Ist diese Option aktiviert, werden bis zu dem im Parameter ADDITIONAL-PROPERTIES-MAX angegebenen Wert Leerzeichen zugeordnet. Die Anzahl der Zeichen in jedem Leerzeichen wird durch den Parameter ADDITIONAL-PROPERTIES-SIZE festgelegt. Jede Eigenschaft wird einem Feld PIC X(z) zugeordnet, wobei z durch den Parameter ADDITIONAL-PROPERTIES-SIZE definiert wird. Ist der Wert von ADDITIONAL-PROPERTIES-MAX größer als 0, werden die Eigenschaften in ähnlicher Weise zugeordnet wie in einem Array, in dem maxItems festgelegt ist.</p> <p>Anmerkung: Es gibt mehrere Möglichkeiten, JSON-Unterstützung in CICS zu konfigurieren, einschließlich der Verwendung von z/OS Connect for CICS. Wenn Sie die ältere CICS-Java-Pipeline-Technologie nutzen, werden <code>AdditionalProperties</code> nur unterstützt, falls die <code>com.ibm.cics.json.enableAxis2HandlersJVM-Systemeigenschaft</code> nicht auf 'true' festgelegt ist.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>Keines dieser Schlüsselwörter wird mit JSON-Objekten unterstützt.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression>", "minLength": l</pre>	<p><code>char[z]</code></p> <p>Dabei basiert der Wert von z auf m , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p>m basiert auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>"pattern"- und "minLength"-Einschränkungen werden über die Sprachstruktur nur in Form eines Kommentars übergeben.</p>

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre>"type":"string" "maxLength": <i>m</i></pre>	<p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>char16_t[<i>z</i>]</p> <p>Dabei basiert der Wert von <i>z</i> auf <i>m</i> , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p><i>m</i> basiert auf dem Schlüsselwort "maxLength " und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*":{ "type":"string", "format":"date-time" }</pre>	<p>char[8]</p> <p>Alle unterstützt, wenn DATETIME=PACKED15</p>
<pre>"*name*":{ "type":"string", "format":"uri" }</pre>	<p>char[<i>m</i>]</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>char16_t[<i>m</i>]</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*":{ "type":"string", "format":"base64Binary" }</pre>	<p>char[<i>m</i>]</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"hexBinary" }</pre>	<p>char[<i>m</i>]</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength".</p>

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre>"*name*":{ "type":"string", "format":"<vordefiniert>" }</pre>	<p>char[<i>m</i>]</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i>. Ein relevantes Muster ("pattern") wird verwendet und an den Kommentar übergeben.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>char16_t[<i>m</i>]</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i> . Ein relevantes Muster ("pattern") wird verwendet und an den Kommentar übergeben.</p>
"type":"boolean"	<p>bool (nur C++) short (nur C)</p>
<pre>"type": "integer", "exclusiveMaximum": true, "exclusiveMinimum": true, "multipleOf": n</pre>	<p>"exclusiveMaximum"- und "exclusiveMinimum"-Einschränkungen werden nur in Form eines Kommentars an die Sprachstruktur übergeben.</p> <p>"multipleOf" wird ignoriert.</p>
<pre>"type":"integer", minimum:-128, maximum:127</pre>	signed char
<pre>"type":"integer", minimum:0, maximum:255</pre>	unsigned char
<pre>"type":"integer", minimum:-32768, maximum:32767</pre>	short
<pre>"type":"integer", minimum:0, maximum:65535</pre>	unsigned short
<pre>"type":"integer", minimum:-2147483648, maximum:2147483647</pre>	int
<pre>"type":"integer", minimum:0, maximum:4294967295</pre>	unsigned int
<pre>"type":"integer", minimum:-9223372036854775808, maximum:9223372036854775807</pre>	long long
<pre>"type":"integer", minimum:0, maximum:18446744073709551615</pre>	unsigned long long

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre>"type": "number", "maximum": m, "minimum": n, "exclusiveMaximum": true, "exclusiveMinimum": true, "multipleOf": n</pre>	<p>"maximum"-, "minimum"-, "exclusiveMaximum"- und "exclusiveMinimum"-Einschränkungen werden nur in Form eines Kommentars an die Sprachstruktur übergeben.</p> <p>"multipleOf" wird ignoriert.</p>
<pre>"type": "number" "format": "float"</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <ul style="list-style-type: none"> char[32] <p>Zuordnungsebene 1.2 und höher:</p> <ul style="list-style-type: none"> float(*) <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt Gleitkommatentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>
<pre>"type": "number" "format": "double"</pre>	<p>Zuordnungsebene 1.0 und niedriger:</p> <ul style="list-style-type: none"> char[32] <p>Zuordnungsebene 1.2 und höher:</p> <ul style="list-style-type: none"> double(*) <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für doppelte Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt doppelter Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre> oneOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] anyOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] allOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] </pre>	<p>Die logischen Pfade durch das Array von Optionen werden so zusammengeführt, als ob ein einzelnes Objekt vom Typ { "properties": { "A":..., "B":... } } definiert wäre. Jede dieser JSON-Eigenschaften wird als optional behandelt. Wenn die Unteroptionen widersprüchliche Definitionen für denselben Eigenschaftsnamen enthalten, wird eine Fehlermeldung ausgegeben. Wenn eine einzelne Eigenschaft beispielsweise sowohl als Zeichenfolge (in Pfad A) als auch als Objekt (in Pfad B) definiert ist, wird diese Definition nicht unterstützt und führt zu einer Fehlermeldung.</p> <p>Es ist möglich, dass ein JSON-Schema komplexe logische Strukturen definiert. Die in den komplexen logischen Strukturen implizierten Feinheiten können jedoch bei der Zuordnung zu einer Sprachstruktur verloren gehen. Der Umsetzungsprozess versucht nicht, die kombinatorischen Regeln aus dem Schema umzusetzen. Es interagiert nur mit den Sprachstrukturfeldern, die angeben, ob eine bestimmte JSON-Eigenschaft vorhanden oder nicht vorhanden ist.</p> <p>Wenn die Unteroptionen kompatible Definitionen für denselben Eigenschaftsnamen enthalten, versucht DFHJS2LS, das zugeordnete Muster von Einschränkungen zusammenzuführen, wobei allerdings einige Feinheiten verloren gehen können. Nehmen wir beispielsweise die folgende Definition an:</p> <pre> "A": { "oneOf": [{ "type": "string", "maxLength": 5 }, { "type": "string", "minLength": 7, "maxLength": 8 }] } </pre> <p>„A“ ist entweder als eine Zeichenfolge mit einer Länge von bis zu 5 Zeichen oder als eine Zeichenfolge mit einer Länge zwischen 7 und 8 Zeichen definiert. Die Zusammenführung kann zu Zuordnungen zu einer Zeichenfolge mit einer Länge von 0 bis 8 Zeichen führen, ohne dass erkannt wird, dass die Länge von 6 Zeichen ungültig ist.</p> <p>Die meisten Szenarios mit logischer Zusammensetzung werden einfachen Sprachstrukturen zugeordnet, aber bei einer komplizierten logischen Zusammensetzung werden während des Zuordnungsprozesses möglicherweise Kompromisse geschlossen. Die besten Ergebnisse erhalten Sie, wenn Sie die logische Zusammensetzung in einem JSON-Schema zum Definieren alternativer Deklarationen für dieselbe JSON-Eigenschaft vermeiden.</p>

Anmerkung: CICS kann keine ganzzahligen Werte größer als den maximalen Wert für 'signed long' ($2^{63} - 1$) umsetzen, es sei denn, sie sind in Anführungszeichen eingeschlossen.

Anmerkung: Minimale und maximale Werte, die in dem Schema für numerische Typen angegeben sind, werden nur zum Zuordnen zu einem C- oder C++-Datentyp verwendet. Die Daten werden während der Laufzeit nicht gegen diese Werte geprüft.

Zuordnung von PL/I zu JSON-Schema:

Das Dienstprogramm DFHLS2JS unterstützt Zuordnungen zwischen PL/I-Datenstrukturen und JSON-Schemadefinitionen. Da sich der Enterprise PL/I-Compiler von älteren PL/I-Compilern unterscheidet, werden zwei Sprachoptionen unterstützt: PLI-ENTERPRISE und PLI-OTHER.

PL/I-Namen werden entsprechend den folgenden Regeln in JSON-Namen konvertiert:

1. Zeichen, die in JSON-Eigenschaftsnamen nicht gültig sind, werden durch x ersetzt.
Beispiel: `monthly$total` wird zu `monthlyxtotal`.
2. Doppelte Namen werden durch Hinzufügen einer oder mehrerer Ziffern eindeutig gemacht.
Zwei Instanzen von `year` werden zu `year` und `year1`.

DFHLS2JS ordnet PL/I-Datentypen Schemaelementen entsprechend der folgenden Tabelle zu. PL/I-Typen, die in der Tabelle nicht enthalten sind, werden von DFHLS2JS nicht unterstützt. Es gelten außerdem die folgenden Einschränkungen:

- Datenelemente mit dem Attribut **COMPLEX** werden nicht unterstützt.
- Datenelemente mit dem Attribut **FLOAT** werden auf Zuordnungsebene 1.2 oder höher unterstützt. Enterprise PL/I **FLOAT IEEE** wird nicht unterstützt.
- Die reinen DBCS-Zeichenfolgen **VARYING** und **VARYINGZ** werden auf Zuordnungsebene 1.2 oder höher unterstützt.
- Datenelemente, die als **DECIMAL**(*p* , *q*) angegeben sind, werden nur unterstützt, wenn $p \geq q$.
- Datenelemente, die als **BINARY**(*p* , *q*) angegeben sind, werden nur unterstützt, wenn $q = 0$.
- Wenn das Attribut **PRECISION** für ein Datenelement angegeben ist, wird es ignoriert.
- **PICTURE**-Zeichenfolgen werden nicht unterstützt.
- **ORDINAL**-Datenelemente werden als **FIXED BINARY(7)**-Datentypen behandelt.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf **NULL** festgelegt ist, werden Zeichenarrays einem **string**-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet. Diese Zuordnung gilt nicht für Enterprise PL/I.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf **BINARY** festgelegt ist, werden Zeichenarrays einem **string**-Element zugeordnet und als binäre Daten verarbeitet.

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf COLLAPSE gesetzt ist, werden führende und abschließende Leerzeichen entfernt, und mehrere Leerzeichen werden durch ein einzelnes Leerzeichen ersetzt.

DFHLS2JS implementiert die Padding-Algorithmen von PL/I nicht vollständig, deshalb müssen Sie Padding-Bytes in Ihrer Datenstruktur explizit deklarieren. DFHLS2JS gibt eine Nachricht aus, wenn es erkennt, dass Padding-Bytes fehlen. Jede übergeordnete Struktur muss an einer Doppelwortgrenze beginnen und jedes Byte in der Struktur muss der korrekten Grenze zugeordnet werden. Sehen Sie sich dieses Codefragment an:

```
3 FIELD1 FIXED BINARY(7),
3 FIELD2 FIXED BINARY(31),
3 FIELD3 FIXED BINARY(63);
```

In diesem Beispiel gilt Folgendes:

- FIELD1 ist 1 Byte lang und kann an jeder Grenze ausgerichtet werden.
- FIELD2 ist 4 Bytes lang und muss an einer Vollwortgrenze ausgerichtet werden.
- FIELD3 ist 8 Bytes lang und muss an einer Doppelwortgrenze ausgerichtet werden.

Der Enterprise PL/I-Compiler richtet die Felder in der folgenden Reihenfolge aus:

1. FIELD3 wird zuerst ausgerichtet, weil es die stärksten Anforderungen an die Grenze hat.
2. FIELD2 wird an der Vollwortgrenze unmittelbar vor FIELD3 ausgerichtet.
3. FIELD1 wird an der Bytegrenze unmittelbar vor FIELD3 ausgerichtet.

Schließlich fügt der Compiler drei Padding-Bytes unmittelbar vor FIELD1 ein, so dass die gesamte Struktur an einer Vollwortgrenze ausgerichtet ist.

Da DFHLS2JS keine äquivalenten Padding-Bytes einfügt, müssen Sie diese explizit deklarieren, bevor die Struktur von DFHLS2JS verarbeitet wird. Beispiel:

```
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);
```

Alternativ können Sie die Struktur ändern, um alle Felder als nicht ausgerichtet zu deklarieren und die Anwendung erneut zu kompilieren, die die Struktur verwendet. Weitere Informationen zu den Anforderungen an die strukturelle PL/I-Speicherausrichtung finden Sie unter Enterprise PL/I for z/OS product information.

PL/I-Datenbeschreibung	JSON-Schemadefinition
FIXED BINARY (<i>n</i>)	<pre>"type": "integer", "minimum": - (<i>n</i> + 1), "maximum": <i>n</i></pre> <p>Dabei ist <i>n</i> der Maximalwert, der durch das Basiselement dargestellt werden kann.</p>

PL/I-Datenbeschreibung	JSON-Schemadefinition
UNSIGNED FIXED BINARY(n) Restriction: Nur Enterprise PL/I	<pre>"type":"integer", "minimum":0, "maximum": n</pre> <p>Dabei ist n der Maximalwert, der durch das Basiselement dargestellt werden kann.</p>
FIXED DECIMAL(n , m)	<pre>"type":"number", "format":"decimal", "multipleOf": x , "maximum": y , "minimum":- z</pre> <p>Dabei gilt Folgendes:</p> <p>x ist die kleinste verfügbare Einheit = $1 / 10^m$.</p> <p>y ist der maximale Wert, der durch die Kombination von n und m dargestellt werden kann.</p> <p>z ist der maximale Wert, der durch die Kombination von n und m dargestellt werden kann.</p>
FIXED DECIMAL(15) Unterstützt auf Zuordnungsebene 3.0 und höher, wenn DATETIME=PACKED15	<pre>"type":"string", "format":"date-time"</pre> <p>Das Format der Zeitmarke wird von RFC3339 definiert.</p>
BIT(n) Dabei ist n ein Vielfaches von 8. Andere Werte werden nicht unterstützt.	<pre>"type":"string" "maxLength": m</pre> <p>Dabei ist $m = n / 8$</p>
CHARACTER(n) VARYING und VARYINGZ werden auch auf Zuordnungsebene 1.2 und höher unterstützt. Restriction: VARYINGZ wird nur von Enterprise PL/I unterstützt.	<pre>"type":"string" "maxLength": n</pre>
GRAPHIC(n) VARYING und VARYINGZ werden auch auf Zuordnungsebene 1.2 und höher unterstützt. Restriction: VARYINGZ wird nur von Enterprise PL/I unterstützt.	<p>Auf Zuordnungsebene 1.0 und 1.1, wobei $m = 2 * n$:</p> <pre>"type":"string" "maxLength": m</pre> <p>Auf Zuordnungsebene 1.2 oder höher:</p> <pre>"type":"string" "maxLength": n</pre>

PL/I-Datenbeschreibung	JSON-Schemadefinition
<p>WIDECCHAR(<i>n</i>)</p> <p>Restriction: Nur Enterprise PL/I</p>	<p>Auf Zuordnungsebene 1.0 und 1.1, wobei $m = 2 * n$:</p> <pre>"type": "string" "maxLength": m</pre> <p>Auf Zuordnungsebene 1.2 oder höher:</p> <pre>"type": "string" "maxLength": n</pre> <p>Auf Zuordnungsebene 4.0 und höher füllt CICS das Anwendungsdatenstrukturfeld mit UTF-16-Daten.</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>Restriction: Nur Enterprise PL/I</p>	<pre>"type": "integer", "minimum": 0, "maximum": 255</pre>
<p>BINARY FLOAT(<i>n</i>)</p> <p>Dabei ist $n \leq 21$.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt BINARY FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre>"type": "number", "format": "float"</pre>

PL/I-Datenbeschreibung	JSON-Schemadefinition
<p>BINARY FLOAT(<i>n</i>)</p> <p>Dabei ist $21 < n \leq 53$.</p> <p>Werte größer als 53 werden nicht unterstützt.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt BINARY FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre>"type": "number", "format": "double"</pre>
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>Dabei ist $n \leq 6$.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre>"type": "number", "format": "float"</pre>

PL/I-Datenbeschreibung	JSON-Schemadefinition
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>Dabei ist $6 < n \leq 16$.</p> <p>Werte größer als 16 werden nicht unterstützt.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre>"type": "number", "format": "double"</pre>
<p><i>name</i> (<i>n</i>) <i>datenbeschreibung</i></p>	<p>Für Basiselemente:</p> <pre>"type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { "type": "object", "properties": { <i>name</i> : { <i>datenbeschreibung_JSON</i> } } } "required": [<i>name</i>]</pre> <p>Für Datendeklarationen:</p> <pre>"type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { <i>datenbeschreibung_JSON</i> }</pre> <p>Dabei ist <i>datenbeschreibung_JSON</i> die JSON-Schemadarstellung der PL/I-Datenbeschreibung.</p>

Zuordnung von JSON-Schema zu PL/I:

Das Dienstprogramm DFHJS2LS unterstützt Zuordnungen zwischen JSON-Schemas und PL/I-Datenstrukturen. Da sich der Enterprise PL/I-Compiler von älteren PL/I-Compilern unterscheidet, werden zwei Sprachoptionen unterstützt: PLI-ENTERPRISE und PLI-OTHER.

Regeln für die Zuordnung von Schemaelementnamen zu PL/I

Die CICS-Assistenten generieren eindeutige und gültige Namen für PL/I-Variablen aus den Schemaelementnamen mithilfe der folgenden Regeln:

1. Alle Zeichen außer A-Z, a-z, 0-9, @, #, _ oder \$ werden ersetzt durch X.

Beispiel: `monthly-total` wird zu `monthlyXtotal`.

Sie können den Parameter **MAPPING-OVERRIDES** verwenden, um anzupassen, wie andere Zeichen behandelt werden. Wenn Sie beispielsweise den Wert **HYPHENS-AS-UNDERSCORES** festlegen, werden alle Bindestriche im JSON-Schema in einen Unterstrich statt in ein X konvertiert. Beispiel: `monthly-total` wird zu `monthly_total`.

2. Doppelte Namen in demselben Bereich werden durch das Hinzufügen von ein oder zwei numerischen Ziffern zu der zweiten Instanz und zu allen nachfolgenden Instanzen des Namens eindeutig gemacht.

Beispiel: Drei Instanzen von `year` werden zu `year`, `year1` und `year2`.

Sollte das oben beschriebene Verhalten unerwünscht sein, kann der Benutzer **MAPPING-OVERRIDES=NO-ARRAY-NAME-INDEXING** als Eingabe für das Dienstprogramm angeben, die das Hinzufügen einer oder mehrerer Ziffern zur zweiten Instanz und allen nachfolgenden Instanzen des Namens inaktiviert.

3. Ein JSON-Schema gibt an, dass eine Variable eine variierende Kardinalität hat, wenn das Element "type" den Wert "array" hat und wenn die Schlüsselwörter "minItems" und "maxItems" nicht angegeben sind oder abweichende Werte aufweisen. Wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat, werden Feldnamen mit den Suffixes "_cont" und "_num" erstellt.

Weitere Informationen finden Sie unter „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.

4. Ein JSON-Schema gibt an, dass eine Variable optional ist, wenn sie nicht in dem "required"-Schlüsselwortarray enthalten ist, das dem einschließenden JSON-Schema vom Typ "object" zugeordnet ist. Bei optionalen Feldern wird ein zusätzliches Feld mit dem Suffix _num zu dem Elementnamen hinzugefügt. Zur Laufzeit ist dies gleich null, um anzugeben, dass der Wert in den JSON-Daten fehlt, und ungleich null, wenn der Wert in den JSON-Daten vorhanden war.
5. Feldnamen sind auf 31 Zeichen begrenzt. Wenn ein generierter Name, einschließlich aller Präfixe und Suffixe, diese Länge überschreitet, wird der Elementname abgeschnitten.

Der resultierende Name ist höchstens 31 Zeichen lang.

Regeln für die Zuordnung von Schematypen zu PL/I

DFHJS2LS ordnet Schematypwerte zu PL/I-Datentypen entsprechend der folgenden Tabelle zu. Beachten Sie außerdem die folgenden Punkte:

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL gesetzt ist, werden Zeichendaten mit variabler Länge auf null endenden Zeichenfolgen zugeordnet, und ein zusätzliches Zeichen wird dem Nullabschlusszeichen zugeordnet.

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher festgelegt und der Parameter **CHAR-VARYING** nicht angegeben ist, werden standardmäßig Zeichendaten mit variabler Länge zu einem VARYINGZ-Datentyp für Enterprise PL/I und einem VARYING-Datentyp für andere PL/I-Versionen zugeordnet.
- Binärdaten mit variabler Länge werden einem VARYING-Datentyp zugeordnet, wenn sie kleiner als 32.768s Byte sind, und einem Container, wenn sie größer als 32.768 Bytes sind.

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
Alle: "type": "null" "type": []	Nicht unterstützt
"\$schema": "http://json-schema.org/draft-04/schema#"	Dieses Schlüsselwort wird ignoriert, es wird jedoch davon ausgegangen, dass es mit der JSON-Schemaspezifikation 'draft-04' kompatibel ist.
"title": "same text" "description": "more text"	Diese Schlüsselwörter werden ignoriert.
"format": "<vordefinierte werte>"	Das Schlüsselwort "format" wird verwendet, um entweder die generierte Struktur oder den Laufzeitwert zu ändern. Lesen Sie die folgenden Informationen zur unterstützten Verwendung von "format".
<div> <div> </div> <div> "array", "items": {<JSON-unterschema>}, "additionalItems": false, "maxItems": m , "minItems": n </div> </div>	<p>Mehrdimensionale Arrays werden ab Zuordnungsebene 4.3 aufwärts unterstützt, und Arrays mit gemischtem Typ werden nicht unterstützt.</p> <p>"additionalItems" wird als falsch angenommen und es wird kein anderer Wert unterstützt.</p> <p>Wenn sowohl "minItems" als auch "maxItems" vorhanden und gleichwertig sind, wird das Array als feste Kardinalität behandelt. Ist dies nicht der Fall, wird es als variierende Kardinalität behandelt. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.</p>
"type": "array", "uniqueItems": true	"uniqueItems" wird mit JSON-Arrays nicht unterstützt. Das <JSON-unterschema> muss einen unterstützten Typ ("type") definieren, aber bei diesem Typ kann es sich nicht um ein Array ("array") handeln. Dies ist eine Einschränkung für die generierte Sprachstruktur.

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre> "type": "object", "additionalProperties": false, "properties": { ["<elementname>": {<JSON-unterschema> } [,]]* } "required": [["<elementname>" [,]]*] </pre>	<p>Das einzige JSON-Objekt, das derzeit unterstützt wird, ist eine feste Gruppe benannter Elemente.</p> <p>Dieses generiert eine Struktur (oder Unterstruktur), die diese Elementnamen verwendet.</p> <p>Für "additionalProperties" wird false angenommen, wenn kein Wert im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist.</p> <p>Alle Elemente im Objekt "properties" werden als "optional" angenommen, wenn sie sich nicht im Array "required" befinden oder wenn kein Array "required" vorhanden ist. Ein optionales Element ("optional") erhält eine variable Ordinalität von null bis X. Dabei ist X entweder 1 oder die maximale Anzahl von Elementen im Array, vorausgesetzt, das Element ist als Array definiert. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre>"type": "object", "additionalProperties": true</pre>	<p>Das Objekt wird wie im vorherigen Beispiel zugeordnet, mit zusätzlichen Feldern, die zusätzliche Eigenschaften unterstützen. Für die Eigenschaft <code>additionalProperties</code> wird 'false' angenommen, wenn sie nicht im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist.</p> <p>Ist diese Option aktiviert, werden bis zu dem im Parameter ADDITIONAL-PROPERTIES-MAX angegebenen Wert Leerzeichen zugeordnet. Die Anzahl der Zeichen in jedem Leerzeichen wird durch den Parameter ADDITIONAL-PROPERTIES-SIZE festgelegt. Jede Eigenschaft wird einem Feld PIC X(z) zugeordnet, wobei z durch den Parameter ADDITIONAL-PROPERTIES-SIZE definiert wird. Ist der Wert von ADDITIONAL-PROPERTIES-MAX größer als 0, werden die Eigenschaften in ähnlicher Weise zugeordnet wie in einem Array, in dem maxItems festgelegt ist.</p> <p>Anmerkung: Es gibt mehrere Möglichkeiten, JSON-Unterstützung in CICS zu konfigurieren, einschließlich der Verwendung von z/OS Connect for CICS. Wenn Sie die ältere CICS-Java-Pipeline-Technologie nutzen, werden <code>AdditionalProperties</code> nur unterstützt, falls die <code>com.ibm.cics.json.enableAxis2HandlersJVM-Systemeigenschaft</code> nicht auf 'true' festgelegt ist.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>Keines dieser Schlüsselwörter wird mit JSON-Objekten unterstützt.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression>", "minLength": l</pre>	<p><code>char[z]</code></p> <p>Dabei basiert der Wert von z auf m , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p>m basiert auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>"pattern"- und "minLength"-Einschränkungen werden über die Sprachstruktur nur in Form eines Kommentars übergeben.</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre>"type":"string" "maxLength": <i>m</i></pre>	<p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>WIDECHAR(<i>z</i>)</p> <p>Dabei basiert der Wert von <i>z</i> auf <i>m</i> , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p><i>m</i> basiert auf dem Schlüsselwort "maxLength " und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*":{ "type":"string", "format":"date-time" }</pre>	<p>FIXED DECIMAL (15,0)</p> <p>Alle unterstützt, wenn DATETIME=PACKED15</p>
<pre>"*name*":{ "type":"string", "format":"uri" }</pre>	<p>CHAR(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>WIDECHAR(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*":{ "type":"string", "format":"base64Binary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"hexBinary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength".</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre>"*name*":{ "type":"string", "format":"<vordefiniert>" }</pre>	<p>CHAR(<i>m</i>) Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist entweder <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i>. Ein relevantes Muster ("pattern") wird an den Kommentar übergeben.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>WIDECHAR(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i> . Ein relevantes Muster ("pattern") wird verwendet und an den Kommentar übergeben.</p>
"type":"boolean"	<p>Zuordnungsebene 1.1 und niedriger:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Andere PL/I-Versionen FIXED BINARY (7)</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I BIT(7) BIT(1)</p> <p>Andere PL/I-Versionen BIT(7) BIT(1)</p> <p>Dabei wird BIT(7) für die Ausrichtung angegeben und BIT(1) enthält den zugeordneten booleschen Wert.</p>
<pre>"type": "integer", "exclusiveMaximum": true, "exclusiveMinimum": true, "multipleOf": n</pre>	<p>"exclusiveMaximum"- und "exclusiveMinimum"-Einschränkungen werden nur in Form eines Kommentars an die Sprachstruktur übergeben.</p> <p>"multipleOf" wird ignoriert.</p>
<pre>"type":"integer", minimum:-128, maximum:127</pre>	<p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Andere PL/I-Versionen FIXED BINARY (7)</p>
<pre>"type":"integer", minimum:0, maximum:255</pre>	<p>Enterprise PL/I UNSIGNED FIXED BINARY (8)</p> <p>Andere PL/I-Versionen FIXED BINARY (8)</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
"type": "integer", minimum: -32768, maximum: 32767	Enterprise PL/I SIGNED FIXED BINARY (15) Andere PL/I-Versionen FIXED BINARY (15)
"type": "integer", minimum: 0, maximum: 65535	Enterprise PL/I UNSIGNED FIXED BINARY (16) Andere PL/I-Versionen FIXED BINARY (16)
"type": "integer", minimum: -2147483648, maximum: 2147483647	Enterprise PL/I SIGNED FIXED BINARY (31) Andere PL/I-Versionen FIXED BINARY (31)
"type": "integer", minimum: 0, maximum: 4294967295	Zuordnungsebene 1.1 und niedriger: Enterprise PL/I UNSIGNED FIXED BINARY (32) Zuordnungsebene 1.2 und höher: Enterprise PL/I CHAR(<i>y</i>) Dabei ist <i>y</i> eine feste Länge kleiner 16 MB. Alle Zuordnungsebenen: Andere PL/I-Versionen BIT (64)

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre>"type": "integer", minimum: -9223372036854775808, maximum: 9223372036854775807</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p>Enterprise PL/I SIGNED FIXED BINARY(63) Anmerkung: Die LIMITS-Compilersteueranweisung kann sich auf die Interpretation dieses Felds durch den PL/I-Compiler auswirken. CICS erwartet eine deklarierte Größe dieses Felds, aber der Compiler optimiert das Feld unter Umständen, indem er es verkleinert, was zu einer Abweichung führt. Vermeiden Sie solche Probleme, indem Sie die Kompilierzeitoption LIMITS(FIXEDBIN(63)) verwenden.</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I CHAR(y)</p> <p>Dabei ist <i>y</i> eine feste Länge kleiner 16 MB.</p> <p>Alle Zuordnungsebenen:</p> <p>Andere PL/I-Versionen BIT(64)</p>
<pre>"type": "integer", minimum: 0, maximum: 18446744073709551615</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(64) Anmerkung: Die LIMITS-Compilersteueranweisung kann sich auf die Interpretation dieses Felds durch den PL/I-Compiler auswirken. CICS erwartet eine deklarierte Größe dieses Felds, aber der Compiler optimiert das Feld unter Umständen, indem er es verkleinert, was zu einer Abweichung führt. Vermeiden Sie solche Probleme, indem Sie die Kompilierzeitoption LIMITS(FIXEDBIN(63)) verwenden.</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I CHAR(y)</p> <p>Dabei ist <i>y</i> eine feste Länge kleiner 16 MB.</p> <p>Andere PL/I-Versionen BIT(64)</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
"type": "number" "description": "decimal"	FIXED DECIMAL(<i>n</i> , <i>m</i>)
"type": "number" "description": "float"	<p>Zuordnungsebenen 1.0 und 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I DECIMAL FLOAT(6) HEXADEC</p> <p>Andere PL/I-Versionen DECIMAL FLOAT(6)</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>
"type": "number" "description": "double"	<p>Zuordnungsebenen 1.0 und 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I DECIMAL FLOAT(16) HEXADEC</p> <p>Andere PL/I-Versionen DECIMAL FLOAT(16)</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre> oneOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] anyOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] allOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] </pre>	<p>Die logischen Pfade durch das Array von Optionen werden so zusammengeführt, als ob ein einzelnes Objekt vom Typ { "properties": { "A":..., "B":... } } definiert wäre. Jede dieser JSON-Eigenschaften wird als optional behandelt. Wenn die Unteroptionen widersprüchliche Definitionen für denselben Eigenschaftsnamen enthalten, wird eine Fehlermeldung ausgegeben. Wenn eine einzelne Eigenschaft beispielsweise sowohl als Zeichenfolge (in Pfad A) als auch als Objekt (in Pfad B) definiert ist, wird diese Definition nicht unterstützt und führt zu einer Fehlermeldung.</p> <p>Es ist möglich, dass ein JSON-Schema komplexe logische Strukturen definiert. Die in den komplexen logischen Strukturen implizierten Feinheiten können jedoch bei der Zuordnung zu einer Sprachstruktur verloren gehen. Der Umsetzungsprozess versucht nicht, die kombinatorischen Regeln aus dem Schema umzusetzen. Es interagiert nur mit den Sprachstrukturfeldern, die angeben, ob eine bestimmte JSON-Eigenschaft vorhanden oder nicht vorhanden ist.</p> <p>Wenn die Unteroptionen kompatible Definitionen für denselben Eigenschaftsnamen enthalten, versucht DFHJS2LS, das zugeordnete Muster von Einschränkungen zusammenzuführen, wobei allerdings einige Feinheiten verloren gehen können. Nehmen wir beispielsweise die folgende Definition an:</p> <pre> "A": { "oneOf": [{ "type": "string", "maxLength": 5 }, { "type": "string", "minLength": 7, "maxLength": 8 }] } </pre> <p>„A“ ist entweder als eine Zeichenfolge mit einer Länge von bis zu 5 Zeichen oder als eine Zeichenfolge mit einer Länge zwischen 7 und 8 Zeichen definiert. Die Zusammenführung kann zu Zuordnungen zu einer Zeichenfolge mit einer Länge von 0 bis 8 Zeichen führen, ohne dass erkannt wird, dass die Länge von 6 Zeichen ungültig ist.</p> <p>Die meisten Szenarios mit logischer Zusammensetzung werden einfachen Sprachstrukturen zugeordnet, aber bei einer komplizierten logischen Zusammensetzung werden während des Zuordnungsprozesses möglicherweise Kompromisse geschlossen. Die besten Ergebnisse erhalten Sie, wenn Sie die logische Zusammensetzung in einem JSON-Schema zum Definieren alternativer Deklarationen für dieselbe JSON-Eigenschaft vermeiden.</p>

Anmerkung: CICS kann keine ganzzahligen Werte größer als den maximalen Wert für 'signed long' ($2^{63} - 1$) umsetzen, es sei denn, sie sind in Anführungszeichen eingeschlossen.

Anmerkung: Minimale und maximale Werte, die in dem Schema für numerische Typen angegeben sind, werden nur zum Zuordnen zu einem PL/I-Datentyp verwendet. Die Daten werden während der Laufzeit nicht gegen diese Werte geprüft.

Variable Arrays von Elementen in DFHJS2LS

JSON kann Gruppen mit einer unterschiedlichen Anzahl an Elementen enthalten. Im Allgemeinen werden JSON-Schemas, die eine unterschiedliche Anzahl an Elementen enthalten, einer einzelnen Datenstruktur einer höheren Programmiersprache nicht effizient zugeordnet. CICS verarbeitet eine unterschiedliche Anzahl an Elementen in JSON-Daten mithilfe von containerbasierten Zuordnungen oder Inline-Zuordnungen.

Ein Array mit einer variierenden Anzahl von Elementen wird in dem JSON-Schema mithilfe der Schlüsselwörter `minItems` und `maxItems` in dem Schema mit einem "type"-Wert von "array" dargestellt:

- Das Schlüsselwort `minItems` gibt die Mindestanzahl von Vorkommen des Elements an. Es hat den Wert 0 oder eine beliebige positive ganze Zahl. Der Standardwert ist 0.
- Das Schlüsselwort `maxItems` gibt die Höchstanzahl von Vorkommen des Elements an. Es kann als Wert jede positive ganze Zahl größer-gleich dem Wert des Schlüsselworts `minItems` haben.
- Wenn das Schlüsselwort `maxItems` nicht angegeben ist, bedeutet das, dass das Array unbegrenzt ist.

Ein optionales Feld kann durch ein variables Array von `"maxItems":1` bezeichnet werden. Beispiel: Eine optionale 8-Byte-Zeichenfolge namens "component" :

```
"properties":{
  "component": {
    "type":"array",
    "maxItems":1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

Derselbe Effekt kann erzeugt werden, indem der Feldname nicht in den Schlüsselwortwert "required" eingeschlossen wird:

```
"properties":{
  "component": {
    "type": "string",
    "maxLength": 8
  }
}
```

Im Allgemeinen werden JSON-Schemas, die eine unterschiedliche Anzahl an Elementen enthalten, einer einzelnen Datenstruktur einer höheren Programmiersprache nicht effizient zugeordnet. Für diese Fälle verwendet CICS eine Reihe von verbundenen Datenstrukturen, die in einer Reihe von Containern an das

Anwendungsprogramm übergeben werden. Diese Strukturen werden als Eingabe und Ausgabe der Anwendung verwendet:

- Wenn CICS JSON-Daten in Anwendungsdaten umsetzt, füllt es diese Strukturen mit den Anwendungsdaten, und die Anwendung liest sie.
- Wenn CICS die Anwendungsdaten in JSON-Daten umsetzt, liest es die Anwendungsdaten in den Strukturen, die von der Anwendung gefüllt wurden.

Das folgende Beispiel veranschaulicht das Format dieser Datenstrukturen. Diese Beispiele verwenden ein Array von einfachen 8-Byte-Feldern. Das Modell unterstützt jedoch Arrays komplexer Datentypen ebenso wie Arrays von Datentypen, die andere Arrays enthalten.

Beispiel 1. Feste Anzahl von Elementen

Dieses Beispiel stellt ein Element dar, das genau dreimal vorkommt.

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 3,
    "minItems": 3,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

In diesem Beispiel ist die Anzahl der Vorkommen des Elements vorab bekannt, deshalb kann es als Array fester Länge in einer einfachen COBOL-Deklaration oder als das Äquivalent in anderen Sprachen dargestellt werden:

```
05 component PIC X(8) OCCURS 3 TIMES
```

Beispiel 2. Variierende Anzahl von Elementen auf Zuordnungsebene 2 und niedriger

Dieses Beispiel stellt ein obligatorisches Element dar, das ein- bis fünfmal vorkommen kann:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

Die Hauptdatenstruktur enthält eine Deklaration mit zwei Feldern. Wenn CICS die JSON-Daten in binäre Daten umsetzt, enthält das erste Feld, component-num, die Anzahl von Vorkommen des Elements in den JSON-Daten und das zweite Feld, component-cont, den Namen eines Containers:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Eine zweite Datenstruktur enthält die Deklaration des Elements selbst:

```
01 DFHJS-component
02 component PIC X(8)
```

Sie müssen den Wert von `component-num` untersuchen, wobei es sich um einen Wert im Bereich zwischen 1 und 5 handelt, um herauszufinden, wie oft das Element vorkommt. Der Elementinhalt befindet sich im Container namens `component-cont`. Der Container enthält ein Array von Elementen, wobei jedes Element von der Datenstruktur `DFHJS-component` zugeordnet wird.

Wenn `minItems="0"` oder nicht angegeben ist und wenn `maxItems="1"`, ist das Element optional. Um die Datenstruktur in Ihrem Anwendungsprogramm zu verarbeiten, müssen Sie den Wert von `component-num` untersuchen:

- Ist er 0, hat die Nachricht kein Komponentenelement und der Inhalt von `component-cont` ist nicht definiert.
- Ist er 1, befindet sich das Komponentenelement im Container namens `component-cont`.

Der Inhalt des Containers wird von der Datenstruktur `DFHJS-component` zugeordnet.

Anmerkung: Wenn die JSON-Daten aus einem einzelnen wiederkehrenden Element bestehen, generiert DFHJS2LS zwei Sprachstrukturen. Die Hauptsprachstruktur enthält die Anzahl von Elementen in dem Array und den Namen eines Containers, der das Array von Elementen enthält. Die zweite Sprachstruktur ordnet eine einzelne Instanz des wiederkehrenden Elements zu.

Beispiel 3. Variierende Anzahl von Elementen auf Zuordnungsebene 2.1 und höher

Auf Zuordnungsebene 2.1 und höher können Sie den Parameter **INLINE-MAXOCCURS-LIMIT** in den CICS-Assistenten verwenden. Der Parameter **INLINE-MAXOCCURS-LIMIT** gibt an, wie die variierende Anzahl von Elementen verarbeitet wird. Bei den Zuordnungsoptionen für eine variierende Anzahl von Elementen handelt es sich containerbasierte Zuordnung, wie unter „Beispiel 2. Variierende Anzahl von Elementen auf Zuordnungsebene 2 und niedriger“ auf Seite 305 beschrieben, oder um eine Inline-Zuordnung. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein:

- Der Standardwert von **INLINE-MAXOCCURS-LIMIT** ist 1, was sicherstellt, dass optionale Elemente inline zugeordnet werden.
- Der Wert 0 für den Parameter **INLINE-MAXOCCURS-LIMIT** verhindert eine Inline-Zuordnung.
- Wenn `maxItems` kleiner-gleich dem Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die Inline-Zuordnung verwendet.
- Wenn `maxItems` größer als der Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die containerbasierte Zuordnung verwendet.

Die Inline-Zuordnung einer variierenden Anzahl von Elementen führt dazu, dass sowohl ein Array (wie in dem obigen Beispiel für die feste Anzahl von Vorkommen) als auch ein Zähler generiert wird. Das Feld `component-num` gibt an, wie viele Instanzen des Elements vorhanden sind, und auf diese wird vom Array verwiesen. Für das Beispiel in „Beispiel 2. Variierende Anzahl von Elementen auf Zuordnungsebene 2 und niedriger“ auf Seite 305, bei dem **INLINE-MAXOCCURS-LIMIT** kleiner-gleich 5 ist, sieht die generierte Datenstruktur wie folgt aus:

```
05 component-num PIC S9(9) COMP-5 SYNC.
05 component OCCURS 5 PIC X(8).
```

Das erste Feld, component-num , ist identisch mit der Ausgabe für das Beispiel für die containerbasierte Zuordnung im vorherigen Abschnitt. Das zweite Feld enthält ein Array mit der Länge 5, das groß genug ist, um die maximale Anzahl von Elementen, die generiert werden können, zu enthalten.

Die Inline-Zuordnung unterscheidet sich von der containerbasierten Zuordnung, die die Anzahl von Vorkommen des Elements und den Namen des Containers, in dem die Daten enthalten sind, speichert, insofern als dass sie alle Daten in dem aktuellen Container speichert. Durch das Speichern der Daten im aktuellen Container wird die Leistung im Allgemeinen verbessert, was die Inline-Zuordnung zur präferierten Lösung macht.

Beispiel 4. Verschachtelte variierende Arrays

Komplexe JSON-Schemas können variierende wiederkehrende Elemente enthalten, die ihrerseits wiederum variierende wiederkehrende Elemente enthalten können. In diesem Fall erstreckt sich die beschriebene Struktur über die beiden in den Beispielen beschriebenen Ebenen hinaus.

Dieses Beispiel stellt ein optionales Element namens "component2" dar, das in einem obligatorischen Element namens "component1" verschachtelt ist, wobei das obligatorische Element ein- bis fünfmal vorkommen kann:

```
"properties":{
  "component1": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "object",
      "properties":{
        "component2":{
          "type": "string",
          "maxLength": 8
        }
      },
      "required": ["component2"]
    },
    "required": ["component1"]
  }
}
```

Die übergeordnete Datenstruktur ist identisch mit früheren Beispielen:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

Die zweite Datenstruktur enthält jedoch die folgenden Elemente:

```
01 DFHJS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Eine Struktur auf dritter Ebene enthält die folgenden Elemente:

```
01 DFHJS-component2
02 component2 PIC X(8)
```

Die Anzahl von Vorkommen des äußersten Elements "component1" ist in component1-num angegeben.

Der in component1-cont benannte Container enthält ein Array mit dieser Anzahl von Instanzen der zweiten Datenstruktur DFHJS-component1.

Jede Instanz von component2-cont benennt einen anderen Container, der jeweils die Datenstruktur enthält, die von der Struktur auf dritter Ebene, DFHJS-component2, zugeordnet wird.

Um diese Struktur zu veranschaulichen, ziehen Sie das Fragment von JSON-Daten in Betracht, das dem folgenden Beispiel entspricht:

```
{ "component1":  
  [  
    {  
      "component2": "string1"  
    },  
    {  
      "component2": "string2"  
    },  
  ]  
}
```

"component1" kommt dreimal vor. Die ersten beiden Vorkommen enthalten eine Instanz von "component2" , das dritte Vorkommen nicht.

In der übergeordneten Datenstruktur enthält component-num den Wert 3. Der in component1-cont benannte Container enthält drei Instanzen von DFHJS-component1 :

1. In der ersten Instanz hat component2-num den Wert 1 und der in component2-cont benannte Container enthält *string1*.
2. In der zweiten Instanz hat component2-num den Wert 1 und der in component2-cont benannte Container enthält *string2*.
3. In der dritten Instanz hat component2-num den Wert 0 und der Inhalt von component2-cont ist nicht definiert.

In dieser Instanz wird die gesamte Datenstruktur von vier Containern dargestellt:

- Stammdatenstruktur in Container DFHJS-DATA
- In component1-cont benannter Container
- In den ersten beiden Instanzen von component2-cont benannte zwei Container

Optionale Strukturen und das Schlüsselwort **required**

Datenstrukturen werden durch das JSON-Schema "type" von "object" definiert. Die Schemas setzen Feldnamen zu einzelnen Typen in Bezug mithilfe des durch das Schlüsselwort "properties" bereitgestellten Objekts. Die Anforderung, dass diese Felder Bestandteil der JSON-Daten sein müssen, die durch das JSON-Schema beschrieben werden, wird von dem im Schlüsselwort "required" angegebenen Array gesteuert. In diesem Array werden alle Feldnamen aufgelistet, die in den JSON-Daten vorhanden sein müssen. Optionale Felder werden daher entweder durch ihr Fehlen in diesem Array dargestellt oder durch das Fehlen des Schlüsselworts "required", da das Array nicht leer sein darf. In diesem Fall sind alle Felder optional.

Optionale Felder werden als variierendes Array von null oder einem Element behandelt. Dadurch wird ein zusätzliches Feld mit dem Suffix "-num" an den Elementnamen angehängt. Wenn die Gesamtlänge mehr als 28 Zeichen beträgt, wird der Elementname abgeschnitten. Zur Laufzeit wird dies ungleich null sein, um anzugeben, dass der Wert in den JSON-Daten vorhanden war, und gleich null, wenn er nicht vorhanden war.

In diesem Beispiel werden zwei Felder dargestellt, ein obligatorisches namens "required-structure" und ein optionales namens "optional-structure" :

```

{
  "type": "object",
  "properties": {
    "required-structure": {
      "type": "string",
      "maxLength": 8
    },
    "optional-structure": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": [
    "required-structure"
  ]
}

```

Die generierte COBOL-Struktur zeigt beide Felder an, aber dem zweiten wird "optional-structure-num" vorangestellt. Dies ist eine ganzzahlige Angabe der Elemente, wobei 0 angibt, dass keine Elemente vorhanden sind, und 1 angibt, dass eines vorhanden ist. Der Wert wird festgelegt, um anzugeben, ob "optional-structure" gültige Daten enthält oder nicht.

```

03 OutputData.
06 required-structure PIC X(8).
06 optional-structure-num PIC S9(9) COMP-5 SYNC.
06 optional-structure PIC X(8).

```

Unterstützung für UTF-16 in Anwendungsdaten

CICS-Web-Services unterstützen die Konvertierung von Anwendungsdaten, die in UTF-16 codiert sind, in XML oder JSON und umgekehrt. Verwenden Sie UTF-16, wenn Sie Daten in mehreren Sprachen speichern und verarbeiten müssen.

SOAP- und JSON-WebServices von CICS unterstützen die Konvertierung von Anwendungsdaten, die in UTF-16 codiert sind, in XML oder JSON und umgekehrt. Unicode ist ein Schema für Codeumsetzung mit variabler Breite, das es Systemen ermöglicht, Daten effizient zu verarbeiten.

UTF-16 ist eine Unicode-Codierung mit variabler Breite, bei der jedes Zeichen durch zwei oder vier Bytes dargestellt wird. CICS-Web-Services unterstützen CC-SID 1200 für Anwendungsdaten im Format UTF-16 BE (Big Endian) mit IBM Private Use Area. Dieses Verhalten ist konsistent mit der UTF-16-Unterstützung in allen unterstützten Sprachen.

UTF-16 wird auf Zuordnungsebene 4.0 und höher unterstützt. Sie können mithilfe von Zuordnungseinstellungen in den Assistenten anpassen, wie Anwendungsdaten konvertiert werden. Weitere Informationen zu XML-Zuordnungsebenen finden Sie unter Mapping levels for the CICS assistants. Weitere Informationen zu JSON-Zuordnungsebenen finden Sie unter Mapping levels for the CICS JSON assistants.

Anmerkung: UTF-16 erfordert mehr Verarbeitungszeit und ist weniger speichereffizient als EBCDIC-Codierungen. Hinzu kommt, dass gemischte Codierungstypen zu zusätzlicher Laufzeitverarbeitung führen.

UTF-16 aus einem XML- oder JSON-Schema Sprachstrukturen zuordnen

Die Unterstützung für UTF-16 hängt davon ab, wie Sie den Web-Service erstellen. Die Zuordnung von XML- oder JSON-Schemas zu Sprachstrukturen, auch bekannt als Top-down-Zuordnung, weist die folgenden Merkmale auf. Wenn UTF-16 akti-

viert ist, werden alle Textfelder UTF-16-Feldern zugeordnet, während numerische Anzeigedatentypen in COBOL als EBCDIC zugeordnet werden. Für die Verwendung von UTF-16 setzen Sie den CCSID-Parameter von DFHJS2LS, DFHSC2LS oder DFHWS2LS auf 1200.

Beispiel: Wenn das folgende XML-Schemafragment in der WSDL vorhanden ist:

```
<xsd:element name="myString" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="20"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Dann generiert der DFHWS2LS-Assistent das folgende Feld in einer COBOL-Sprachstruktur:

```
myString PIC N(
20
) USAGE NATIONAL
```

Mit dem Parameter CHAR-MULTIPLIER der Web-Service-Assistenten kann die Länge eines Felds angegeben werden, das die Assistenten generieren.

CHAR-MULTIPLIER

Wenn Sie UTF-16 verwenden, sind die einzig gültigen Werte für den Parameter **CHAR-MULTIPLIER** 2 oder 4 , wobei 2 der Standardwert ist.

CHAR-MULTIPLIER = 2 , wobei das Schema die Zeichenfolge `maxlength x` beschreibt, generiert `PIC N(x)` . Das Festlegen von **CHAR-MULTIPLIER** = 2 schließt nicht die Verwendung von Ersatzzeichenpaaren in einer UTF-16-Zeichenfolge aus, wirkt sich aber auf die Anzahl von Zeichen aus, die in das Feld passen.

CHAR-MULTIPLIER = 4 generiert `PIC N(2x)` . Wenn **CHAR-MULTIPLIER** = 4 , wird der Wert zur Laufzeit aufgefüllt, wenn die Zeichenfolge Zeichen umfasst, die in einer einzelnen Codierungseinheit ausgedrückt werden können.

UTF-16 aus Sprachstrukturen XML- oder JSON-Schemas zuordnen

Die Zuordnung aus einer Sprachstruktur zu einem XML- oder JSON-Schema, auch als Bottom-up-Zuordnung bezeichnet, wird auf andere Weise verwaltet als die Top-down-Zuordnung. Wenn eine UTF-16-Zeichenfolge in der Sprachstruktur deklariert wird, werden die Daten von CICS als UTF-16-codiert interpretiert, andernfalls wird angenommen, dass die Daten EBCDIC-codiert sind. Der CCSID-Parameter für DFHLS2JS, DFHLS2SC oder DFHLS2WS gibt die Codierung eines beliebigen EBCDIC-Texts innerhalb der Anwendungsdaten an. Er darf nicht UTF-16 angeben.

Die Datentypen, die als UTF-16-Zeichen interpretiert werden, sind folgende: `PIC N(n)` in COBOL, `WIDECHAR(n)` in PL/I und `char16_t[n]` in C und C++.

Der Parameter CHAR-USAGE des Web-Service-Assistenten kann verwendet werden, um Datentypen anzugeben.

CHAR-USAGE

In COBOL kann der nationale Datentyp, `PIC N`, für UTF-16- oder DBCS-Daten verwendet werden. Diese Einstellung wird durch die Compileroption `NSYMBOL` gesteuert. Sie müssen den Parameter **CHAR-USAGE** im Assistenten

auf denselben Wert setzen wie die Compileroption NSYMBOL, um sicherzustellen, dass die Daten richtig verarbeitet werden. Dieser Wert wird in der Regel auf CHAR-USAGE=NATIONAL gesetzt, wenn Sie UTF-16 verwenden.

Wenn Sie nationale Datentypen, die UTF-16- und DBCS-Daten in demselben Copybook enthalten, mischen möchten, können Sie die Qualifikationsmerkmale USAGE NATIONAL oder USAGE DISPLAY-1 in einzelnen Feldern verwenden.

Anmerkung: DFHLS2WS, DFHLS2SC und DFHLS2JS bieten keine Unterstützung für die COBOL-Klausel GROUP USAGE NATIONAL.

Zuordnungen aus einer Sprachstruktur generieren

Zum Erstellen von JavaScript Object Notation (JSON) aus Anwendungsdaten oder von Anwendungsdaten aus JSON erstellen Sie die Zuordnungen, die beschreiben, wie CICS die Daten und JSON zur Laufzeit umsetzen soll. Sie können mit einem Anwendungsdatensatz anfangen, beispielsweise mit einem Kommunikationsbereich, einer VSAM-Datei, einer Warteschlange für temporären Speicher oder einem IBM Db2-Datensatz.

Before you begin

Bevor Sie die Zuordnungen erstellen, müssen Sie sicherstellen, dass diese Bedingungen erfüllt sind:

- Sie müssen über eine Sprachstruktur verfügen, die den Anwendungsdatensatz in einer partitionierten Datei beschreibt. Die Sprachstruktur kann in allen höheren Programmiersprachen geschrieben werden, die vom CICS JSON-Assistenten unterstützt werden: COBOL, PL/I, C und C++.
- Sie müssen die Benutzer-ID konfigurieren, unter der DFHLS2JS ausgeführt wird, um z/OS UNIX zu verwenden.
- Die Benutzer-ID muss über Leseberechtigung verfügen, um auf die Sprachstruktur zugreifen zu können, und über Schreibberechtigung, um die Ausgabe in die entsprechenden Verzeichnisse unter z/OS UNIX zu stellen.
- Sie müssen der Benutzer-ID ausreichend Speicherplatz zuordnen, damit Java ausgeführt werden kann. Sie können alle unterstützten Versionen von Java verwenden.

About this task

Verwenden Sie den CICS JSON-Assistenten, um die Datenzuordnungen für den Anwendungsdatensatz zu erstellen. Der CICS JSON-Assistent erstellt ein CICS-Bundle und gibt Fehlermeldungen zu allen nicht unterstützten Elementen aus, die er in Ihrer Sprachstruktur erkennt. Die Referenzinformationen für den CICS JSON-Assistenten listen die Einschränkungen auf, die für die einzelnen höheren Programmiersprachen gelten.

Procedure

Führen Sie den DFHLS2JS-Stapeljob aus. DFHLS2JS hat optionale Parameter, die Sie auswählen, um Ihre Anforderungen zu erfüllen, z. B. die Auswahl einer bestimmten Codepage. Verwenden Sie mindestens die folgenden Parameter:

- Geben Sie die höhere Programmiersprache Ihrer Sprachstruktur im Parameter **LANG** an.

- Geben Sie den Namen und die Position einer Bundleressource im Parameter **BUNDLE** an.
- Geben Sie die Zuordnungsebene im Parameter **MAPPING-LEVEL** an. Obwohl Sie auf jeder Zuordnungsebene auf die erweiterten Zuordnungsoptionen zugreifen können, verwenden Sie die aktuelle Zuordnungsebene.
- Geben Sie die Position und die Codepage der Sprachstrukturen an, die den Anwendungsdatensatz in den Parametern **PDSMEM** und **PDSCP** beschreiben.
- Geben Sie den Namen und die Position der JSON-Schemadatei (.json) im Parameter **JSON-SCHEMA** an. Wenn die Datei nicht vorhanden ist, erstellt DFHLS2JS das JSON-Schema, aber nicht die Verzeichnisstruktur.
- Geben Sie den Namen an, der für die JSONTRANSFRM-Bundleressource verwendet wird. Dieser Name wird von Anwendungen zum Identifizieren der JSON-Zuordnungen verwendet.

Der Stapeljob erstellt eine Bundleverzeichnisstruktur unter z/OS UNIX. Das Bundleverzeichnis hat ein Unterverzeichnis META-INF, das das Bundlemanifest enthält. Der Stapeljob erstellt außerdem ein JSON-Schema und eine JSON-Bindung in dem Bundle, wobei er die Dateinamen verwendet, die Sie für die Parameter **JSONTRANSFRM** und **JSON-SCHEMA** angegeben haben.

Installieren Sie die BUNDLE-Ressource, die diese JSON-Bindung angibt. Die JSONTRANSFRM-Bundleressource erstellt dynamisch eine JSONTRANSFRM-Ressource, die die Position des JSON-Schemas und der Bindungsdatei definiert. Sie können die Betriebsansichten des CICS Explorer-Bundles und der Bundleteile verwenden, um den Status von installierten BUNDLE-Ressourcen zu überprüfen.

Results

Ein CICS-Bundle, das eine JSON-Umsetzung enthält, wird generiert.

Das folgende Beispiel zeigt DFHLS2JS mit dem Mindestsatz von angegebenen Parametern an.

```
//LS2JS JOB 'abrechnungsdaten',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
/*
```

What to do next

Schreiben Sie ein Anwendungsprogramm, um die Anwendungsdaten in JSON umzusetzen und umgekehrt. Sie können dieselben Zuordnungen für beide Umsetzungen verwenden.

Zuordnungen aus einem JSON-Schema generieren

Zum Erstellen von Anwendungsdaten aus JavaScript Object Notation (JSON) oder von JSON aus Anwendungsdaten erstellen Sie die Zuordnungen, die beschreiben, wie CICS die Daten und JSON zur Laufzeit umsetzen soll. Sie können mit einem JSON-Schema starten. Nach dem Generieren der Sprachstruktur und der Zuord-

nung können Sie eine CICS-Anwendung mithilfe der Sprachstruktur entwickeln und JSON in Anwendungsdaten umsetzen und umgekehrt.

Before you begin

Bevor Sie die Zuordnungen erstellen, müssen Sie sicherstellen, dass diese Vorbedingungen erfüllt sind:

- Sie müssen über das JSON-Schema verfügen, das einen JSON-Datensatz beschreibt.
- Sie müssen die Benutzer-ID konfigurieren, unter der DFHJS2LS ausgeführt wird, um z/OS UNIX zu verwenden.
- Die Benutzer-ID muss über Leseberechtigung verfügen, um auf das JSON-Schema zugreifen zu können, und über Schreibberechtigung, um die Ausgabe in die entsprechenden Verzeichnisse unter z/OS UNIX zu stellen.
- Sie müssen der Benutzer-ID ausreichend Speicherplatz zuordnen, damit Java ausgeführt werden kann. Sie können alle unterstützten Versionen von Java verwenden.

About this task

Verwenden Sie den CICS JSON-Assistenten, um die Datenzuordnungen für den Anwendungsdatensatz zu erstellen. Der CICS JSON-Assistent erstellt ein CICS-Bundle und gibt Fehlermeldungen zu allen nicht unterstützten Elementen aus, die er in Ihrer Sprachstruktur erkennt. Die Referenzinformationen für den CICS JSON-Assistenten listen die Einschränkungen auf, die für die einzelnen höheren Programmiersprachen gelten. Weitere Informationen finden Sie unter „DFHJS2LS: Konvertierung eines JSON-Schemas in eine höhere Programmiersprache für eine verknüpfbare Schnittstelle“ auf Seite 444.

Procedure

Führen Sie den DFHJS2LS-Stapeljob aus. DFHJS2LS hat optionale Parameter, die Sie auswählen, um Ihre Anforderungen zu erfüllen, z. B. die Auswahl einer bestimmten Codepage. Verwenden Sie mindestens die folgenden Parameter:

- Geben Sie den Namen und die Position einer Bundleressource im Parameter **BUNDLE** an.
- Geben Sie den Namen und die Position der JSON-Schemadatei im Parameter **JSON-SCHEMA** an.
- Geben Sie die Zuordnungsebene im Parameter **MAPPING-LEVEL** an. Obwohl Sie auf jeder Zuordnungsebene auf die erweiterten Zuordnungsoptionen zugreifen können, verwenden Sie die aktuelle Zuordnungsebene.
- Geben Sie die höhere Programmiersprache für die generierte Sprachstruktur im Parameter **LANG** an.
- Geben Sie die Position und die Codepage der Sprachstrukturen an, die den Anwendungsdatensatz in den Parametern **PDSMEM** und **PDSCP** beschreiben. DFHJS2LS erstellt die Sprachstruktur, aber nicht die Verzeichnisstruktur.
- Geben Sie den Namen an, der für die JSONTRANSFRM-Bundleressource in CICS verwendet wird. Dieser Name wird von Anwendungen zum Identifizieren der JSON-Zuordnungen verwendet.

Der Stapeljob erstellt eine Bundleverzeichnisstruktur unter z/OS UNIX. Das Bundleverzeichnis hat ein Unterverzeichnis META-INF, das das Bundlemanifest enthält. Der Stapeljob erstellt außerdem eine JSON-Bindung und kopiert das JSON-Schema in das Bundleverzeichnis, wobei er die Dateinamen verwendet,

die Sie für die Parameter **JSONTRANSFRM** und **JSON-SCHEMA** angegeben haben. Der Stapeljob erstellt auch die Sprachstruktur an der in den Parametern **PDSMEM** und **PDSLIB** angegebenen Position.

Installieren Sie die BUNDLE-Ressource, die diese JSON-Bindung angibt. Die JSONTRANSFRM-Bundleressource erstellt dynamisch eine JSONTRANSFRM-Ressource, die die Position des JSON-Schemas und der Bindungsdatei definiert. Diese Bundleressource ist in CICS sichtbar, wenn Sie die Inhalte des installierten Bundles mithilfe von CICS Explorer anzeigen. Es handelt sich nicht um eine reguläre CICS-Ressource und sie ist nicht sichtbar, wenn Sie CEMT oder die CICSplex SM-Webbenutzerschnittstelle verwenden.

Results

Ein CICS-Bundle, das eine JSON-Umsetzung enthält, wird generiert. Eine Sprachstruktur wird generiert.

Das folgende Beispiel zeigt DFHJS2LS mit dem angegebenen Mindestsatz von Parametern an.

```
//JS2LS JOB 'abrechnungsdaten',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
/*
```

What to do next

Schreiben Sie ein Anwendungsprogramm, um die Anwendungsdaten in JSON umzusetzen und umgekehrt. Sie können dieselben Zuordnungen für beide Umsetzungen verwenden.

Anwendungsdaten durch Verknüpfung mit DFHJSON in JSON umsetzen

Die verknüpfbare Schnittstelle des JSON-Umsetzungsprogramms, DFHJSON, ist ein von CICS bereitgestelltes Programm, das aufgerufen werden kann, um Umsetzungen zwischen Anwendungsdaten und JSON durchzuführen. Ihr Anwendungsprogramm kann Anwendungsdaten in JSON umsetzen, indem es eine Verknüpfung mit DFHJSON einrichtet.

Before you begin

Sie müssen über eine aktivierte JSONTRANSFRM-Bundleressource verfügen, die die JSON-Bindung und das JSON-Schema definiert. Wenn Sie Umsetzungen mithilfe von Java ausführen möchten, muss bereits ein Axis2-JVM-Server aktiv sein.

About this task

Erstellen oder aktualisieren Sie ein Anwendungsprogramm, das mit dem von CICS bereitgestellten Programm DFHJSON verknüpft wird, um die Umsetzung durchzuführen.

Procedure

1. Das Anwendungsprogramm muss einen Kanal erstellen, z. B. *mein_kanalname*, und die folgenden Container in dem Kanal platzieren.
 - DFHJSON-DATA
 - DFHJSON-TRANSFRM
 - DFHJSON-JVMSERVER (optional)

Weitere Informationen zu diesen Containern finden Sie unter JSON transformer linkable interface containers.

2. Verwenden Sie den **EXEC CICS LINK PROGRAM**-Befehl, um die Daten in JSON umzusetzen:

```
EXEC CICS LINK PROGRAM('DFHJSON') CHANNEL('mein_kanalname')
```
3. Rufen Sie den Container DFHJSON-ERROR ab und überprüfen Sie, ob während der Umsetzung Fehler aufgetreten sind.
4. Rufen Sie den Container DFHJSON-JSON ab und verwenden Sie die JSON-Daten in Ihrer Anwendung.
5. Installieren Sie die Anwendung.

Results

Wenn die Anwendung den Befehl **LINK** ausführt, prüft CICS die JSONTRANSFRM-Bundleressource, um die Zuordnungen in der JSON-Bindung zu finden, und setzt die Anwendungsbinärdaten unter Verwendung der Container im Kanal in JSON um. Die JSON-Daten werden wiederum in dem Container DFHJSON-JSON platziert. Die JSON entspricht dem JSON-Schema, das in der Bundleressource JSONTRANSFRM definiert ist.

What to do next

Sie können dieselben Zuordnungen auch verwenden, um JSON in Anwendungsdaten umzusetzen. Weitere Informationen finden Sie unter „JSON durch Verknüpfung mit DFHJSON in Anwendungsdaten umsetzen“ auf Seite 517.

Anwendungsdaten mit dem API-Befehl TRANSFORM DATATOJSON in JSON umsetzen

Sie können den API-Befehl **TRANSFORM DATATOJSON** in Ihrer Anwendung verwenden, um Anwendungsdaten in JSON umzusetzen.

Before you begin

Sie müssen über eine aktivierte JSONTRANSFRM-Ressource verfügen, die die JSON-Bindung und das JSON-Schema definiert. Wenn Sie Umsetzungen mithilfe von Java ausführen möchten, muss bereits ein Axis2-JVM-Server aktiv sein.

About this task

Die Anwendung muss eine kanalbasierte Schnittstelle verwenden.

Procedure

1. Erstellen Sie einen Kanal und platzieren Sie in dem Kanal einen Eingabecontainer, der die zu konvertierenden Anwendungsdaten enthält.

Anmerkung: Dieser Kanal wird außerdem über einen Ausgabecontainer verfügen, der die JSON-Ausgabe enthält, nachdem der Befehl **TRANSFORM DATATOJSON** ausgeführt wurde. Erstellen Sie den Ausgabecontainer nicht, bevor Sie den Befehl **TRANSFORM DATATOJSON** ausgeben, da der Container als Teil ebendieses Befehls erstellt und gefüllt wird.

2. Verwenden Sie den Befehl **TRANSFORM DATATOJSON** zum Umsetzen der Daten in JSON. Beispiel:

```
EXEC CICS TRANSFORM DATATOJSON CHANNEL(  
  kanalname  
) INCONTAINER(  
  eingabecontainername  
) OUTCONTAINER(  
  ausgabecontainername  
) TRANSFORMER(  
  bundlename  
)
```

Results

Wenn die Anwendung den Befehl **TRANSFORM DATATOJSON** ausführt, prüft CICS die JSONTRANSFRM-Bundleressource, um die Zuordnungen in der JSON-Bindung zu finden, und setzt die Anwendungsbinärdaten unter Verwendung der Container im Kanal in JSON um. Bei ihrer Rückgabe wird die JSON in dem Container gestellt, der in der Option **OUTCONTAINER** des Befehls **TRANSFORM DATATOJSON** angegeben ist. Ist die Option nicht angegeben, wird standardmäßig DFHJSON-JSON verwendet. Die JSON entspricht dem JSON-Schema, das in der Bundleressource JSONTRANSFRM definiert ist.

JSON durch Verknüpfung mit DFHJSON in Anwendungsdaten umsetzen

Die verknüpfbare Schnittstelle des JSON-Umsetzungsprogramms, DFHJSON, ist ein von CICS bereitgestelltes Programm, das aufgerufen werden kann, um Umsetzungen zwischen Anwendungsdaten und JSON durchzuführen. Ihr Anwendungsprogramm kann JSON in Anwendungsdaten umsetzen, indem es eine Verknüpfung mit DFHJSON einrichtet.

Before you begin

Sie müssen über eine aktivierte JSONTRANSFRM-Ressource verfügen, die die JSON-Bindung und das JSON-Schema definiert. Wenn Sie Umsetzungen mithilfe von Java ausführen möchten, muss bereits ein Axis2-JVM-Server aktiv sein.

About this task

Erstellen oder aktualisieren Sie ein Anwendungsprogramm, das mit dem von CICS bereitgestellten Programm DFHJSON verknüpft wird, um die Umsetzung durchzuführen.

Procedure

1. Erstellen Sie einen Kanal, z. B. *mein_kanalname* und platzieren Sie die folgenden Container in dem Kanal.

- DFHJSON-JSON
- DFHJSON-TRANSFRM
- DFHJSON-JVMSERVER

Weitere Informationen zu diesen Containern finden Sie unter JSON transformer linkable interface containers.

2. Verwenden Sie den API-Befehl **EXEC CICS LINK PROGRAM** , um die Daten in JSON umzusetzen:

```
EXEC CICS LINK PROGRAM('DFHJSON') CHANNEL('mein_kanalname')
```

3. Installieren Sie das Anwendungsprogramm.

Results

Wenn die Anwendung den Befehl **LINK PROGRAM** ausführt, prüft CICS die JSON-TRANSFRM-Ressource, um die Zuordnungen in der JSON-Bindung zu finden, und setzt die JSON-Daten unter Verwendung der Container im Kanal in die Anwendungsdaten um. Bei ihrer Rückgabe werden die Anwendungsdaten in den Bitcontainer DFHJSON-DATA gestellt.

What to do next

Sie können dieselben Zuordnungen auch verwenden, um Anwendungsdaten in JSON umzusetzen. Weitere Informationen finden Sie unter „Anwendungsdaten durch Verknüpfung mit DFHJSON in JSON umsetzen“ auf Seite 515.

JSON mithilfe des API-Befehls TRANSFORM JSONTODATA in Anwendungsdaten umsetzen

Sie können den API-Befehl **TRANSFORM JSONTODATA** in Ihrer Anwendung verwenden, um JSON in Anwendungsdaten umzusetzen.

Before you begin

Sie müssen über eine aktivierte JSONTRANSFRM-Ressource verfügen, die die JSON-Bindung und das JSON-Schema definiert. Wenn Sie Umsetzungen mithilfe von Java ausführen möchten, muss bereits ein Axis2-JVM-Server aktiv sein.

About this task

Die Anwendung muss eine kanalbasierte Schnittstelle verwenden.

Procedure

1. Erstellen Sie einen Kanal und platzieren Sie in dem Kanal einen Eingabecontainer, der die zu konvertierende JSON enthält.

Anmerkung: Dieser Kanal wird außerdem über einen Ausgabecontainer verfügen, der die konvertierten Daten enthält, nachdem der Befehl **TRANSFORM JSONTODATA** ausgeführt wurde. Erstellen Sie den Ausgabecontainer nicht, bevor Sie den Befehl **TRANSFORM JSONTODATA** ausgeben, da der Container als Teil eben-diesen Befehls erstellt und gefüllt wird.

2. Verwenden Sie den Befehl **TRANSFORM JSONTODATA** zum Umsetzen von JSON in Anwendungsdaten. Beispiel:

```
EXEC CICS TRANSFORM JSONTODATA CHANNEL(  
  kanalname  
) INCONTAINER(
```

```
    eingabecontainername
  ) OUTCONTAINER(
    ausgabecontainername
  ) TRANSFORMER(
    bundlename
  )
```

Results

Wenn die Anwendung den Befehl **TRANSFORM JSONTODATA** ausführt, prüft CICS die JSONTRANSFRM-Bundleressource, um die Zuordnungen in der JSON-Bindung zu finden, und setzt die JSON unter Verwendung der Container im Kanal in die Anwendungsbinärdaten um. Bei ihrer Rückgabe werden die konvertierten Daten in den Container gestellt, der in der Option **OUTCONTAINER** des Befehls **TRANSFORM JSONTODATA** angegeben ist. Ist die Option nicht angegeben, wird standardmäßig DFHJSON-DATA verwendet.

JSON-Web-Service-Clientanwendung erstellen

Sie können ein Anwendungsprogramm zum Aufrufen eines REST-konformen Web-Service schreiben, das die verknüpfbare Schnittstelle verwendet, um JSON umzusetzen, und die JSON anschließend mithilfe der WEB-Befehle an den fernen Service-Provider sendet.

Before you begin

Sie müssen mit der verknüpfbaren Schnittstelle vertraut sein, um JSON umzusetzen, wie unter Mapping and transforming application data and JSON beschrieben, und mit den EXEC CICS WEB-APIs, wie unter Making HTTP requests through CICS as an HTTP client beschrieben.

About this task

Möglicherweise möchten Sie als Teil einer CICS-Anwendung einen REST-konformen Web-Service aufrufen, der auf einem anderen System gehostet ist. Dazu müssen Sie zunächst die Daten beschreiben, die mit dem fernen Service ausgetauscht werden sollen. Anschließend können Sie ein Anwendungsprogramm schreiben, das über die EXEC CICS WEB-API mit dem fernen Service mithilfe des HTTP-Protokolls kommuniziert, um Anforderungsdaten an den Service zu senden und Antwortdaten zu empfangen. Dann können Sie die verknüpfbare Schnittstelle verwenden, um Ihre Anwendungsdaten in JSON umzusetzen, die als Teil der Anforderung verwendet werden soll, und um die JSON-Antwort in Anwendungsdaten umzusetzen. Manche Services unterstützen möglicherweise keine Nutzdaten sowohl für die Anforderung als auch für die Antwort.

Procedure

1. Definieren Sie die Schnittstelle für den fernen Service.
 - a. Wenn der ferne Service vorhanden ist, prüfen Sie, ob ein JSON-Schema verfügbar ist, das die Anforderungs- und Antwortnutzdaten beschreibt. Wenn keines vorhanden ist, müssen Sie eines erstellen. Verwenden Sie dann den JSON-Assistenten, um eine Zuordnung zu einer Sprachstruktur zu generieren. Weitere Informationen finden Sie unter Generating mappings from a JSON schema.
 - b. Wenn der ferne Service noch nicht vorhanden ist und Sie dessen Schnittstelle auf der Datenstruktur Ihrer Anwendung basieren lassen möchten, verwenden Sie den JSON-Assistenten, um ein JSON-Schema zu generieren.

Übergeben Sie anschließend das JSON-Schema an den Entwickler der fernen Serviceanwendung. Weitere Informationen finden Sie unter *Generating mappings from language structure*.

2. Definieren Sie eine BUNDLE-Ressource für das Bundle, das vom JSON-Assistenten generiert wird, und installieren Sie das Bundle in CICS.
3. Definieren Sie eine URIMAP-Ressource für den fernen Serviceendpunkt und installieren Sie sie. Weitere Informationen finden Sie unter *URIMAP resources*.
4. Erstellen oder aktualisieren Sie ein Anwendungsprogramm, um den fernen Service wie folgt aufzurufen:
 - a. Wenn für den fernen Service JSON-Nutzdaten für die Anforderung erforderlich sind (z. B. wenn die HTTP-Methode POST oder PUT ist), verwenden Sie die verknüpfbare Schnittstelle, um Ihre Anwendungsdaten in JSON umzusetzen. Weitere Informationen finden Sie unter *Transforming application data to JSON by linking to DFHJSON*.
 - b. Öffnen Sie eine Verbindung zu dem Server, auf dem der ferne Service gehostet ist, indem Sie den Befehl **EXEC CICS WEB OPEN** absetzen. Weitere Informationen finden Sie unter *WEB OPEN*.
 - c. Abhängig von den Anforderungen des Service sollten Sie den Befehl **EXEC CICS WEB WRITE HTTPHEADER** so codieren, dass er den 'Content-Type'-Header *application/JSON* angibt, um deutlich zu machen, dass JSON bereitgestellt wird. Weitere Informationen finden Sie unter *WEB WRITE HTTPHEADER*.
 - d. Codieren Sie einen Befehl **EXEC CICS WEB CONVERSE** so, dass er die Anforderung an den fernen Service sendet und die Antwort empfängt. Geben Sie bei Bedarf die Abfragezeichenfolge oder den Anforderungshauptteil (aus dem Container *DFHJSON-JSON*) an. Wenn Sie eine Antwort vom Service erwarten, geben Sie den Container *DFHJSON-JSON* an, um die Antwort-JSON zu empfangen. Weitere Informationen finden Sie unter *WEB CONVERSE*.
 - e. Wenn Sie nicht davon ausgehen, dass weitere Anforderungen gestellt werden, codieren Sie einen Befehl **EXEC CICS WEB CLOSE** zum Beenden der Verbindung. Weitere Informationen finden Sie unter *WEB CLOSE*.
 - f. Prüfen Sie den HTTP-Antwortcode, der von dem Befehl **EXEC CICS WEB CONVERSE** zurückgegeben wird, und ergreifen Sie entsprechende Maßnahmen, falls ein Fehler aufgetreten ist. Führen Sie die Anforderung beispielsweise erneut aus oder geben Sie einen Fehler an den Benutzer zurück.
 - g. Wenn ein Antworthauptteil vom fernen Service erwartet wurde, verwenden Sie die verknüpfbare Schnittstelle, um die JSON in Anwendungsdaten umzusetzen. Weitere Informationen finden Sie unter *Transforming application data to JSON by linking to DFHJSON*.

Results

Sie haben eine Anwendung erstellt, die einen REST-konformen Web-Service mit JSON-Nutzdaten aufrufen kann.

SOAP-Web-Service erstellen

Sie können vorhandene CICS-Anwendungen als SOAP-Web-Services verfügbar machen und neue CICS-Anwendungen erstellen, die als Provider oder Requester von SOAP-Web-Services fungieren.

Before you begin

Führen Sie vor dem Erstellen eines SOAP-Web-Service die folgenden Tasks aus:

1. Konfigurieren Sie Ihr CICS-System für die Unterstützung von Web-Services; siehe *Configuring your CICS system for web services*.
2. Erstellen Sie die notwendige Infrastruktur zur Unterstützung der Implementierung Ihrer Web-Services; siehe *Creating the web services infrastructure*.
3. Entscheiden Sie, ob Sie den Web-Service-Assistenten verwenden möchten; siehe *Planning to use SOAP web services*.

About this task

Der Web-Service-Assistent von CICS ist ein bereitgestelltes Dienstprogramm, das Sie beim Erstellen der erforderlichen Artefakte für eine neue Anwendung eines SOAP-Web-Service-Providers oder -Service-Requesters bzw. beim Aktivieren einer vorhandenen Anwendung als Web-Service-Provider unterstützt.

Der Web-Service-Assistent von CICS kann ein WSDL-Dokument aus einer einfachen Sprachstruktur oder eine Sprachstruktur aus einem vorhandenen WSDL-Dokument erstellen. Er unterstützt COBOL, C/C++ und PL/I. Er generiert außerdem Informationen, die zum Aktivieren der automatischen Laufzeitkonvertierung der SOAP-Nachrichten in Container und Kommunikationsbereiche und umgekehrt dienen. Diese Informationen werden von der CICS-Web-Service-Unterstützung während der Pipelineverarbeitung verwendet.

Erstellen Sie Ihren Web-Service wie in der folgenden Prozedur beschrieben und prüfen Sie, ob er korrekt funktioniert:

Procedure

1. Erstellen Sie einen SOAP-Web-Service auf eine von vier Arten:
 - Verwenden Sie den Web-Service-Assistenten, um die Web-Service-Beschreibung oder die Sprachstrukturen zu erstellen und sie in CICS zu implementieren. Verwenden Sie den Befehl **PIPELINE SCAN**, um automatisch die erforderlichen CICS-Ressourcen zu erstellen.
 - Verwenden Sie IBM Developer for Z oder die Java-API, um die Web-Service-Beschreibung oder die Sprachstrukturen zu erstellen und sie in CICS zu implementieren. Verwenden Sie den Befehl **PIPELINE SCAN**, um automatisch die erforderlichen CICS-Ressourcen zu erstellen.
 - Erstellen oder ändern Sie ein Anwendungsprogramm, das die XML in den eingehenden und ausgehenden Nachrichten verarbeitet, einschließlich Datenkonvertierung, und die korrekten Container in der Pipeline mit Daten füllt. Sie müssen die erforderlichen CICS-Ressourcen manuell erstellen.
 - Implementieren Sie eine Axis2-Anwendung als Web-Service.
2. Starten Sie den Web-Service, um zu testen, ob er wie gewünscht funktioniert. Wenn Sie Ihren Web-Service mithilfe des Web-Service-Assistenten implementieren, können Sie den Befehl **SET WEBSERVICE** verwenden, um die Validierung zu aktivieren. Diese Validierung prüft, ob die Daten korrekt konvertiert werden.

What to do next

Diese Schritte werden in den folgenden Themen ausführlicher erläutert.

CICS-Web-Service-Assistent

Der CICS-Web-Service-Assistent umfasst eine Gruppe von Stapeldienstprogrammen, die Sie dabei unterstützen, vorhandene CICS-Anwendungen in Web-Services umzusetzen und CICS-Anwendungen für die Verwendung von Web-Services vor-

zubereiten, die von externen Providern bereitgestellt werden. Der Assistent unterstützt eine schnelle Implementierung von CICS-Anwendungen für die Verwendung in Service-Providern und Service-Requestern, mit minimalem Programmieraufwand.

Wenn Sie den Web-Service-Assistenten für CICS verwenden, müssen Sie nicht Ihren eigenen Code für das Parsen von eingehenden Nachrichten und für das Zusammenstellen von ausgehenden Nachrichten schreiben. CICS ordnet Daten zwischen dem Textkörper einer SOAP-Nachricht und der Datenstruktur eines Anwendungsprogramms zu.

Der Assistent kann ein WSDL-Dokument aus einer einfachen Sprachstruktur oder aus einer Sprachstruktur eines vorhandenen WSDL-Dokuments erstellen. Er unterstützt COBOL, C/C++ und PL/I. Er generiert außerdem Informationen, die zum Aktivieren einer automatischen Laufzeitkonvertierung der SOAP-Nachrichten in Container und Kommunikationsbereiche und umgekehrt dienen.

Der CICS-Web-Service-Assistent umfasst zwei Dienstprogramme:

DFHLS2WS

Generiert eine Web-Service-Bindungsdatei aus einer Sprachstruktur. Dieses Dienstprogramm generiert außerdem eine Web-Service-Beschreibung.

DFHWS2LS

Generiert eine Web-Service-Bindungsdatei aus einer Web-Service-Beschreibung. Dieses Dienstprogramm generiert außerdem eine Sprachstruktur, die Sie in Ihren Anwendungsprogrammen verwenden können.

Die JCL-Prozeduren zum Ausführen beider Programme befinden sich in der Bibliothek *hlq* .XDFHINST .

Weitere Informationen zu den Dienstprogrammen und den Datenzuordnungen des Web-Service-Assistenten finden Sie in den folgenden Themen.

DFHLS2WS: Konvertierung einer höheren Programmiersprache in WSDL

Die Prozedur DFHLS2WS generiert aus einer Datenstruktur einer höheren Programmiersprache eine Web-Service-Beschreibung und eine Web-Service-Bindungsdatei. Sie können DFHLS2WS verwenden, wenn Sie ein CICS-Anwendungsprogramm als Service-Provider verfügbar machen.

Jobsteueranweisungen für DFHLS2WS

JOB Startet den Job.

EXEC Gibt den Prozedurnamen an (DFHLS2WS).

INPUT.SYSUT1 DD

Gibt die Eingabe an. Die Eingabeparameter werden in der Regel im Eingabedatenstrom angegeben. Sie können jedoch in einer Datei oder in einem Member einer partitionierten Datei definiert werden.

Symbolische Parameter

Die folgenden symbolischen Parameter sind in DFHLS2WS definiert:

JAVADIR = *pfad*

Gibt den Namen des Java-Verzeichnisses an, das von DFHLS2WS verwendet

wird. Der Wert dieses Parameters wird an `/usr/lpp/` angehängt, um den vollständigen Pfadnamen `/usr/lpp/pfad` zu erstellen.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **JAVADIR** angegeben wurde.

PATHPREF = *präfix*

Gibt ein optionales Präfix an, das den z/OS UNIX-Verzeichnispfad erweitert, der in anderen Parametern verwendet wird. Der Standardwert ist eine leere Zeichenfolge.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **PATHPREF** angegeben wurde.

SERVICE = *wert*

Verwenden Sie diesen Parameter nur, wenn sie vom IBM Support dazu aufgefordert werden.

TMPDIR = *tmp-verzeichnis*

Gibt die Position eines Verzeichnisses in z/OS UNIX an, das DFHLS2WS als temporären Arbeitsbereich verwendet. Die Benutzer-ID, unter der der Job ausgeführt wird, muss über Lese- und Schreibberechtigungen für dieses Verzeichnis verfügen.

Der Standardwert ist `/tmp`.

TMPFILE = *tmp-präfix*

Gibt ein Präfix an, das DFHLS2WS verwendet, um die Namen der temporären Arbeitsbereichsdateien zu erstellen.

Der Standardwert ist `LS2WS`.

USSDIR = *pfad*

Gibt den Namen des CICS TS-Verzeichnisses im UNIX System Services-Dateisystem an. Der Wert dieses Parameters wird an `/usr/lpp/cicsts/` angehängt, um den vollständigen Pfadnamen `/usr/lpp/cicsts/pfad` zu erstellen.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **USSDIR** angegeben wurde.

Der temporäre Arbeitsbereich

DFHLS2WS erstellt die folgenden drei temporären Dateien zur Laufzeit:

```
tmp-verzeichnis/tmp-präfix.in
tmp-verzeichnis/tmp-präfix.out
tmp-verzeichnis/tmp-präfix.err
```

Dabei gilt Folgendes:

tmp-verzeichnis ist der Wert, der im Parameter **TMPDIR** angegeben ist.

tmp-präfix ist der Wert, der im Parameter **TMPFILE** angegeben ist.

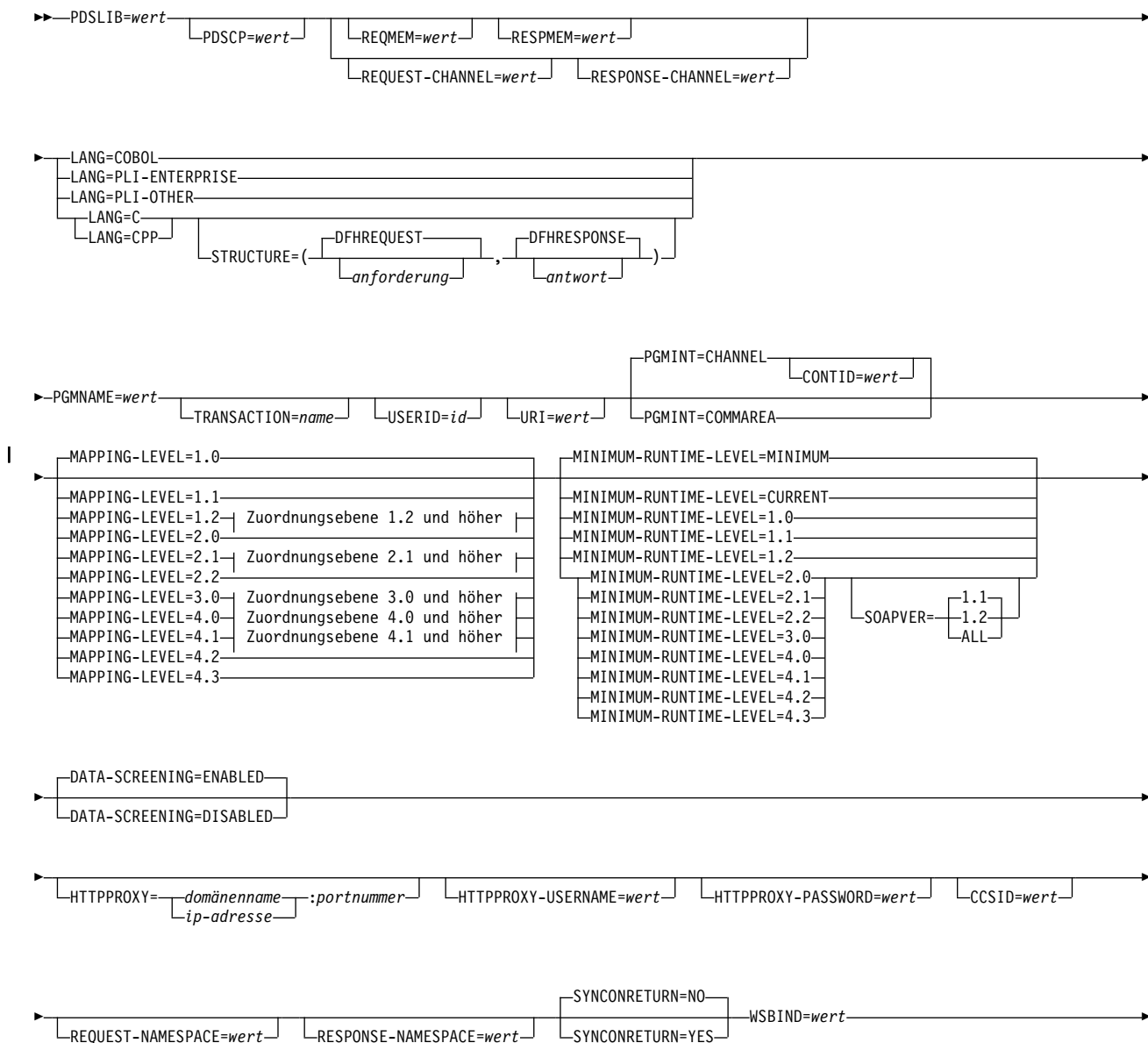
Die Standardnamen für die Dateien lauten wie folgt, wenn **TMPDIR** und **TMPFILE** nicht angegeben sind:

```
/tmp/LS2WS.in
/tmp/LS2WS.out
/tmp/LS2WS.err
```

Important: DFHLS2WS sperrt nicht den Zugriff auf die z/OS UNIX-Dateien oder -Dateimember. Wenn zwei oder mehr Instanzen von DFHLS2WS gleichzeitig ausgeführt werden und dieselben temporären Arbeitsbereichsdateien verwenden, kann nicht verhindert werden, dass ein Job die Arbeitsbereichsdateien überschreibt, während ein anderer Job sie verwendet, was zu unvorhersehbaren Fehlern führen kann.

Deshalb wird empfohlen, eine Namenskonvention und Betriebsprozeduren auszu-
arbeiten, die diese Situation verhindern. Sie können beispielsweise den symboli-
schen Systemparameter **SYSUID** verwenden, um Arbeitsbereichsdateinamen zu ge-
nerieren, die für einen einzelnen Benutzer eindeutig sind. Diese temporären
Dateien werden gelöscht, bevor der Job beendet wird.

Eingabeparameter für DFHLS2WS





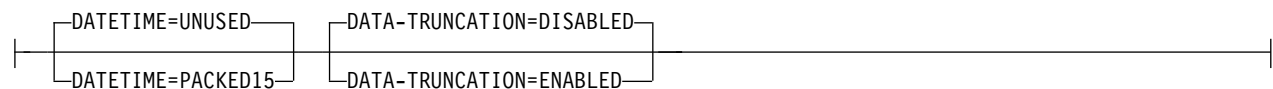
Zuordnungsebene 1.2 und höher:



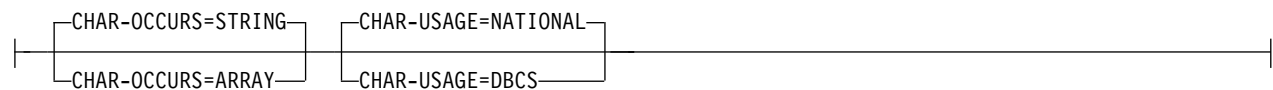
Zuordnungsebene 2.1 und höher:



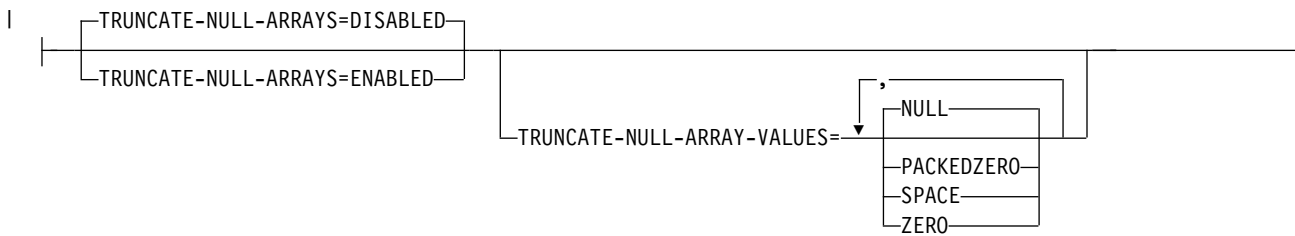
Zuordnungsebene 3.0 und höher:



Zuordnungsebene 4.0 und höher:



Zuordnungsebene 4.1 und höher:



Anmerkungen:

- 1 Jeder WSRR-Parameter, der angegeben werden kann, wenn der Parameter **WSRR-SERVER** festgelegt wird, kann nur ein einziges Mal angegeben werden. Die Ausnahme ist der Parameter **WSRR-CUSTOM**, den Sie maximal 255 Mal angeben können.

Parameterverwendung

- Sie können die Eingabeparameter in beliebiger Reihenfolge angeben.
- Jeder Parameter muss in einer neuen Zeile eingegeben werden.
- Ein Parameter (und ggf. das Fortsetzungszeichen) darf nicht über Spalte 72 hinausgehen; die Spalten 73-80 müssen Leerzeichen enthalten.
- Wenn ein Parameter zu lang ist, um in eine einzelne Zeile zu passen, verwenden Sie am Ende der Zeile einen Stern (*), um anzugeben, dass der Parameter in der nächsten Zeile fortgesetzt wird. Alles, einschließlich der Leerzeichen, vor dem Stern wird als Teil des Parameters betrachtet. Beispiel:

```
WSBIND=wsbinddir*  
      /app1
```

ist äquivalent zu

```
WSBIND=wsbinddir/app1
```

- Ein #-Zeichen an der ersten Zeichenposition der Zeile ist ein Kommentarzeichen. Die Zeile wird ignoriert.
- Ein Komma an der letzten Zeichenposition der Zeile ist ein optionales Zeilentrennzeichen und wird ignoriert.

Parameterbeschreibungen

BINDING-NAME = *wert*

Gibt den Bindungsnamen an, der in dem generierten WSDL-Dokument verwendet wird. Wenn kein Wert angegeben wird, wird ein Standardbindungsname mithilfe des Werts des Parameters **PGMNAME**, gefolgt von „HTTPSoapBinding“, generiert. Wenn SOAPVER auf ALL gesetzt ist, wird das Suffix „12“ an den Namen der SOAP 1.2-Bindung angehängt.

CCSID = *wert*

Gibt die CCSID an, die zur Laufzeit verwendet wird, um Zeichendaten in der Anwendungsdatenstruktur zu codieren. Der Wert dieses Parameters überschreibt den Wert des Systeminitialisierungsparameters **LOCALCCSID**. *wert* muss eine EBCDIC-CCSID sein, die von Java- und z/OS-Konvertierungsservices unterstützt wird (siehe z/OS Unicode Services User's Guide and Reference). Wenn Sie diesen Parameter nicht angeben, wird die Anwendungsdatenstruktur unter Verwendung der im Systeminitialisierungsparameter angegebenen CCSID codiert.

Sie können diesen Parameter mit jeder Zuordnungsebene verwenden.

CHAR-VARYING = { NO | **NULL** | **COLLAPSE** | **BINARY** }

Gibt an, wie Zeichenfelder in der Sprachstruktur auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Ein Zeichenfeld in COBOL ist eine Picture-Klausel vom Typ X, z. B. PIC(X) 10; ein Zeichenfeld in C/C++ ist ein Zeichenarray. Sie haben folgende Optionen:

NO Zeichenfelder werden einem <xsd:string>-Element zugeordnet und als Felder mit fester Länge verarbeitet. Die maximale Länge der Daten ist gleich der Länge des Felds. NO ist der Standardwert für den Parameter **CHAR-VARYING** für COBOL und PL/I auf den Zuordnungsebenen 2.0 und niedriger.

Dieser Wert gilt nicht für Enterprise PL/I und andere PL/I-Sprachstrukturen.

NULL Zeichenfelder werden einem <xsd:string>-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet. CICS fügt bei der Umsetzung aus einer SOAP-Nachricht ein abschließendes Nullzeichen (NULL) hinzu. Die maximale Länge der Zeichenfolge wird als ein Zeichen weniger als die in der Sprachstruktur angegebene Länge berechnet. NULL ist der Standardwert für den Parameter **CHAR-VARYING** für C/C++.

Dieser Wert gilt nicht für Enterprise PL/I und andere PL/I-Sprachstrukturen.

COLLAPSE

Zeichenfelder werden einem <xsd:string>-Element zugeordnet. Abschließende und eingebettete Leerzeichen in dem Feld werden nicht in die SOAP-Nachricht eingeschlossen, z. B. wird
<leerzeichen>AB<leerzeichen><leerzeichen><leerzeichen>C<leerzeichen>
zu AB<leerzeichen>C. Die eingehende SOAP-Nachricht wird geparkt, um alle führenden, abschließenden und eingebetteten Leerzeichen zu entfernen. COLLAPSE ist der Standardwert für den Parameter **CHAR-VARYING** für COBOL und PL/I ab Zuordnungsebene 2.1.

Weitere Informationen zu Werten mit variabler Länge und Leerzeichen finden Sie unter Unterstützung für Werte mit variabler Länge und für Leerzeichen.

BINARY

Zeichenfelder werden einem <xsd:base64binary>-Element zugeordnet und als Felder mit fester Länge verarbeitet. Der BINARY-Wert im Parameter **CHAR-VARYING** ist nur ab Zuordnungsebene 2.1 verfügbar.

CHAR-OCCURS = { STRING | **ARRAY** }

Gibt an, wie Zeichenarrays in der Sprachstruktur auf Zuordnungsebene 4.0 oder höher zugeordnet werden. Beispiel: PIC X OCCURS 20. Dieser Parameter kann nur in der COBOL-Sprache verwendet werden.

ARRAY

Zeichenarrays werden einem XML-Array zugeordnet. Das heißt, dass jedes Zeichen als einzelnes XML-Element zugeordnet wird. Dies ist auch das Verhalten auf Zuordnungsebene 3.0 und niedriger.

STRING

Zeichenarrays werden einer XML-Zeichenfolge zugeordnet. Das heißt, dass das gesamte COBOL-Array als einzelnes XML-Element zugeordnet wird.

CHAR-USAGE = { NATIONAL | **DBCS** }

In COBOL kann der nationale Datentyp, PIC N, für UTF-16- oder DBCS-Daten verwendet werden. Diese Einstellung wird durch die Compileroption NSYMBOL gesteuert. Sie müssen den Parameter **CHAR-USAGE** im Assistenten auf denselben Wert setzen wie die Compileroption NSYMBOL, um sicherzustellen, dass die Daten richtig verarbeitet werden. Dieser Wert wird in der Regel auf CHAR-USAGE=NATIONAL gesetzt, wenn Sie UTF-16 verwenden.

DBCS Daten aus PIC(*n*)-Feldern werden als DBCS-codierte Daten behandelt.

NATIONAL

Daten aus PIC(*n*)-Feldern werden als UTF-16-codierte Daten behandelt.

CONTID = *wert*

Gibt in einem Service-Provider den Namen des Containers an, der die übergeordnete Datenstruktur enthält, die zur Darstellung einer SOAP-Nachricht verwendet wird.

Die Länge des Containers, die CICS an das Ziellanwendungsprogramm übergibt, ist die größere der beiden Längen des Anforderungscontainers bzw. des Antwortcontainers.

DATA-SCREENING = { ENABLED | DISABLED }

Gibt an, ob die von Anwendungen bereitgestellten Daten auf Fehler überprüft werden.

ENABLED

Alle von Anwendungen bereitgestellten Laufzeitdaten, die nicht mit der Sprachstruktur konsistent sind, werden als Fehler behandelt, und es wird die Nachricht DFHPI1010 ausgegeben. Eine Fehlerantwort wird an die Anwendung zurückgegeben.

DISABLED

Werte in den von Anwendungen bereitgestellten Laufzeitdaten, die nicht konsistent mit der Sprachstruktur sind, werden durch Standardwerte ersetzt. Eine Null ersetzt beispielsweise einen fehlerhaften Wert in einem numerischen Feld. Die Nachricht DFHPI1010 wird nicht ausgegeben und es wird eine normale Antwort an die Anwendung zurückgegeben. Dieses Feature kann verwendet werden, um die Fehlerantworten INVALID_PACKED_DEC und INVALID_ZONED_DEC zu vermeiden, die von nicht initialisierten Ausgabefeldern generiert werden.

DATA-TRUNCATION = { DISABLED | ENABLED }

Gibt an, ob Daten mit variabler Länge in einer Feldstruktur mit fester Länge toleriert werden:

DISABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, weist CICS die abgeschnittenen Daten zurück und gibt eine Fehlermeldung aus.

ENABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, toleriert CICS die abgeschnittenen Daten und verarbeitet die fehlenden Daten als Nullwerte.

DATETIME = { UNUSED | PACKED15 }

Gibt an, ob potenzielle ABSTIME-Felder in der Struktur einer höheren Programmiersprache als Zeitmarken zugeordnet werden:

PACKED15

Gepackte Dezimalfelder mit einer Länge von 15 (8 Byte) werden als CICS ABSTIME-Felder behandelt und als Zeitmarken zugeordnet.

UNUSED

Gepackte Dezimalfelder mit einer Länge von 15 (8 Byte) werden nicht als Zeitmarken behandelt.

Sie können diesen Parameter auf der Zuordnungsebene 3.0 festlegen.

HTTPPROXY = { *domänenname* : *portnummer* | *ip-adresse* : *portnummer* }

Wenn Ihre WSDL Referenzen auf andere WSDL-Dateien enthält, die sich im Internet befinden, und das System, auf dem DFHLS2WS ausgeführt wird, einen Proxy-Server für den Zugriff auf das Internet verwendet, geben Sie den Domännennamen oder die IP-Adresse und die Portnummer des Proxy-Servers an. Beispiel:

HTTPPROXY=proxy.example.com:8080

In anderen Fällen ist dieser Parameter nicht erforderlich.

HTTPPROXY-PASSWORD = *wert*

Gibt das HTTP-Proxy-Kennwort an, das mit **HTTPPROXY-USERNAME** verwendet werden muss, wenn das System, auf dem DFHLS2WS ausgeführt wird, einen HTTP-Proxy-Server für den Zugriff auf das Internet verwendet und wenn der HTTP-Proxy-Server Basisauthentifizierung verwendet. Sie können diesen Parameter nur verwenden, wenn Sie auch **HTTPPROXY** angeben.

HTTPPROXY-USERNAME = *wert*

Gibt den HTTP-Proxy-Benutzernamen an, der mit **HTTPPROXY-PASSWORD** verwendet werden muss, wenn das System, auf dem DFHLS2WS ausgeführt wird, einen HTTP-Proxy-Server für den Zugriff auf das Internet verwendet und wenn der HTTP-Proxy-Server Basisauthentifizierung verwendet. Sie können diesen Parameter nur verwenden, wenn Sie auch **HTTPPROXY** angeben.

LANG = **COBOL**|**PLI-ENTERPRISE**|**PLI-OTHER**|**C**|**CPP**

Gibt die Datenstruktur der höheren Programmiersprache an:

COBOL

COBOL

PLI-ENTERPRISE

Enterprise PL/I

PLI-OTHER

Andere PL/I-Ebene als Enterprise PL/I

C **C**

CPP **C++**

LOGFILE = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die DFHLS2WS das Aktivitätenprotokoll und die Traceinformationen schreibt. DFHLS2WS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur.

In der Regel verwenden Sie diese Datei nicht, aber dies kann von der IBM Serviceorganisation angefordert werden, wenn Sie Probleme mit DFHLS2WS feststellen.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 }

Gibt die Zuordnungsebene an, die DFHLS2WS bei der Generierung der Web-Service-Bindungsdatei und der Web-Service-Beschreibung verwendet. Sie haben folgende Optionen:

- 1.0** Dies ist die Standardzuordnungsebene. Sie gibt an, dass die Web-Service-Bindungsdatei mithilfe von CICS TS 3.1-Zuordnungsebenen generiert wird.
- 1.1** Verwenden Sie diese Zuordnung, um eine Bindungsdatei auf dieser spezifischen Ebene neu zu generieren.
- 1.2** Auf dieser Zuordnungsebene können Sie den Parameter **CHAR-VARYING** verwenden, um zu steuern, wie Zeichenarrays zur Laufzeit verarbeitet werden. Die Arrays **VARYING** und **VARYINGZ** werden auch in PL/I unterstützt.
- 2.0** Verwenden Sie diese Zuordnungsebene in einer Region in CICS TS 3.2 oder höher, um die Erweiterungen der Zuordnung zwischen der Sprachstruktur und der Web-Service-Bindungsdatei zu nutzen.
- 2.1** Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 3.2 oder höher. Auf dieser Zuordnungsebene können Sie die neuen Werte für den Parameter **CHAR-VARYING**, **COLLAPSE** und **BINARY** nutzen. **FILLER**-Felder in COBOL und *****-Felder in PL/I werden auf dieser Zuordnungsebene systematisch ignoriert, die Felder werden nicht im generierten WSDL-Dokument angezeigt und eine entsprechende Lücke wird in den Datenstrukturen zur Laufzeit beibehalten.
- 2.2** Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 3.2 oder höher, um bei Verwendung von DFHWS2LS Zuordnungserweiterungen zu nutzen.
- 3.0** Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 4.1 oder höher. Auf dieser Zuordnungsebene können Sie einen Web-Service aus einer Anwendung erstellen, die viele Container in ihrer Schnittstelle verwendet, indem Sie die Parameter **REQUEST-CHANNEL** und **RESPONSE-CHANNEL** festlegen. Sie können auch Datum/Zeit-Felder zu XML-Zeitmarken zuordnen, indem Sie den Parameter **DATETIME** festlegen.
- 4.0** Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher. Auf dieser Zuordnungsebene können Sie **OCCURS DEPENDING ON**-COBOL-Felder und den Parameter **CHAR-OCCURS** verwenden.
- 4.1** Zur Unterstützung abschneidbarer Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher.
- 4.2** Keine signifikanten Änderungen. Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.
- 4.3** Keine signifikanten Änderungen. Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.

Weitere Informationen zu Zuordnungsebenen finden Sie unter Zuordnungsebenen für die CICS-Assistenten.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 | CURRENT }

Gibt die älteste CICS-Laufzeitumgebung an, in der die Web-Service-Bindungs-

datei implementiert werden kann. Wenn Sie eine Version auswählen, die nicht mit den anderen Parametern übereinstimmt, die Sie angegeben haben, erhalten Sie eine Fehlermeldung. Sie haben folgende Optionen:

MINIMUM

Die niedrigste mögliche Laufzeitebene von CICS wird anhand der von Ihnen ausgewählten Parameter automatisch zugewiesen.

- 1.0 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS TS 3.1-Region implementiert. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 1.1 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS TS 3.1-Region implementiert. Sie können die Zuordnungsebene 1.1 oder niedriger für den Parameter **MAPPING-LEVEL** verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 1.2 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS TS 3.1-Region implementiert. Sie können die Zuordnungsebene 1.2 oder niedriger für den Parameter **MAPPING-LEVEL** verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 2.0 Die generierte Web-Service-Bindungsdatei wird erfolgreich in CICS TS 3.2 oder einer höheren Region implementiert. Sie können die Zuordnungsebene 2.0 oder niedriger für den Parameter **MAPPING-LEVEL** verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 2.1 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 3.2 oder höher implementiert. Sie können die Zuordnungsebene 2.1 oder niedriger für den Parameter **MAPPING-LEVEL** verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 2.2 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 3.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 2.2 oder niedriger verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 3.0 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 4.1 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 3.0 oder niedriger verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 4.0 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.0 oder niedriger verwenden. Sie können alle optionalen Parameter auf dieser Ebene verwenden.
- 4.1 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.1 oder niedriger verwenden.
- 4.2 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS V5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.2 oder niedriger verwenden.

4.3 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.3 oder niedriger verwenden.

CURRENT

Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS-Region auf derselben Laufzeitebene implementiert, die Sie zum Generieren der Web-Service-Bindungsdatei verwenden.

OPERATION-NAME = *wert*

Gibt den Operationsnamen an, der in dem generierten WSDL-Dokument verwendet wird. Wenn kein Wert angegeben wird, wird ein Standardname mithilfe des Werts des Parameters **PGMNAME**, gefolgt von dem Wert **operation**, generiert.

PDSLIB = *wert*

Gibt den Namen der partitionierten Datei an, die die zu verarbeitenden Datenstrukturen der höheren Programmiersprachen enthält. Die Dateimember, die für die Anforderung und Antwort verwendet werden, sind in den Parametern **REQMEM** bzw. **RESPMEM** angegeben.

Restriction: Die Datensätze in der partitionierten Datei müssen eine feste Länge von 80 Bytes haben.

PDSCP = *wert*

Gibt die Codepage an, die in den Mitgliedern einer partitionierten Datei verwendet wird, die in den Parametern **REQMEM** und **RESPMEM** angegeben sind. Dabei ist *wert* eine CCSID-Nummer oder eine Java-Codepagenummer. Wenn dieser Parameter nicht angegeben wird, wird die Codepage von z/OS UNIX System Services verwendet. Beispielsweise können Sie **PDSCP** = 037 angeben.

PGMINT = { **CHANNEL** | **COMMAREA** }

Gibt für einen Service-Provider an, wie CICS Daten an das Ziellanwendungsprogramm übergibt.

CHANNEL

CICS verwendet eine Kanalschnittstelle, um Daten an das Ziellanwendungsprogramm zu übergeben.

- Auf Zuordnungsebenen niedriger als 3.0 kann der Kanal nur einen Container enthalten, der sowohl für Eingaben als auch für Ausgaben verwendet wird. Mit dem Parameter **CONTID** geben Sie den Namen des Containers an. Der Standardname ist DFHWS-DATA.
- Auf Zuordnungsebene 3.0 kann der Kanal mehrere Container enthalten. Verwenden Sie die Parameter **REQUEST-CHANNEL** und **RESPONSE-CHANNEL**. Geben Sie nicht **PDSLIB**, **REQMEM** oder **RESPMEM** an.

COMMAREA

CICS verwendet einen Kommunikationsbereich (COMMAREA), um Daten an das Ziellanwendungsprogramm zu übergeben.

Wenn das Ziellanwendungsprogramm die Anforderung verarbeitet hat, muss es denselben Mechanismus verwenden, um die Antwort zurückzugeben. Wenn die Anforderung in einem Kommunikationsbereich empfangen wurde, muss die Antwort im Kommunikationsbereich zurückgegeben werden. Wenn die Anforderung in einem Container empfangen wurde, muss die Antwort in einem Container zurückgegeben werden. Die Länge des Kommunikationsbereichs oder des Containers, die CICS an das Ziellanwendungsprogramm zurückgibt,

ist größer als die Länge des Anforderungskommunikationsbereichs oder -containers und des Antwortkommunikationsbereichs oder -containers.

PGMNAME = *wert*

Gibt den Namen der PROGRAM-Ressource von CICS für das Zielanwendungsprogramm an, das als Web-Service zugänglich gemacht wird. Der CICS-Web-Service-Support stellt eine Verknüpfung zu diesem Programm bereit.

PORT-NAME = *wert*

Gibt den Namen an, der für den Port und den Porttyp (portType) in dem generierten WSDL-Dokument verwendet wird. Wenn kein Wert angegeben wird, wird ein Standardname mithilfe des Werts des Parameters **PGMNAME**, gefolgt von „Port“, generiert. Wenn SOAPVER auf ALL gesetzt ist, wird das Suffix „12“ an den Namen des SOAP 1.2-Ports angehängt.

REQMEM = *wert*

Gibt den Namen des Members der partitionierten Datei an, die die Struktur der höheren Programmiersprache für die Web-Service-Anforderung enthält. Für einen Service-Provider ist die Web-Service-Anforderung die Eingabe für das Anwendungsprogramm.

REQUEST-CHANNEL = *wert*

Gibt den Namen und die Position eines Kanalbeschreibungsdokuments an. In der Kanalbeschreibung werden die Container beschrieben, die die Web-Service-Provider-Anwendung in ihrer Schnittstelle verwenden kann, wenn sie eine SOAP-Nachricht von einem Web-Service-Requester empfängt. Die Kanalbeschreibung ist ein XML-Dokument, das mit dem von CICS bereitgestellten Kanalschema übereinstimmen muss.

Sie können diesen Parameter nur auf Zuordnungsebene 3.0 verwenden.

REQUEST-NAMESPACE = *wert*

Gibt den Namensbereich des XML-Schemas für die Anforderungsnachricht in der generierten Web-Service-Beschreibung an. Wenn Sie diesen Parameter nicht angeben, generiert CICS automatisch einen Namensbereich.

RESPMEM = *wert*

Gibt den Namen des Members der partitionierten Datei an, die die Struktur der höheren Programmiersprache für die Web-Service-Antwort enthält. Für einen Service-Provider ist die Web-Service-Antwort die Ausgabe des Anwendungsprogramms.

Lassen Sie diesen Parameter weg, wenn keine Antwort erwartet wird, d.h. bei unidirektionalen Nachrichten.

RESPONSE-CHANNEL = *wert*

Gibt den Namen und die Position eines Kanalbeschreibungsdokuments an. In der Kanalbeschreibung werden die Container beschrieben, die die Web-Service-Provider-Anwendung in ihrer Schnittstelle verwenden kann, wenn sie eine SOAP-Nachricht an einen Web-Service-Requester sendet. Die Kanalbeschreibung ist ein XML-Dokument, das mit dem von CICS bereitgestellten Kanalschema übereinstimmen muss.

Sie können diesen Parameter nur auf Zuordnungsebene 3.0 verwenden.

RESPONSE-NAMESPACE = *wert*

Gibt den Namensbereich des XML-Schemas für die Antwortnachricht in der generierten Web-Service-Beschreibung an. Wenn Sie diesen Parameter nicht angeben, generiert CICS automatisch einen Namensbereich.

SERVICE-NAME = *wert*

Gibt den Servicenamen an, der in dem generierten WSDL-Dokument verwen-

det wird. Wenn kein Wert angegeben wird, wird ein Standard servicename mithilfe des Werts des Parameters **PGMNAME**, gefolgt von „Service“, generiert.

SOAPVER = { 1.1 | 1.2 | ALL }

Gibt die SOAP-Version an, die in der generierten Web-Service-Beschreibung verwendet werden soll. Dieser Parameter ist nur verfügbar, wenn **MINIMUM-RUNTIME-LEVEL** auf 2.0 oder höher gesetzt ist.

- 1.1** Das SOAP 1.1-Protokoll wird als Bindung für die Web-Service-Beschreibung verwendet.
- 1.2** Das SOAP 1.2-Protokoll wird als Bindung für die Web-Service-Beschreibung verwendet.
- ALL** Sowohl das SOAP 1.1- als auch das 1.2-Protokoll können als Bindung für die Web-Service-Beschreibung verwendet werden.

Wenn Sie keinen Wert für diesen Parameter angeben, hängt der Standardwert von der WSDL-Version ab, die Sie erstellen möchten.

- Wenn Sie nur WSDL 1.1 benötigen, wird die SOAP 1.1-Bindung verwendet.
- Wenn Sie nur WSDL 2.0 benötigen, wird die SOAP 1.2-Bindung verwendet.
- Wenn Sie sowohl WSDL 1.1 als auch WSDL 2.0 verwenden, werden sowohl SOAP 1.1- als auch 1.2-Bindungen für die einzelnen Web-Service-Beschreibungen verwendet.

SSL-KEYSTORE = *wert*

Dieser optionale Parameter gibt die vollständig qualifizierte Position der Schlüsselspeicherdatei an.

Verwenden Sie diesen Parameter, wenn der Web-Service-Assistent SSL-Verschlüsselung (Secure Sockets Layer) verwenden soll, um über ein Netz mit einem IBM WebSphere Service Registry and Repository (WSRR) zu kommunizieren.

SSL-KEYPWD = *wert*

Dieser optionale Parameter gibt das Kennwort für den Schlüsselspeicher an.

Verwenden Sie diesen Parameter, wenn der Web-Service-Assistent SSL-Verschlüsselung (Secure Sockets Layer) verwenden soll, um über ein Netz mit einem IBM WebSphere Service Registry and Repository (WSRR) zu kommunizieren.

SSL-TRUSTSTORE = *wert*

Dieser optionale Parameter gibt die vollständig qualifizierte Position der Truststore-Datei an.

Verwenden Sie diesen Parameter, wenn der Web-Service-Assistent SSL-Verschlüsselung (Secure Sockets Layer) verwenden soll, um über ein Netz mit einem IBM WebSphere Service Registry and Repository (WSRR) zu kommunizieren.

SSL-TRUSTPWD = *wert*

Dieser optionale Parameter gibt das Kennwort für den Truststore an.

Verwenden Sie diesen Parameter, wenn der Web-Service-Assistent SSL-Verschlüsselung (Secure Sockets Layer) verwenden soll, um über ein Netz mit einem IBM WebSphere Service Registry and Repository (WSRR) zu kommunizieren.

STRUCTURE = (*anforderung* , *antwort*)

Gibt nur für C und C++ die Namen der übergeordneten Strukturen an, die in

den Mitgliedern der partitionierten Dateien enthalten sind, die wiederum in den Parametern **REQMEM** und **RESPMEM** angegeben sind:

anforderung

Gibt den Namen der übergeordneten Struktur an, die die Anforderung enthält, wenn der Parameter **REQMEM** angegeben wird. Der Standardwert ist **DFHREQUEST**.

Das Mitglied der partitionierten Datei muss eine übergeordnete Struktur mit dem von Ihnen angegebenen Namen oder eine Struktur namens **DFHREQUEST** enthalten, wenn Sie keinen Namen angeben.

antwort

Gibt den Namen der übergeordneten Struktur an, die die Antwort enthält, wenn der Parameter **RESPMEM** angegeben wird. Der Standardwert ist **DFHRESPONSE**.

Wenn Sie einen Wert angeben, muss das Mitglied der partitionierten Datei eine übergeordnete Struktur mit dem von Ihnen angegebenen Namen oder eine Struktur namens **DFHRESPONSE** enthalten, wenn Sie keinen Namen angeben.

SYNCONRETURN = { **NO** | **YES** }

Gibt an, ob der ferne Web-Service einen Synchronisationspunkt absetzen kann.

NO Der ferne Web-Service kann keinen Synchronisationspunkt absetzen. Dies ist der Standardwert. Wenn der ferne Web-Service einen Synchronisationspunkt absetzt, schlägt er mit einem ADPL-Abbruch fehl.

YES Der ferne Web-Service kann einen Synchronisationspunkt absetzen. Wenn Sie **YES** auswählen, wird die ferne Task als separate Arbeitseinheit festgeschrieben, wenn die Steuerung vom fernen Web-Service zurückgegeben wird. Wenn der ferne Web-Service eine wiederherstellbare Ressource aktualisiert und ein Fehler auftritt, nachdem sie zurückgegeben wurde, kann die Aktualisierung dieser Ressource nicht zurückgesetzt werden.

TRANSACTION = *name*

In einem Service-Provider gibt dieser Parameter den ein bis vier Zeichen langen Namen einer Aliastransaktion an, die die Pipeline starten kann. Der Wert dieses Parameters wird verwendet, um das Attribut **TRANSACTION** der URIMAP-Ressource zu definieren, wenn es automatisch mithilfe des **PIPELINE**-Scanfehls erstellt wird.

Zulässige Zeichen:

A-Z a-z 0-9 \$ @ # _ < >

TRUNCATE-NULL-ARRAYS = { **DISABLED** | **ENABLED** }

Gibt an, wie strukturierte Arrays auf Zuordnungsebene 4.1 oder höher verarbeitet werden. Wenn diese Option aktiviert ist, versucht CICS, leere Datensätze innerhalb eines Arrays zu erkennen (weitere Informationen zur Identifizierung leerer Datensätze finden Sie unter **TRUNCATE-NULL-ARRAY-VALUES**). Wenn fünf aufeinanderfolgende leere Arraydatensätze erkannt werden, wird das Array bei der Generierung von XML/JSON beim ersten solchen Datensatz abgeschnitten. Diese Abschneidefunktion ist nur für Arrays mit strukturiertem Inhalt aktiviert, Arrays aus einfachen primitiven Feldern werden nicht abgeschnitten. Das Abschneiden von Arrays kann zu einer kürzeren Darstellung der Daten in JSON/XML führen, dies ist jedoch nicht ohne Risiko. Wenn fünf aufeinanderfolgende Datensätze fehlerhaft als nicht initialisierter Speicher

identifiziert werden (beispielsweise weil sie legitim niedrige Werte enthalten), kann es zu Datenverlusten kommen. Wenn TRUNCATE-NULL-ARRAYS aktiviert und TRUNCATE-NULL-ARRAY-VALUES nicht festgelegt ist, wird der Standardwert für TRUNCATE-NULL-ARRAY-VALUES verwendet.

TRUNCATE-NULL-ARRAY-VALUES = { NULL | **PACKEDZERO**|**SPACE** | **ZERO** }

Gibt an, welche Werte für die Verarbeitung von TRUNCATE-NULL-ARRAYS auf Zuordnungsebene 4.1 oder höher als leer behandelt werden. Standardmäßig wird der Nullwert ('0x00' oder 'low-values') als leer behandelt. Wenn alle Bytes des Speichers in einem Datensatz eines strukturierten Arrays Nullen enthalten, wird der gesamte Datensatz als leer betrachtet. Ein Wert oder mehrere Werte vom Typ NULL, PACKEDZERO, SPACE und ZERO können in einer durch Kommas getrennten Liste angegeben werden.

NULL Impliziert ein Nullzeichen (0x00).

PACKEDZERO

Impliziert eine gepackte Dezimalnull mit positivem Vorzeichen (0x0C), eine gepackte Dezimalnull mit negativem Vorzeichen (0x0D) oder eine gepackte Dezimalnull ohne Vorzeichen (0x0F).

SPACE

Impliziert einen SBCS EBCDIC-Bereich (0x40).

ZERO Impliziert eine gezonte Dezimalnull ohne Vorzeichen (0xF0).

Jede passende Kombination der ausgewählten Bytes innerhalb eines strukturierten Arraydatensatzes führt dazu, dass der gesamte Datensatz als leer erkannt wird.

Wenn für TRUNCATE-NULL-ARRAY-VALUES ein Wert definiert ist, muss TRUNCATE-NULL-ARRAYS aktiviert sein.

URI = *wert*

Dieser Parameter gibt den relativen oder absoluten URI an, den ein Client für den Zugriff auf den Web-Service verwendet. CICS verwendet den angegebenen Wert, wenn eine URIMAP-Ressource aus der Web-Service-Bindungsdatei generiert wird, die von DFHLS2WS erstellt wird. Der Parameter gibt die Pfadkomponente des URI an, für den die URIMAP-Definition gilt.

USERID = *id*

In einem Service-Provider gibt dieser Parameter eine ein bis acht Zeichen lange Benutzer-ID an, die von jedem Web-Client verwendet werden kann. Für eine anwendungsgenerierte Antwort oder einen Web-Service wird die Aliastransaktion unter dieser Benutzer-ID angehängt. Der Wert dieses Parameters wird verwendet, um das Attribut USERID der URIMAP-Ressource zu definieren, wenn es automatisch mithilfe des **PIPELINE**-Scanbefehls erstellt wird.

Zulässige Zeichen:

A-Z a-z 0-9 \$ @ #

WSBIND = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Web-Service-Bindungsdatei. DFHLS2WS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur. Die Dateierweiterung ist .wsbind.

WSDL = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die die Web-Service-Beschreibung geschrieben wird. Die Web-Service-Beschreibung entspricht

der WSDL 1.1-Spezifikation. DFHLS2WS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur. Die Dateierweiterung ist `.wsdl`.

WSDL_1.1 = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die die Web-Service-Beschreibung geschrieben wird. Die Web-Service-Beschreibung entspricht der WSDL 1.1-Spezifikation. DFHLS2WS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur. Die Dateierweiterung ist `.wsdl`. Dieser Parameter erzeugt dasselbe Ergebnis wie der Parameter **WSDL**, sie können also nur einen der beiden angeben.

WSDL_2.0 = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die die Web-Service-Beschreibung geschrieben wird. Die Web-Service-Beschreibung entspricht der WSDL 2.0-Spezifikation. DFHLS2WS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur. Die Dateierweiterung ist `.wsdl`. Dieser Parameter kann mit den Parametern **WSDL** oder **WSDL_1.1** verwendet werden. Er ist nur verfügbar, wenn **MINIMUM-RUNTIME-LEVEL** auf 2.0 oder höher gesetzt ist.

WSDLCP = { LOCAL | **UTF-8** | **EBCDIC-CP-US** }

Gibt die Codepage an, die zum Generieren des WSDL-Dokuments verwendet wird.

LOCAL

Gibt an, dass das WSDL-Dokument mithilfe der lokalen Codepage generiert wird und dass kein Codierungstag im WSDL-Dokument generiert wird.

UTF-8 Gibt an, dass das WSDL-Dokument unter Verwendung der UTF-8-Codepage generiert wird. Ein Codierungstag wird im WSDL-Dokument generiert. Wenn Sie diese Option angeben, müssen Sie sicherstellen, dass die Codierung weiterhin korrekt ist, wenn Sie das WSDL-Dokument zwischen verschiedenen Plattformen kopieren.

EBCDIC-CP-US

Dieser Wert gibt an, dass das WSDL-Dokument unter Verwendung der US EBCDIC-Codepage generiert wird. Ein Codierungstag wird im WSDL-Dokument generiert.

WSDL-NAMESPACE = *wert*

Gibt den Namensbereich für CICS an, der in dem generierten WSDL-Dokument verwendet werden soll.

Wenn Sie diesen Parameter nicht angeben, generiert CICS automatisch einen Namensbereich.

WSRR-CUSTOM-*eigenschaftsname* = *wert*

Verwenden Sie diesen optionalen Parameter, um dem WSDL-Dokument in WSRR angepasste Metadaten hinzuzufügen. The WSRR-CUSTOM-Paare mit *eigenschaftsname* = *wert* werden im WSDL-Dokument hinzugefügt und in WSRR ohne das Präfix WSRR-CUSTOM angezeigt.

Sie können maximal 255 angepasste "*eigenschaftsname* = *wert*"-Paare angeben. Vermeiden Sie doppelte und leere "*eigenschaftsname* = *wert*"-Paare.

Verwenden Sie diesen Parameter nur, wenn der Parameter **WSRR-SERVER** angegeben ist.

WSRR-DESCRIPTION = *wert*

Verwenden Sie diesen optionalen Parameter, um die Metadaten anzugeben, die das veröffentlichte WSDL-Dokument beschreiben.

Verwenden Sie diesen Parameter nur, wenn der Parameter **WSRR-SERVER** angegeben ist.

WSRR-ENCODING = *wert*

Verwenden Sie diesen optionalen Parameter, um die Zeichensatzcodierung des WSDL-Dokuments anzugeben. Wenn der Parameter **WSRR-ENCODING** nicht angegeben ist, verwendet WSRR den im WSDL-Dokument angegebenen Wert.

Verwenden Sie diesen Parameter nur, wenn der Parameter **WSRR-SERVER** angegeben ist.

WSRR-LOCATION = *wert*

Verwenden Sie diesen optionalen Parameter, um den URI anzugeben, der die Position des WSDL-Dokuments angibt. Wenn dieser Parameter nicht angegeben ist, nimmt der URI standardmäßig den im Parameter **WSDL** angegebenen Dateinamen an. Wenn z. B. der Wert des Parameters **WSDL** `wsrr/example.wsdl` lautet, nimmt der Wert des Parameters **WSRR-LOCATION** standardmäßig `example.wsdl` an.

Verwenden Sie diesen Parameter nur, wenn der Parameter **WSRR-SERVER** angegeben ist.

WSRR-PASSWORD = *wert*

Verwenden Sie diesen optionalen Parameter, wenn Sie ein Kennwort für den Zugriff auf WSRR eingeben müssen.

Wenn der Parameter **WSRR-USERNAME** angegeben wird, müssen Sie auch diesen Parameter angeben.

Verwenden Sie diesen Parameter nur, wenn der Parameter **WSRR-SERVER** angegeben ist.

WSRR-SERVER = { *domänenname* : *portnummer* | *ip-adresse* : *portnummer* }

Verwenden Sie diesen Parameter, um die Position des IBM WebSphere Service Registry and Repository-Servers (WSRR) anzugeben. Wenn dieser Parameter angegeben ist, wird die WSRR-Parametervalidierung verwendet.

WSRR-USERNAME = *wert*

Verwenden Sie diesen optionalen Parameter, wenn Sie einen Benutzernamen für den Zugriff auf WSRR angeben müssen. Dieser Benutzername wird von WSRR verwendet, um die Eignereigenschaft festzulegen.

Verwenden Sie diesen Parameter nur, wenn der Parameter **WSRR-SERVER** angegeben ist.

WSRR-VERSION = { 1 | *wert* }

Verwenden Sie diesen Parameter, um die Versionseigenschaft des WSDL-Dokuments in WSRR festzulegen.

Verwenden Sie diesen Parameter nur, wenn der Parameter **WSRR-SERVER** angegeben ist.

Weitere Informationen

- Die Benutzer-ID, unter der DFHLS2SC ausgeführt wird, muss für die Verwendung von UNIX System Services konfiguriert sein. Die Benutzer-ID muss über Leseberechtigungen für die z/OS UNIX-Dateistruktur und PDS-Bibliotheken unter CICS und über Schreibberechtigung für die in den Parametern **LOGFILE**, **WSBIND** und **WSDL** angegebenen Verzeichnisse verfügen.
- Der Benutzer-ID muss ausreichend Speicher zugeordnet sein, um Java auszuführen.

- Die JCL weist eine maximale Parameterlänge von 100 Zeichen auf. Diese kann mithilfe der Anweisung **STDPARM** erhöht werden. Weitere Informationen finden Sie unter *z/OS UNIX System Services User's Guide*.

Beispiel

```
//LS2WS JOB '
abrechnungsdaten
',
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/wsbind/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/wsbind/example.wsbind
WSDL=/u/exampleapp/wsd1/example.wsd1
WSDL_2.0=/u/exampleapp/wsd1/example_20.wsd1
WSDLCP=LOCAL
WSDL-NAMESPACE=http://mywsdlnamespace
/*
```

DFHWS2LS: Konvertierung von WSDL in eine höhere Programmiersprache

Die Prozedur DFHWS2LS generiert aus einer Web-Service-Beschreibung eine Datenstruktur einer höheren Programmiersprache und eine Web-Service-Bindungsdatei. Sie können DFHWS2LS verwenden, wenn Sie ein CICS-Anwendungsprogramm als Service-Provider verfügbar machen oder wenn Sie einen Service-Requester erstellen.

Jobsteueranweisungen für DFHWS2LS

JOB Startet den Job.

EXEC Gibt den Prozedurnamen (DFHWS2LS) an.

INPUT.SYSUT1 DD

Gibt die Eingabe an. Die Eingabeparameter werden in der Regel im Eingabedatenstrom angegeben. Sie können jedoch in einer Datei oder in einem Member einer partitionierten Datei definiert werden.

Symbolische Parameter

Die folgenden symbolischen Parameter sind in DFHWS2LS definiert:

JAVADIR = *pfad*

Gibt den Namen des Java-Verzeichnisses an, das von DFHWS2LS verwendet wird. Der Wert dieses Parameters wird an `/usr/lpp/` angehängt, um den vollständigen Pfadnamen `/usr/lpp/pfad` zu erstellen.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **JAVADIR** angegeben wurde.

PATHPREF = *präfix*

Gibt ein optionales Präfix an, das den z/OS UNIX-Verzeichnispfad erweitert, der in anderen Parametern verwendet wird. Der Standardwert ist eine leere Zeichenfolge.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **PATHPREF** angegeben wurde.

TMPDIR = *tmp-verzeichnis*

Gibt die Position eines Verzeichnisses in z/OS UNIX an, das DFHWS2LS als temporären Arbeitsbereich verwendet. Die Benutzer-ID, unter der der Job ausgeführt wird, muss über Lese- und Schreibberechtigungen für dieses Verzeichnis verfügen.

Der Standardwert ist /tmp.

TMPFILE = *tmp-präfix*

Gibt ein Präfix an, das DFHWS2LS verwendet, um die Namen der temporären Arbeitsbereichsdateien zu erstellen.

Der Standardwert ist WS2LS.

USSDIR = *pfad*

Gibt den Namen des CICS TS-Verzeichnisses im UNIX System Services-Dateisystem an. Der Wert dieses Parameters wird an /usr/lpp/cicsts/ angehängt, um den vollständigen Pfadnamen /usr/lpp/cicsts/*pfad* zu erstellen.

In der Regel geben Sie diesen Parameter nicht an. Der Standardwert ist der Wert, der für den CICS-Installationsjob (DFHISTAR) im Parameter **USSDIR** angegeben wurde.

SERVICE = *wert*

Verwenden Sie diesen Parameter nur, wenn sie vom IBM Support dazu aufgefordert werden.

Der temporäre Arbeitsbereich

DFHWS2LS erstellt die folgenden drei temporären Dateien zur Laufzeit:

```
tmp-verzeichnis / tmp-präfix .in  
tmp-verzeichnis / tmp-präfix .out  
tmp-verzeichnis / tmp-präfix .err
```

Dabei gilt Folgendes:

tmp-verzeichnis ist der Wert, der im Parameter **TMPDIR** angegeben ist.

tmp-präfix ist der Wert, der im Parameter **TMPFILE** angegeben ist.

Die Standardnamen für die Dateien lauten wie folgt, wenn **TMPDIR** und **TMPFILE** nicht angegeben sind:

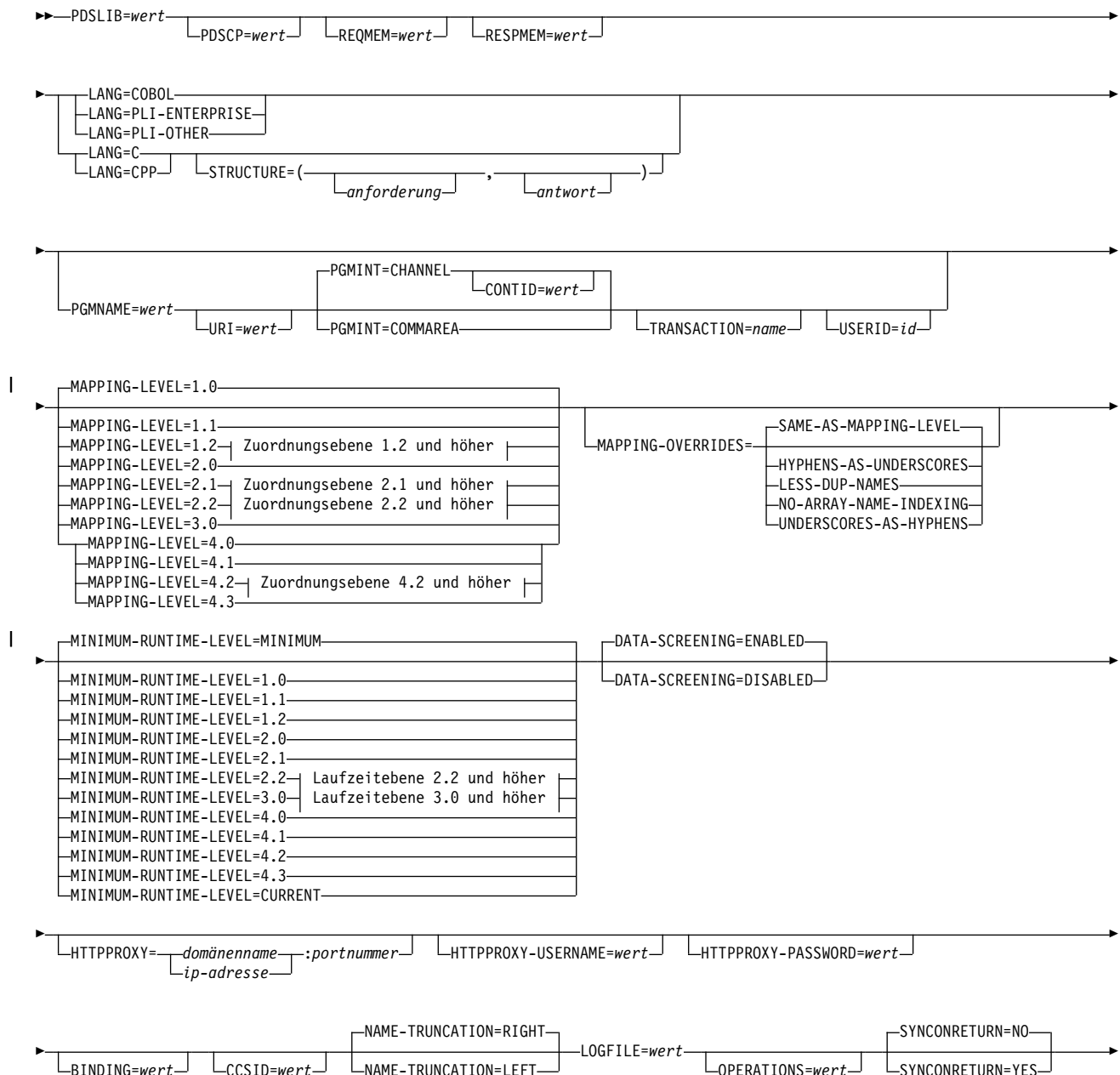
```
/tmp/WS2LS.in  
/tmp/WS2LS.out  
/tmp/WS2LS.err
```

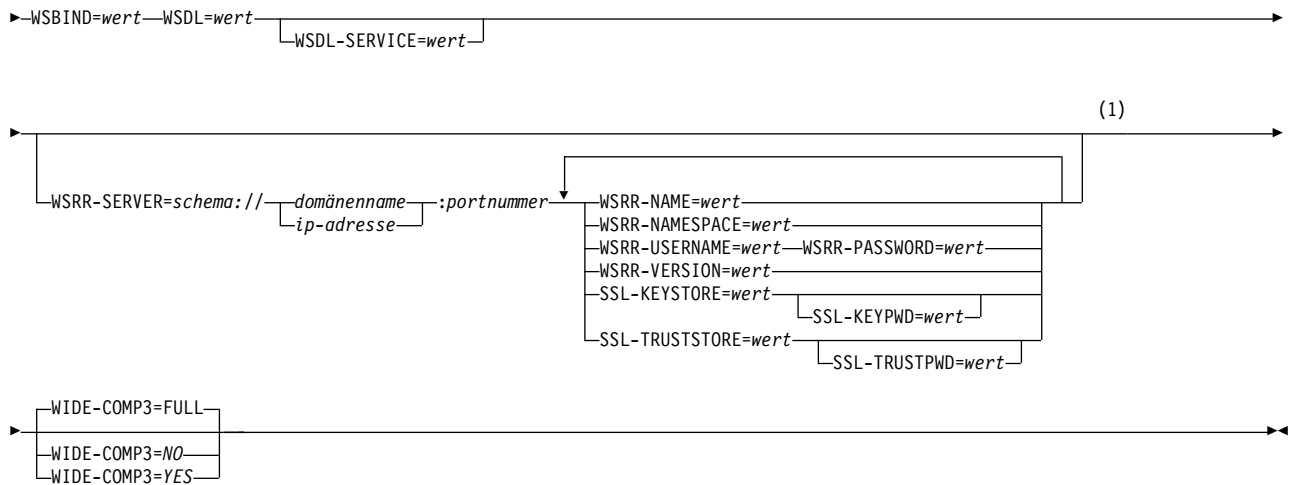
Important: DFHWS2LS sperrt nicht den Zugriff auf die z/OS UNIX-Dateien oder -Dateimember. Wenn folglich zwei oder mehr Instanzen von DFHWS2LS gleichzei-

tig ausgeführt werden und dieselben temporären Arbeitsbereichsdateien verwenden, kann nicht verhindert werden, dass ein Job die Arbeitsbereichsdateien überschreibt, während ein anderer Job sie verwendet, was zu unvorhersehbaren Fehlern führt.

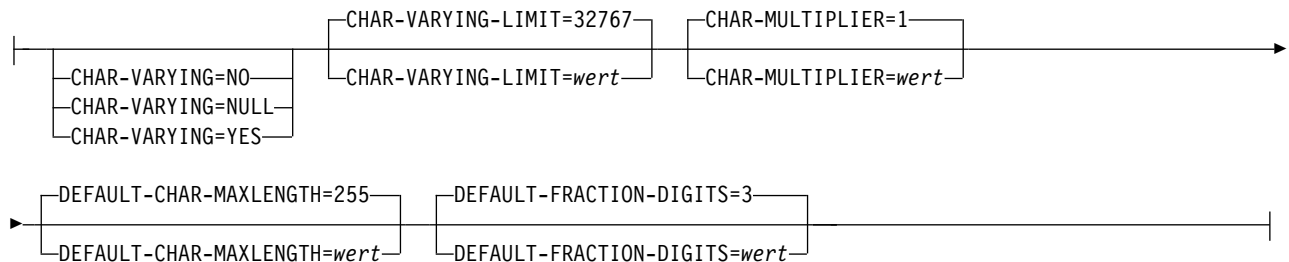
Deshalb wird empfohlen, eine Namenskonvention und Betriebsprozeduren auszuarbeiten, die diese Situation verhindern. Sie können beispielsweise den symbolischen Systemparameter **SYSUID** verwenden, um Arbeitsbereichsdateinamen zu generieren, die für einen einzelnen Benutzer eindeutig sind. Diese temporären Dateien werden gelöscht, bevor der Job beendet wird.

Eingabeparameter für DFHWS2LS





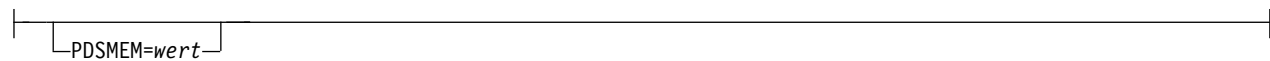
Zuordnungsebene 1.2 und höher:



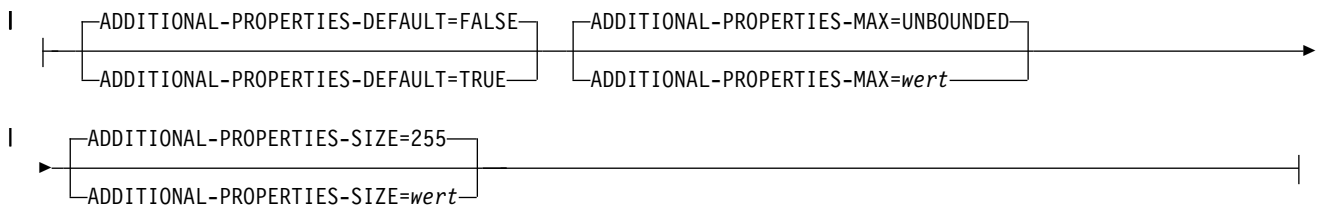
Zuordnungsebene 2.1 und höher:



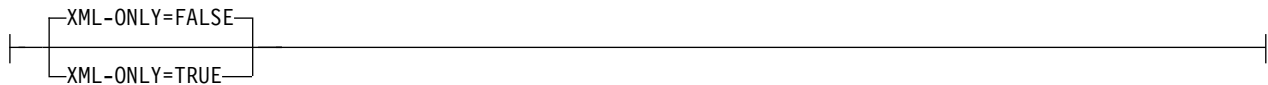
Zuordnungsebene 2.2 und höher:



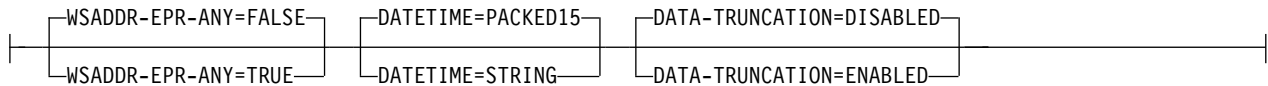
Zuordnungsebene 4.2 und höher:



Laufzeitebene 2.1 und höher:



Laufzeitebene 3.0 und höher:



Anmerkungen:

- 1 Jeder WSRR-Parameter, der angegeben werden kann, wenn der Parameter **WSRR-SERVER** festgelegt wird, kann nur einmal angegeben werden.

Parameterverwendung

- Sie können die Eingabeparameter in beliebiger Reihenfolge angeben.
- Jeder Parameter muss in einer neuen Zeile eingegeben werden.
- Ein Parameter (und ggf. das Fortsetzungszeichen) darf nicht über Spalte 72 hinausgehen; die Spalten 73-80 müssen Leerzeichen enthalten.
- Wenn ein Parameter zu lang ist, um in eine einzelne Zeile zu passen, verwenden Sie am Ende der Zeile einen Stern (*), um anzugeben, dass der Parameter in der nächsten Zeile fortgesetzt wird. Alles, einschließlich der Leerzeichen, vor dem Stern wird als Teil des Parameters betrachtet. Beispiel:

```
WSBIND=wsbinddir*  
      /app1
```

ist äquivalent zu

```
WSBIND=wsbinddir/app1
```

- Ein #-Zeichen an der ersten Zeichenposition der Zeile ist ein Kommentarzeichen. Die Zeile wird ignoriert.
- Ein Komma an der letzten Zeichenposition der Zeile ist ein optionales Zeilentrennzeichen und wird ignoriert.

Parameterbeschreibungen

ADDITIONAL-PROPERTIES-DEFAULT = { **true** | **false** }

Gibt an, ob JSON-Schemaobjekte, die zusätzliche Eigenschaften nicht explizit unterstützen, als unterstützende Elemente interpretiert werden oder nicht. Zusätzliche JSON-Eigenschaften sind alle Eigenschaften in einem JSON-Objekt, die im JSON-Schema nicht vordefiniert sind. Diese Eigenschaften werden in der Regel vom Datenumsetzungsmechanismus als unerwartete Zusatzdaten zurückgewiesen. Wenn **ADDITIONAL-PROPERTIES-DEFAULT** auf TRUE gesetzt ist, oder wenn das JSON-Schema explizit `additionalProperties:true` für ein Objekt festlegt, wird ein Bereich für solche Werte in den generierten Copybooks reserviert. Anwendungen können mit diesen Werten interagieren, indem sie die zugeordneten Felder in den Copybooks verwenden.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

ADDITIONAL-PROPERTIES-MAX = { **0-20** | **UNBOUNDED** }

Gibt an, wie viele zusätzliche Eigenschaften für ein entsprechendes JSON-Objekt unterstützt werden. Siehe **ADDITIONAL-PROPERTIES-DEFAULT**. Die generierten

Copybooks enthalten Strukturen, die sich für die Adressierung aller zusätzlichen Eigenschaften eignen. Standardmäßig gibt es keine maximale Einschränkung für die Anzahl der unterstützten Eigenschaften. Die Copybooks werden in ähnlicher Weise wie Arrays ohne Einschränkungen generiert und verwenden Container. Mit diesem Parameter kann eine maximale Einschränkung angewendet werden, die in Kombination mit dem Parameter **INLINE-MAXOCCURS-LIMIT** dafür sorgt, dass ein Array mit fester Länge für die maximale Anzahl von Eigenschaften zugeordnet werden kann, wodurch keine Container erforderlich sind.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

ADDITIONAL-PROPERTIES-SIZE = { **16-32767** | **255** }

Gibt die maximale Größe für jede der zusätzlichen JSON-Eigenschaften an. Wenn ein JSON-Objekt zusätzliche Eigenschaften unterstützt, wie in **ADDITIONAL-PROPERTIES-DEFAULT** definiert, dann haben die generierten Copybooks Bindungen, um Eigenschaften bis zu der von **ADDITIONAL-PROPERTIES-MAX** angegebenen Anzahl zu unterstützen. Standardmäßig beträgt der für jede zusätzliche Eigenschaft unterstützte Maximalwert 255 Zeichen. Ein Feld dieser Größe wird in den erstellten Copybooks generiert. Diese Größe kann durch Festlegen des Parameters **ADDITIONAL-PROPERTIES-SIZE** angepasst werden. So wird beispielsweise ein JSON-Objekt verarbeitet, das folgende Eigenschaft enthält:

```
"example": { "notes": "this extra property was not defined in the JSON Schema"
}
```

Wenn die Copybooks generiert wurden, um zusätzliche Eigenschaften zu unterstützen, wird der gesamte Wert zur Verarbeitung an die Anwendung übergeben. Der Wert beginnt mit dem führenden Anführungszeichen vor dem Eigenschaftsschlüssel und endet mit der abschließenden rechten geschweiften Klammer im Eigenschaftswert. In diesem Beispiel sind es etwa 100 Zeichen. Der Wert für **ADDITIONAL-PROPERTIES-SIZE** muss groß genug sein, um den größtmöglichen Wert enthalten zu können. Wenn der zugeordnete Puffer für den verarbeiteten Wert zu klein ist, wird eine Fehlerantwort generiert.

Sie können diesen Parameter auf Zuordnungsebene 4.2 oder höher verwenden.

BINDING = *wert*

Wenn die Web-Service-Beschreibung mehrere <wsdl:Binding>-Elemente enthält, verwenden Sie diesen Parameter, um anzugeben, welches Element zum Generieren der Sprachstruktur und der Web-Service-Bindungsdatei verwendet werden soll. Geben Sie den Wert des Attributs name an, das in dem Element <wsdl:Binding> in der Web-Service-Beschreibung verwendet wird.

CCSID = *wert*

Gibt die CCSID an, die zur Laufzeit verwendet wird, um Zeichendaten in der Anwendungsdatenstruktur zu codieren. Der Wert dieses Parameters überschreibt den Wert des Systeminitialisierungsparameters **LOCALCCSID**. *wert* muss eine EBCDIC-CCSID sein, die von Java- und z/OS-Konvertierungsservices unterstützt wird (siehe z/OS Unicode Services User's Guide and Reference). Wenn Sie diesen Parameter nicht angeben, wird die Anwendungsdatenstruktur unter Verwendung der im Systeminitialisierungsparameter angegebenen CCSID codiert.

Sie können diesen Parameter mit jeder Zuordnungsebene verwenden.

CHAR-MULTIPLIER = { **1** | *wert* }

Gibt die Anzahl von Bytes an, die für die einzelnen Zeichen auf Zuordnungsebene 1.2 oder höher zulässig sein sollen. Der Wert (*wert*) dieses Parameters

kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein. Alle nicht numerischen, zeichenbasierten Zuordnungen unterliegen diesem Multiplikator. Binäre, numerische, in Zonen eingeteilte und gepackte Dezimalfelder unterliegen diesem Multiplikator nicht.

Dieser Parameter kann nützlich sein, wenn Sie beispielsweise planen, DBCS-Zeichen zu verwenden, bei denen Sie sich für einen Multiplikator von 3 entscheiden können, um zur Laufzeit Platz zu lassen für mögliche SO- und SI-Zeichen (shift-out = Umschalttaste nicht gedrückt, shift-in = Umschalttaste gedrückt) um jedes Doppelbytezeichen herum.

Wenn Sie **CCSID=1200** (d. h. UTF-16) angeben, sind 2 oder 4 die einzig gültigen Werte für **CHAR-MULTIPLIER**. Wenn Sie UTF-16 verwenden, ist der Standardwert 2. Verwenden Sie **CHAR-MULTIPLIER=2**, wenn Sie erwarten, dass Anwendungsdaten Zeichen enthalten, die eine UTF-16-Codierungseinheit erfordern. Verwenden Sie **CHAR-MULTIPLIER=4**, wenn Sie erwarten, dass Anwendungsdaten Zeichen enthalten, die zwei UTF-16-Codierungseinheiten erfordern.

Anmerkung: Wenn Sie **CHAR-MULTIPLIER** auf 1 festlegen, schließt dies nicht aus, dass DBCS-Zeichen verwendet werden können. Und wenn Sie den Parameter auf 2 festlegen, schließt dies nicht aus, dass UTF-16-Ersatzzeichenpaare verwendet werden können. Wenn jedoch regelmäßig Breitzichen verwendet werden, werden einige gültige Werte nicht in das zugeordnete Feld passen. Wenn ein größerer Wert für **CHAR-MULTIPLIER** verwendet wird, ist es möglich, mehr Zeichen in dem zugeordneten Feld zu speichern als in der XML gültig sind. Achten Sie darauf, die passenden Bereichseinschränkungen einzuhalten.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Gibt an, wie Zeichendaten mit variabler Länge auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Binärdatentypen mit variabler Länge werden immer entweder einem Container oder einer variierenden Struktur zugeordnet. Wenn Sie diesen Parameter nicht angeben, hängt die Standardzuordnung von der angegebenen Sprache ab. Sie haben folgende Optionen:

- NO** Zeichendaten mit variabler Länge werden als Zeichenfolgen mit fester Länge zugeordnet.
- NULL** Zeichendaten mit variabler Länge werden auf null endenden Zeichenfolgen zugeordnet.
- YES** Zeichendaten mit variabler Länge werden in PL/I einem CHAR VARYING-Datentyp zugeordnet. In den COBOL-, C- und C++-Sprachen werden Zeichendaten mit variabler Länge einer äquivalenten Darstellung zugeordnet, die zwei verwandte Elemente umfasst: die Datenlänge und die Daten.

CHAR-VARYING-LIMIT = { **32767** | *wert* }

Gibt die maximale Größe von Binärdaten und Zeichendaten mit variabler Länge an, die der Sprachstruktur auf Zuordnungsebene 1.2 oder höher zugeordnet werden. Wenn die Zeichen oder Binärdaten größer sind als der in diesem Parameter angegebene Wert, werden sie einem Container zugeordnet, und der Containername wird in der generierten Sprachstruktur verwendet. Der Wert kann zwischen 0 und dem Standardwert 32.767 Bytes liegen.

CONTID = *wert*

Gibt in einem Service-Provider den Namen des Containers an, der die übergeordnete Datenstruktur enthält, die zur Darstellung einer SOAP-Nachricht verwendet wird.

Die Länge des Containers, die CICS an das Zielanwendungsprogramm übergibt, ist die größere der beiden Längen des Anforderungscontainers bzw. des Antwortcontainers.

DATA-SCREENING = { **ENABLED** | **DISABLED** }

Gibt an, ob die von Anwendungen bereitgestellten Daten auf Fehler überprüft werden.

ENABLED

Alle von Anwendungen bereitgestellten Laufzeitdaten, die nicht mit der Sprachstruktur konsistent sind, werden als Fehler behandelt, und es wird die Nachricht DFHPI1010 ausgegeben. Eine Fehlerantwort wird an die Anwendung zurückgegeben.

DISABLED

Werte in den von Anwendungen bereitgestellten Laufzeitdaten, die nicht konsistent mit der Sprachstruktur sind, werden durch Standardwerte ersetzt. Eine Null ersetzt beispielsweise einen fehlerhaften Wert in einem numerischen Feld. Die Nachricht DFHPI1010 wird nicht ausgegeben und es wird eine normale Antwort an die Anwendung zurückgegeben. Dieses Feature kann verwendet werden, um die Fehlerantworten **INVALID_PACKED_DEC** und **INVALID_ZONED_DEC** zu vermeiden, die von nicht initialisierten Ausgabefeldern generiert werden.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Gibt an, ob Daten mit variabler Länge in einer Feldstruktur mit fester Länge toleriert werden:

DISABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, weist CICS die abgeschnittenen Daten zurück und gibt eine Fehlermeldung aus.

ENABLED

Wenn die Daten kleiner sind als die feste Länge, die von CICS erwartet wird, toleriert CICS die abgeschnittenen Daten und verarbeitet die fehlenden Daten als Nullwerte.

DATETIME = { **PACKED15** | **STRING** }

Gibt an, wie `<xsd:dateTime>`-Elemente der Sprachstruktur zugeordnet werden.

PACKED15

Standardmäßig wird jedes `<xsd:dateTime>`-Element als Zeitmarke verarbeitet und dem Format CICS ABSTIME zugeordnet.

STRING

Das `<xsd:dateTime>`-Element wird als Text verarbeitet.

DEFAULT-CHAR-MAXLENGTH = { **255** | *wert* }

Gibt die Standardarraylänge von Zeichendaten in Zeichen für Zuordnungen an, in deren Web-Service-Beschreibungsdokument keine Länge eingeschlossen ist, wenn die Zuordnungsebene 1.2 oder höher ist. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 1 - 2.147.483.647 sein.

DEFAULT-FRACTION-DIGITS = { **3** | *wert* }

Gibt die Standardanzahl an Bruchteilstellen für einen XML-Dezimalschematyp an. Der Standardwert ist 3. Für COBOL liegt der gültige Bereich zwischen 0 und 17, wenn der Parameter **WIDE-COMP3** verwendet wird. Für C oder PLI liegt der gültige Bereich zwischen 0 und 30.

HTTPPROXY = { *domänenname* : *portnummer* | *ip-adresse* : *portnummer* }

Wenn Ihre WSDL Referenzen auf andere WSDL-Dateien enthält, die sich im Internet befinden, und das System, auf dem DFHWS2LS ausgeführt wird, einen Proxy-Server für den Zugriff auf das Internet verwendet, geben Sie den Domännennamen oder die IP-Adresse und die Portnummer des Proxy-Servers an. Beispiel:

HTTPPROXY=proxy.example.com:8080

In anderen Fällen ist dieser Parameter nicht erforderlich.

HTTPPROXY-PASSWORD = *wert*

Gibt das HTTP-Proxy-Kennwort an, das mit **HTTPPROXY-USERNAME** verwendet werden muss, wenn das System, auf dem DFHWS2LS ausgeführt wird, einen HTTP-Proxy-Server für den Zugriff auf das Internet verwendet und wenn der HTTP-Proxy-Server Basisauthentifizierung verwendet. Sie können diesen Parameter nur verwenden, wenn Sie auch **HTTPPROXY** angeben.

HTTPPROXY-USERNAME = *wert*

Gibt den HTTP-Proxy-Benutzernamen an, der mit **HTTPPROXY-PASSWORD** verwendet werden muss, wenn das System, auf dem DFHWS2LS ausgeführt wird, einen HTTP-Proxy-Server für den Zugriff auf das Internet verwendet und wenn der HTTP-Proxy-Server Basisauthentifizierung verwendet. Sie können diesen Parameter nur verwenden, wenn Sie auch **HTTPPROXY** angeben.

INLINE-MAXOCCURS-LIMIT = { 1 | *wert* }

Gibt basierend auf dem Attribut maxOccurs an, ob variable, wiederkehrende Inline-Inhalte verwendet werden oder nicht. Variabel wiederkehrende Inhalte, die inline zugeordnet sind, werden zusammen mit der generierten Sprachstruktur im aktuellen Container platziert. Der variabel wiederkehrende Inhalt wird in zwei Teilen gespeichert: als Zähler, der die Anzahl der Datenvorkommen speichert, und als Array, in dem alle Datenvorkommen gespeichert werden. Die alternative Zuordnung für variabel wiederkehrenden Inhalt ist eine containerbasierte Zuordnung, die die Anzahl von Datenvorkommen und den Namen des Containers speichert, in dem die Daten platziert werden. Das Speichern der Daten in einem separaten Container wirkt sich auf die Leistung aus, weshalb eine Inline-Zuordnung die bevorzugte Lösung ist.

Der Parameter **INLINE-MAXOCCURS-LIMIT** ist nur ab Zuordnungsebene 2.1 verfügbar. Der Wert von **INLINE-MAXOCCURS-LIMIT** kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein. Der Wert 0 gibt an, dass die Inline-Zuordnung nicht verwendet wird. Der Wert 1 gibt an, dass optionale Elemente inline zugeordnet werden. Wenn die Variable *wert* des Attributs maxOccurs größer als die Variable *wert* von **INLINE-MAXOCCURS-LIMIT** ist, wird eine containerbasierte Zuordnung verwendet. Andernfalls wird eine Inline-Zuordnung verwendet.

Wenn Sie entscheiden, ob variabel wiederkehrende Listen inline zugeordnet werden sollen, ziehen Sie die Länge der einzelnen Elemente von wiederkehrenden Daten in Betracht. Wenn es wenige lange Instanzen gibt, ist die containerbasierte Zuordnung die bevorzugte Lösung. Wenn es viele kurze Instanzen gibt, ist die Inline-Zuordnung geeigneter.

LANG = **COBOL**|**PLI-ENTERPRISE**|**PLI-OTHER**|**C**|**CPP**

Gibt die Datenstruktur der höheren Programmiersprache an:

COBOL

COBOL

PLI-ENTERPRISE

Enterprise PL/I

PLI-OTHER

Andere PL/I-Ebene als Enterprise PL/I

C C

CPP C++

LOGFILE = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, in die DFHWS2LS das Aktivitätenprotokoll und die Traceinformationen schreibt. DFHWS2LS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur.

In der Regel verwenden Sie diese Datei nicht, aber dies kann von der IBM Serviceorganisation angefordert werden, wenn Sie Probleme mit DFHWS2LS feststellen.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | 4.2 | 4.3 }

Gibt die Zuordnungsebene an, die DFHWS2LS bei der Generierung der Web-Service-Bindungsdatei und der Sprachstruktur verwendet. Sie haben folgende Optionen:

- 1.0** Dies ist die Standardzuordnungsebene. Die Web-Service-Bindungsdatei und die Sprachstruktur werden mithilfe von CICS TS 3.1-Zuordnungsebenen generiert.
- 1.1** XML-Attribute und <list>- sowie <union>-Datentypen werden der Sprachstruktur zugeordnet. Zeichen- und Binärdaten mit einer maximalen Länge von mehr als 32.767 Bytes werden einem Container zugeordnet. Der Containername wird in der Sprachstruktur erstellt.
- 1.2** Verwenden Sie die Parameter **CHAR-VARYING** und **CHAR-VARYING-LIMIT**, um zu steuern, wie Zeichendaten zugeordnet und zur Laufzeit verarbeitet werden. Wenn Sie keinen dieser Parameter angeben, werden Binär- und Zeichendaten mit einer maximalen Länge von weniger als 32.768 Bytes einer VARYING-Struktur für alle Sprachen außer C++ zugeordnet, wo Zeichendaten einer auf null endenden Zeichenfolge zugeordnet werden.
- 2.0** Verwenden Sie diese Zuordnungsebene in einer Region in CICS TS 3.2 oder höher, um die Erweiterungen der Zuordnung zwischen der Sprachstruktur und der Web-Service-Bindungsdatei zu nutzen.
- 2.1** Verwenden Sie diese Zuordnungsebene für eine Region in CICS TS 3.2 oder höher zur Unterstützung für <xsd:any> und xsd:anyType, der Option zur Zuordnung von variabel wiederkehrenden Inline-Inhalten mit dem Parameter **INLINE-MAXOCCURS-LIMIT** und zur Unterstützung für 'minOccurs="0"' in <xsd:sequence>, <xsd:choice> und <xsd:all>.
- 2.2** Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 3.2 oder höher, um die folgende Unterstützung zu nutzen:
 - Elemente mit festen Werten
 - Erweiterte Unterstützung für <xsd:choice>-Elemente
 - Abstrakte Datentypen
 - Abstrakte Elemente
 - Substitutionsgruppen
- 3.0** Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 4.1 oder höher. Auf dieser Zuordnungsebene können Sie Zeitmarken in das CICS ABSTIME-Format umsetzen.

- 4.0 Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher, wenn Sie UTF-16 verwenden möchten.
- 4.1 Zur Unterstützung abschneidbarer Arrays verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.2 oder höher.
- 4.2 Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.
- 4.3 Verwenden Sie diese Zuordnungsebene mit einer Region in CICS TS 5.4 oder höher.

Weitere Informationen zu Zuordnungsebenen finden Sie unter Mapping levels for the CICS assistants.

MAPPING-OVERRIDES = { **SAME-AS-MAPPING-LEVEL** | **HYPHENS-AS-UNDERScores** | **LESS-DUP-NAMES** | **NO-ARRAY-NAME-INDEXING** | **UNDERScores-AS-HYPHENS** }

Gibt an, ob das Standardverhalten für die angegebene Zuordnungsebene beim Generieren von Sprachstrukturen überschrieben wird.

Anmerkung: In einer durch Kommas begrenzten Liste können alle Unteroptionen verwendet werden. Die Optionen schließen sich nicht gegenseitig aus; sie können kombiniert und nicht geordnet verwendet werden.

SAME-AS-MAPPING-LEVEL

Dieser Parameter generiert Sprachstrukturen auf dieselbe Weise wie die Zuordnungsebene. Dies ist die Standardeinstellung.

HYPHENS-AS-UNDERScores

Nur für PL/I. Dieser Parameter konvertiert alle Bindestriche im WSDL-Dokument in Unterstriche und nicht in das Zeichen X, um die Lesbarkeit der generierten PL/I-Sprachstrukturen zu verbessern. Weitere Informationen finden Sie unter XML schema to PL/I mapping. Diese Option ist auf Zuordnungsebene 4.2 automatisch aktiviert.

LESS-DUP-NAMES

Dieser Parameter generiert nicht strukturelle Strukturfeldnamen mit `_value` am Ende des Namens, um das direkte Referenzieren des Felds zu aktivieren. Beispiel: In der folgenden PL/I-Sprachstruktur wird, wenn **MAPPING-OVERRIDES=LESS-DUP-NAMES** angegeben ist, an das Ebene-12-Feld 'streetName' das Element `_value` angehängt:

```
09 streetName,
12 streetName CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

Die resultierende Struktur sieht wie folgt aus:

```
09 streetName,
12 streetName_value CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

Diese Option ist auf Zuordnungsebene 4.2 automatisch aktiviert.

NO-ARRAY-NAME-INDEXING

Nur für COBOL und Enterprise PL/I. Stellt sicher, dass die Feldnamen in einem Array nur im Rahmen der übergeordneten Struktur eindeutig sind.

UNDERScores-AS-HYPHENS

Diese Option ist auf Zuordnungsebene 4.0 automatisch aktiviert.

Nur für COBOL. Dieser Parameter konvertiert alle Unterstriche im WSDL-Dokument in Bindestriche und nicht in das Zeichen X, um die Lesbarkeit der generierten COBOL-Sprachstrukturen zu verbessern. Im Fall von Feldnamenskollisionen werden die Felder nummeriert, um sicherzustellen, dass sie eindeutig sind. Weitere Informationen finden Sie unter XML schema to COBOL mapping.

MINIMUM-RUNTIME-LEVEL = { **MINIMUM** | **1.0** | **1.1** | **1.2** | **2.0** | **2.1** | **2.2** | **3.0** | **4.0** | **4.1** | **4.2** | **4.3** | **CURRENT** }

Gibt die älteste CICS-Laufzeitumgebung an, in der die Web-Service-Bindungsdatei implementiert werden kann. Wenn Sie eine Version auswählen, die nicht mit den anderen Parametern übereinstimmt, die Sie angegeben haben, erhalten Sie eine Fehlermeldung. Sie haben folgende Optionen:

MINIMUM

Die niedrigste mögliche Laufzeitebene von CICS wird anhand der von Ihnen ausgewählten Parameter automatisch zugewiesen.

- 1.0** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS TS 3.1-Region implementiert. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 1.1** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS TS 3.1-Region implementiert. Sie können die Zuordnungsebene 1.1 oder niedriger für den Parameter **MAPPING-LEVEL** verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 1.2** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS TS 3.1-Region implementiert. Sie können die Zuordnungsebene 1.2 oder niedriger für den Parameter **MAPPING-LEVEL** verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 2.0** Die generierte Web-Service-Bindungsdatei wird erfolgreich in CICS TS 3.2 oder einer höheren Region implementiert. Sie können die Zuordnungsebene 2.0 oder niedriger für den Parameter **MAPPING-LEVEL** verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 2.1** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 3.2 oder höher implementiert. Sie können die Zuordnungsebene 2.1 oder niedriger für den Parameter **MAPPING-LEVEL** verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 2.2** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 3.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 2.2 oder niedriger verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 3.0** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 4.1 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 3.0 oder niedriger verwenden. Manche Parameter sind auf dieser Laufzeitebene nicht verfügbar.
- 4.0** Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzei-

tebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.0 oder niedriger verwenden. Sie können alle optionalen Parameter auf dieser Ebene verwenden.

- 4.1 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.2 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.1 oder niedriger verwenden.
- 4.2 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS V5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.2 oder niedriger verwenden.
- 4.3 Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer Region unter CICS TS 5.4 oder höher implementiert. Bei dieser Laufzeitebene können Sie für den Parameter **MAPPING-LEVEL** eine Zuordnungsebene von 4.3 oder niedriger verwenden.

CURRENT

Die generierte Web-Service-Bindungsdatei wird erfolgreich in einer CICS-Region auf derselben Laufzeitebene implementiert, die Sie zum Generieren der Web-Service-Bindungsdatei verwenden.

NAME-TRUNCATION = { LEFT | RIGHT }

Gibt an, ob Namen von XML-Elementen von links oder von rechts abgeschnitten werden. Der CICS-Web-Service-Assistent schneidet Namen von XML-Elementen auf die passende Länge für die angegebene höhere Programmiersprache ab. Standardmäßig werden Namen von rechts abgeschnitten.

OPERATIONS = wert

Gibt für Web-Service-Requester-Anwendungen ein Subset von gültigen <wsdl:Operation>-Elementen aus der Web-Service-Beschreibung an, die zum Generieren der Web-Service-Bindungsdatei verwendet werden. Die einzelnen <wsdl:Operation>-Elemente sind durch Leerzeichen getrennt. Die Liste kann bei Bedarf mehrere Zeilen umfassen. Sie können diesen Parameter für WSDL 1.1- und WSDL 2.0-Dokumente verwenden.

PDSCP = wert

Gibt die Codepage an, die in den Mitgliedern einer partitionierten Datei verwendet wird, die in den Parametern **REQMEM** und **RESPMEM** angegeben sind. Dabei ist *wert* eine CCSID-Nummer oder eine Java-Codepagenummer. Wenn dieser Parameter nicht angegeben wird, wird die Codepage von z/OS UNIX System Services verwendet. Beispielsweise können Sie **PDSCP = 037** angeben.

PDSLIB = wert

Gibt den Namen der partitionierten Datei an, die die generierte höhere Programmiersprache enthält. Die Dateimember, die für die Anforderung und Antwort verwendet werden, sind in den Parametern **REQMEM** bzw. **RESPMEM** angegeben.

PDSMEM = wert

Gibt ein ein bis sechs Zeichen langes Präfix an, das DFHWS2LS zum Generieren der Namen von Mitgliedern partitionierter Dateien verwendet, die die Strukturen der höheren Programmiersprachen für abstrakte Datentypen enthalten werden. Der Membername wird durch das Anhängen einer zweistelligen Zahl an das Präfix generiert.

Verwenden Sie diesen Parameter auf Zuordnungsebene 2.2 oder höher, um die Sprachstrukturen zu benennen, die abstrakten Datentypen zugeordnet sind.

Wenn der Parameter **PDSMEM** nicht angegeben wird, werden die Sprachstrukturen für abstrakte Datentypen mithilfe des Werts im Parameter **REQMEM** benannt.

PGMINT = { **CHANNEL** | **COMMAREA** }

Gibt für einen Service-Provider an, wie CICS Daten an das Ziellanwendungsprogramm übergibt.

CHANNEL

CICS verwendet eine Kanalschnittstelle, um Daten an das Ziellanwendungsprogramm zu übergeben.

COMMAREA

CICS verwendet einen Kommunikationsbereich, um Daten an das Ziellanwendungsprogramm zu übergeben.

Dieser Parameter wird ignoriert, wenn die Ausgabe aus DFHWS2LS in einem Service-Requester verwendet wird.

Wenn das Ziellanwendungsprogramm die Anforderung verarbeitet hat, muss es denselben Mechanismus verwenden, um die Antwort zurückzugeben. Wenn die Anforderung in einem Kommunikationsbereich empfangen wurde, muss die Antwort im Kommunikationsbereich zurückgegeben werden. Wenn die Anforderung in einem Container empfangen wurde, muss die Antwort in einem Container zurückgegeben werden. Die Länge des Kommunikationsbereichs oder des Containers, die CICS an das Ziellanwendungsprogramm zurückgibt, ist größer als die Länge des Anforderungskommunikationsbereichs oder -containers und des Antwortkommunikationsbereichs oder -containers.

PGMNAME = *wert*

Gibt den Namen einer CICS-PROGRAM-Ressource an.

Wenn DFHWS2LS verwendet wird, um eine Web-Service-Bindungsdatei zu generieren, die in einem Service-Provider verwendet wird, müssen Sie diesen Parameter angeben. Er gibt den Ressourcennamen des Anwendungsprogramms an, das als Web-Service zugänglich gemacht wird.

Wenn DFHWS2LS verwendet wird, um eine Web-Service-Bindungsdatei zu generieren, die in einem Service-Requester verwendet wird, dürfen Sie diesen Parameter nicht angeben.

REQMEM = *wert*

Gibt ein ein bis sechs Zeichen langes Präfix an, das DFHWS2LS zum Generieren der Namen von Mitgliedern partitionierter Dateien verwendet, die die Strukturen der höheren Programmiersprachen für die Web-Service-Anforderung enthalten werden.

- Für einen Service-Provider ist die Web-Service-Anforderung die Eingabe für das Anwendungsprogramm.
- Für einen Service-Requester ist die Web-Service-Anforderung die Ausgabe des Anwendungsprogramms.

DFHWS2LS generiert ein Mitglied einer partitionierten Datei pro Operation. Der Membername wird durch das Anhängen einer zweistelligen Zahl an das Präfix generiert.

Obwohl dieser Parameter optional ist, müssen Sie ihn angeben, wenn die Web-Service-Beschreibung eine Definition einer Anforderung enthält.

RESPMEM = *wert*

Gibt ein ein bis sechs Zeichen langes Präfix an, das DFHWS2LS zum Generieren der Namen von Mitgliedern partitionierter Dateien verwendet, die die Strukturen der höheren Programmiersprachen für die Web-Service-Antwort enthalten werden.

- Für einen Service-Provider ist die Web-Service-Antwort die Ausgabe des Anwendungsprogramms.
- Für einen Service-Requester ist die Web-Service-Antwort die Eingabe für das Anwendungsprogramm.

DFHWS2LS generiert ein Member einer partitionierten Datei pro Operation. Der Membername wird durch das Anhängen einer zweistelligen Zahl an das Präfix generiert.

Lassen Sie diesen Parameter weg, wenn keine Antwort erwartet wird, d.h. bei unidirektionalen Nachrichten.

SSL-KEYSTORE = *wert*

Dieser optionale Parameter gibt die vollständig qualifizierte Position der Schlüsselspeicherdatei an.

Verwenden Sie diesen Parameter, wenn der Web-Service-Assistent SSL-Verschlüsselung (Secure Sockets Layer) verwenden soll, um über ein Netz mit einem IBM WebSphere Service Registry and Repository (WSRR) zu kommunizieren.

SSL-KEYPWD = *wert*

Dieser optionale Parameter gibt das Kennwort für den Schlüsselspeicher an.

Verwenden Sie diesen Parameter, wenn der Web-Service-Assistent SSL-Verschlüsselung (Secure Sockets Layer) verwenden soll, um über ein Netz mit einem IBM WebSphere Service Registry and Repository (WSRR) zu kommunizieren.

SSL-TRUSTSTORE = *wert*

Dieser optionale Parameter gibt die vollständig qualifizierte Position der Truststore-Datei an.

Verwenden Sie diesen Parameter, wenn der Web-Service-Assistent SSL-Verschlüsselung (Secure Sockets Layer) verwenden soll, um über ein Netz mit einem IBM WebSphere Service Registry and Repository (WSRR) zu kommunizieren.

SSL-TRUSTPWD = *wert*

Dieser optionale Parameter gibt das Kennwort für den Truststore an.

Verwenden Sie diesen Parameter, wenn der Web-Service-Assistent SSL-Verschlüsselung (Secure Sockets Layer) verwenden soll, um über ein Netz mit einem IBM WebSphere Service Registry and Repository (WSRR) zu kommunizieren.

STRUCTURE = (*anforderung* , *antwort*)

Nur für C und C++: Gibt an, wie die Namen der Anforderungs- und Antwortstrukturen generiert werden.

Die generierten Anforderungs- und Antwortstrukturen werden *anforderung nn* und *antwort nn* genannt, wobei *nn* ein numerisches Suffix ist, das generiert wird, um die Strukturen der einzelnen Operationen voneinander zu unterscheiden.

Wenn ein Name oder beide Namen fehlen, entsprechen die Namen der Strukturen den Namen der Member der partitionierten Dateien, die aus den angegebenen Parametern **REQMEM** und **RESPMEM** generiert werden.

SYNCONRETURN = { **NO** | **YES** }

Gibt an, ob der ferne Web-Service einen Synchronisationspunkt absetzen kann.

NO Der ferne Web-Service kann keinen Synchronisationspunkt absetzen.

Dies ist der Standardwert. Wenn der ferne Web-Service einen Synchronisationspunkt absetzt, schlägt er mit einem ADPL-Abbruch fehl.

YES Der ferne Web-Service kann einen Synchronisationspunkt absetzen. Wenn Sie YES auswählen, wird die ferne Task als separate Arbeitseinheit festgeschrieben, wenn die Steuerung vom fernen Web-Service zurückgegeben wird. Wenn der ferne Web-Service eine wiederherstellbare Ressource aktualisiert und ein Fehler auftritt, nachdem sie zurückgegeben wurde, kann die Aktualisierung dieser Ressource nicht zurückgesetzt werden.

TRANSACTION = *name*

In einem Service-Provider gibt dieser Parameter den ein bis vier Zeichen langen Namen einer Aliastransaktion an, die die Pipeline starten kann. Der Wert dieses Parameters wird verwendet, um das Attribut TRANSACTION der URIMAP-Ressource zu definieren, wenn es automatisch mithilfe des **PIPELINE**-Scanbefehls erstellt wird.

Zulässige Zeichen:

A-Z a-z 0-9 \$ @ # _ < >

URI = *wert*

In einem Service-Provider gibt dieser Parameter den relativen URI an, über den ein Client auf den Web-Service zugreift. CICS verwendet den angegebenen Wert, wenn eine URIMAP-Ressource aus der Web-Service-Bindungsdatei generiert wird, die von DFHWS2LS erstellt wird. Der Parameter gibt die Pfadkomponente des URI an, für den die URIMAP-Definition gilt.

In einem Service-Requester wird der URI des Ziel-Web-Service *nicht* mit diesem Parameter angegeben. CICS generiert keine URIMAP-Ressource für einen Service-Requester. Sie können Ihre eigene URIMAP-Ressource für Service-Requester definieren, um sie zu verwenden, wenn Clientanforderungen an den URI des Ziel-Web-Service gestellt werden. Wenn ein Service-Requester den Befehl **INVOKE SERVICE** ausgibt, verwendet CICS die Position soap:address aus dem Element wsdl:port, das ggf. in der Web-Service-Beschreibung angegeben ist. Sie können diesen Wert überschreiben und mithilfe der Optionen URIMAP oder URI im Befehl **INVOKE SERVICE** einen anderen URI angeben.

USERID = *id*

In einem Service-Provider gibt dieser Parameter eine ein bis acht Zeichen lange Benutzer-ID an, die von jedem Web-Client verwendet werden kann. Für eine anwendungsgenerierte Antwort oder einen Web-Service wird die Aliastransaktion unter dieser Benutzer-ID angehängt. Der Wert dieses Parameters wird verwendet, um das Attribut USERID der URIMAP-Ressource zu definieren, wenn es automatisch mithilfe des **PIPELINE**-Scanbefehls erstellt wird.

Zulässige Zeichen:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3 = { FULL | NO | YES }

Steuert die maximale Länge der gepackten Dezimalvariablen in der generierten COBOL- oder PL/I-Sprachstruktur.

FULL Für COBOL und PL/I generiert DFHJS2LS ein gepacktes Dezimalfeld, das groß genug ist, um alle gültigen Werte aufzunehmen. Die maximale Länge sind 31 Ziffern. Dies ist die Standardeinstellung.

- NO** Nur für COBOL. DFHJS2LS begrenzt die Länge der gepackten Dezimalvariablen auf 18 Ziffern, wenn der COBOL-Sprachstrukturtyp COMP-3 generiert wird. Wenn die gepackte Dezimalzahl länger als 18 Ziffern ist, wird die Nachricht DFHPI9022W ausgegeben, um darauf hinzuweisen, dass der angegebene Typ auf höchstens 18 Ziffern begrenzt ist.
- YES** Nur für COBOL. DFHJS2LS unterstützt die maximale Länge von 31 Ziffern, wenn der COBOL-Sprachstrukturtyp COMP-3 generiert wird.

Anmerkung: Die Optionen NO und YES generieren Felder, die nicht alle gültigen Werte darstellen können. Mit der Option FULL wird dieses Problem behoben. Allerdings lässt die Option FULL auch die Darstellung einiger ungültiger Werte in dem gepackten Dezimalfeld zu. Wenn ein Schema beispielsweise angibt, dass es maximal fünf Ziffern und maximal zwei Nachkommastellen gibt, generiert die Option FULL ein gepacktes Dezimalfeld, das sieben Ziffern zulässt, wodurch Platz für gültige Werte wie 25000 und 999,99 verfügbar ist, aber auch genug Platz für ungültige Werte wie 9999,99. Wenn Sie die Option FULL verwenden, achten Sie darauf, keine ungültigen Werte in Anwendungsdaten zu generieren.

WSADDR-EPR-ANY = { **TRUE** | **FALSE** }

Gibt an, ob CICS eine WS-Addressing-Endpunktreferenz (EPR) in ihre einzelnen Komponenten in den Sprachstrukturen umsetzt oder die EPR als <xsd:any>-Typ behandelt. Wenn die EPR als <xsd:any>-Typ behandelt wird, kann die **WSACONTEXT BUILD-API** die EPR-XML direkt verwenden.

FALSE

DFHWS2LS verhält sich typisch, die XML wird in die Struktur einer höheren Programmiersprache umgesetzt.

TRUE Wenn Sie diese Option auf TRUE festlegen, behandelt CICS die ganze EPR zur Laufzeit als <xsd:any>-Typ und platziert die EPR-XML in einem Container, der von der Anwendung referenziert werden kann. Die Anwendung kann die EPR-XML mit der **WSACONTEXT BUILD-API** verwenden, um eine EPR im Adressierungskontext zu erstellen.

Dieser Parameter ist nur auf Laufzeitebene 3.0 verfügbar.

WSBIND = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Web-Service-Bindungsdatei. DFHWS2LS erstellt ggf. die Datei, aber nicht die Verzeichnisstruktur. Die Dateierweiterung ist standardmäßig .wsbind.

WSDL = *wert*

Der vollständig qualifizierte z/OS UNIX-Name der Datei, die die Web-Service-Beschreibung enthält. Der Dateiname ist auf Zeichen beschränkt, die für eine URL gültig sind, und darf insbesondere kein #-Zeichen enthalten. Wenn Sie WSRR verwenden, um das WSDL-Dokument abzurufen, gibt dieser Parameter die Position im Dateisystem an, an der eine lokale Kopie des WSDL-Dokuments gespeichert wird.

WSDL-SERVICE = *wert*

Gibt das Element wsdl:Service an, das verwendet wird, wenn die Web-Service-Beschreibung mehr als ein Serviceelement für ein Bindungselement enthält. Wenn Sie einen Wert für den Parameter **BINDING** angeben, muss das Serviceelement, das Sie für diesen Parameter angeben, mit dem angegebenen Bindungselement konsistent sein. Sie können diesen Parameter entweder mit WSDL 1.1- oder mit WSDL 2.0-Dokumenten verwenden.

WSRR-NAME = *wert*

Gibt den Namen des WSDL-Dokuments an, das aus WSRR abgerufen werden soll. Verwenden Sie diesen Parameter nur, wenn der Parameter **WSRR-SERVER** angegeben ist.

WSRR-NAMESPACE = *wert*

Gibt den Namensbereich des WSDL-Dokuments an, das aus WSRR abgerufen werden soll. Sie können diesen Parameter optional verwenden, wenn der Parameter **WSRR-SERVER** angegeben wird, um den Namen des WSDL-Dokuments, der im Parameter **WSRR-NAME** angegeben ist, vollständig zu qualifizieren.

WSRR-PASSWORD = *wert*

Verwenden Sie diesen optionalen Parameter, wenn Sie ein Kennwort für den Zugriff auf WSRR eingeben müssen.

Wenn der Parameter **WSRR-USERNAME** angegeben wird, müssen Sie auch diesen Parameter angeben.

Verwenden Sie diesen Parameter nur, wenn der Parameter **WSRR-SERVER** angegeben ist.

WSRR-SERVER = { *domänenname* : *portnummer* | *ip-adresse* : *portnummer* }

Verwenden Sie diesen Parameter, um die Position des IBM WebSphere Service Registry and Repository-Servers (WSRR) anzugeben. Wenn dieser Parameter angegeben ist, wird die WSRR-Parametervalidierung verwendet.

WSRR-USERNAME = *wert*

Verwenden Sie diesen optionalen Parameter, wenn Sie einen Benutzernamen für den Zugriff auf WSRR angeben müssen. Dieser Benutzername wird von WSRR verwendet, um die Eignereigenschaft festzulegen.

Verwenden Sie diesen Parameter nur, wenn der Parameter **WSRR-SERVER** angegeben ist.

WSRR-VERSION = *wert*

Gibt die Version des WSDL-Dokuments an, das aus WSRR abgerufen werden soll. Sie können diesen Parameter nur verwenden, wenn der Parameter **WSRR-SERVER** angegeben ist.

XML-ONLY = { **TRUE** | **FALSE** }

Gibt an, ob CICS die XML in der SOAP-Nachricht in Anwendungsdaten umsetzt oder nicht. Verwenden Sie den Parameter **XML-ONLY**, um Web-Service-Anwendungen zu schreiben, die die XML selbst verarbeiten.

TRUE CICS führt keine Umsetzungen für die XML durch. Die Service-Requester- oder Service-Provider-Anwendung muss mit den Inhalten des Containers DFHWS-BODY direkt arbeiten, um Daten zwischen der XML und einer höheren Programmiersprache zuzuordnen.

FALSE

CICS setzt die XML in eine höhere Programmiersprache um.

Dieser Parameter ist nur auf Laufzeitebene 2.1 und höher verfügbar.

Weitere Informationen

- Die Benutzer-ID, unter der DFHLS2SC ausgeführt wird, muss für die Verwendung von UNIX System Services konfiguriert sein. Die Benutzer-ID muss über Leseberechtigungen für die z/OS UNIX-Dateistruktur und PDS-Bibliotheken unter CICS und über Schreibberechtigung für die in den Parametern **LOGFILE**, **WSBIND** und **WSDL** angegebenen Verzeichnisse verfügen.

- Der Benutzer-ID muss ausreichend Speicher zugeordnet sein, um Java auszuführen.
- Die JCL weist eine maximale Parameterlänge von 100 Zeichen auf. Dieser Wert kann mithilfe der Anweisung **STDPARM** erhöht werden. Weitere Informationen finden Sie unter z/OS UNIX System Services User's Guide.

Beispiel

```
//WS2LS JOB '
abrechnungsdaten
',
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/wsbind/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/wsbind/example.wsbind
WSDL=/u/exampleapp/wsd1/example.wsd1
/*
```

Für DFHWS2LS erforderliche Parameter zur Unterstützung von WS-Addressing:

Wenn Sie Ihre WSDL für die Web-Service-Adressierung konfigurieren, müssen Sie die Parameter **MINIMUM-RUNTIME** und **MAPPING-LEVEL** im Web-Service-Assistenten DFHWS2LS auf einen Wert 3.0 oder höher setzen. Sie können außerdem in Betracht ziehen, den Parameter **WSADDR-EPR-ANY** auf den Wert TRUE zu setzen.

Legen Sie den Parameter **MINIMUM-RUNTIME** im Web-Service-Assistenten DFHWS2LS auf 3.0 oder höher fest. Eine Laufzeitebene 3.0 oder höher stellt sicher, dass alle WSBIND-Dateien, die vom Assistenten generiert werden, die Web-Service-Adressierung vollständig unterstützen, und mit anderen Web-Service-Plattformen zusammenarbeiten können.

Legen Sie den Parameter **MAPPING-LEVEL** im Web-Service-Assistenten DFHWS2LS auf 3.0 oder höher fest.

Wenn Elemente vom Typ `wsa:EndpointReferenceType` in den Anforderungs- oder Antwortnachrichten enthalten sind, die in Ihrem WSDL-Dokument definiert sind, und wenn Sie diese Elemente als Eingabe für den API-Befehl **WSACONTEXT BUILD** zur Laufzeit verwenden möchten, legen Sie den Parameter **WSADDR-EPR-ANY** auf TRUE fest. Indem Sie den Parameter **WSADDR-EPR-ANY** auf TRUE festlegen, geben Sie an, dass CICS die EPR-Daten zur Laufzeit nicht in eine Struktur umsetzen darf. Stattdessen muss CICS die EPR-Daten als `<xsd:any>`-Element behandeln und sie in einem benannten Container speichern.

Dieses WSDL-Beispielfragment zeigt an, wie eine `<wsa:To>`-Zuordnung als ein Element vom Typ `wsa:EndpointReferenceType` weitergegeben wird:

```

<?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>

```

Wenn das Element '`<xs:element name="myEpr" type="wsa:EndpointReferenceType"/>`' **1** von DFHWS2LS verarbeitet wird, wobei der Parameter **WSADDR-EPR-ANY** auf TRUE gesetzt ist, werden die myEpr-Elementdaten in einem benannten Container als `<xsd:any>`-Element zur Laufzeit gespeichert, und ein Verweis auf den Container wird zur generierten Sprachstruktur hinzugefügt.

Im folgenden Beispiel sehen Sie die von DFHWS2LS generierte COBOL-Sprachstruktur für das Element myEpr:

```

09 myEpr.
   12 myEpr-xml-cont          PIC X(16).
   12 myEpr-xmlns-cont       PIC X(16).

```

Der Container myEpr-xml-cont speichert den Namen des Containers, der die myEpr-Daten enthält. myEpr-xmlns-cont ist ein optionaler Container, der mit allen XML-Namensbereichsdeklarationen in diesem Bereich gefüllt wird.

Zuordnungsebenen für die CICS-Assistenten

Eine Zuordnung besteht aus einer Anzahl von Regeln, die angeben, wie Informationen zwischen Sprachstrukturen und XML-Schemas konvertiert werden. Um die ausgereiftesten Zuordnungen nutzen zu können, müssen Sie für den Parameter **MAPPING-LEVEL** in den CICS-Assistenten die aktuelle Ebene festlegen.

Jede Zuordnungsebene übernimmt die Funktionalität der vorherigen Zuordnung, wobei die höchste Zuordnungsebene die beste Auswahl an Funktionen bietet. Die höchste Zuordnungsebene bietet mehr Kontrolle über die Datenkonvertierung zur Laufzeit und entfernt Einschränkungen bei der Unterstützung für bestimmte Datentypen und XML-Elemente.

Sie können den Parameter **MAPPING-LEVEL** auf eine niedrigere Ebene festlegen, wenn Sie Anwendungen erneut implementieren möchten, die bereits auf dieser Ebene implementiert wurden.

Zuordnungsebene 4.3

Zuordnungsebene 4.3 ist mit CICS TS V5.4 mit APAR PI88519 und höher kompatibel.

Zuordnungsebene 4.3 dient in erster Linie der Verwendung mit DFHJS2LS, ist aber auch in den CICS-Web-Service-Assistenten, den XML-Assistenten und den JSON-Assistenten enthalten. Diese Zuordnungsebene implementiert Unterstützung für mehrdimensionale Arrays in JSON.

Zuordnungsebene 4.2

Zuordnungsebene 4.2 ist mit CICS TS V5.4 mit angewendetem APAR PI86039 und höher kompatibel.

Zuordnungsebene 4.2 dient in erster Linie der Verwendung mit DFHJS2LS, ist aber auch in den CICS-Web-Service-Assistenten, den XML-Assistenten und den JSON-Assistenten enthalten. Diese Zuordnungsebene implementiert Unterstützung für zusätzliche Eigenschaften in JSON und führt die folgenden drei Parameter in DFHJS2LS ein: **ADDITIONAL-PROPERTIES-DEFAULT**, **ADDITIONAL-PROPERTIES-MAX** und **ADDITIONAL-PROPERTIES-SIZE**.

Zuordnungsebene 4.1

Zuordnungsebene 4.1 ist mit einer CICS TS 5.3-Region oder einer CICS TS 5.2-Region mit angewendetem APAR PI67641 und höher kompatibel.

Zuordnungsebene 4.1 wird den Web-Service-Assistenten, XML-Assistenten und JSON-Assistenten von CICS hinzugefügt. Diese Zuordnungsebene implementiert verbesserte Zuordnungen für einfache Arrays, die von unten nach oben aus vorhandenen Copybooks generiert werden. Sie fügt außerdem die Möglichkeit hinzu, in CICS nicht initialisierten abschließenden Speicher in Arrays automatisch zu erkennen und diese Datensätze aus dem generierten XML/JSON-Formular auszuschießen. Weitere Informationen finden Sie unter „Zuordnung von COBOL zu JSON-Schema“ auf Seite 461.

DFHLS2WS, DFHWS2LS, DFHLS2SC, DFHSC2LS, DFHJS2LS und DFHLS2JS unterstützen die Parameter **TRUNCATE-NULL-ARRAYS** und **TRUNCATE-NULL-ARRAY-VALUES**.

Wenn Sie einen beliebigen Wert für **TRUNCATE-NULL-ARRAY-VALUES** angeben, müssen Sie auch **TRUNCATE-NULL-ARRAYS=ENABLED** angeben.

Zuordnungsebene 4.0

Zuordnungsebene 4.0 ist mit einer CICS TS 5.2-Region kompatibel.

Auf Zuordnungsebene 4.0 und höher unterstützen DFHLS2SC und DFHLS2WS die COBOL OCCURS DEPENDING ON-Klausel und die Zuordnung von COBOL-Zeichenarrays in XML-Zeichenfolgen. Sie können dieses Verhalten mithilfe des Parameters **CHAR-OCCURS** in den CICS-Assistenten festlegen.

- Sie müssen den Parameter **DATA-TRUNCATION=ENABLED** angeben.
- Eine komplexe Klausel OCCURS DEPENDING ON wird nicht unterstützt. Diese Einschränkung bedeutet, dass OCCURS DEPENDING ON nur für das letzte Feld einer Struktur verwendet wird.

- CICS unterstützt keine qualifizierten Namen (unter Verwendung des Schlüsselworts 'OF') als Ziel einer OCCURS DEPENDING ON-Klausel, z. B. FIELD1 OF STRUCTURE1.
- CICS unterstützt nicht das Schlüsselwort UNBOUNDED. Sie müssen die maximale Größe der Tabelle angeben, die von der Anwendung erwartet wird.

Auf Zuordnungsebene 4.0 und höher unterstützen CICS-Web-Services die Konvertierung von Anwendungsdaten, die mit UTF-16 Unicode codiert sind.

- Wenn Sie LS2WS oder LS2SC verwenden, können Sie dieses Verhalten aktivieren, indem Sie sprachspezifische Datentypen für UTF-16 verwenden.
- Wenn Sie WS2LS oder SC2LS verwenden, können Sie dieses Verhalten aktivieren, indem Sie CCSID=1200 festlegen.
- CICS unterstützt nur eine einzelne Unicode-Codepage, „ UTF-16BE mit IBM Private Use Area “ (CCSID 1200).
- Die Konvertierung von Anwendungsdaten, die mit UTF-8 codiert sind, wird nicht unterstützt.

Anmerkung: DFHLS2WS und DFHLS2SC bieten keine Unterstützung für die COBOL-Klausel GROUP USAGE NATIONAL.

Zuordnungsebene 3.0 und höher

Zuordnungsebene 3.0 ist mit einer CICS TS 4.1-Region und höher kompatibel.

Diese Zuordnungsebene stellt die folgende Unterstützung bereit:

- DFHSC2LS und DFHWS2LS ordnen xsd:dateTime-Datentypen dem CICS ASK-TIME-Format zu.
- DFHLS2WS kann ein WSDL-Dokument und eine Web-Service-Bindung aus einer Anwendung erstellen, die nicht nur einen, sondern viele Container verwendet.
- Das Tolerieren von abgeschnittenen Daten, die durch eine Datenstruktur fester Länge beschrieben werden. Sie können dieses Verhalten festlegen, indem Sie den Parameter **DATA-TRUNCATION** in den CICS-Assistenten verwenden.

Zuordnungsebene 2.2 und höher

Zuordnungsebene 2.2 ist mit einer CICS TS 3.2-Region mit angewendetem APAR PK69738 und höher kompatibel.

Auf Zuordnungsebene 2.2 und höher unterstützen DFHSC2LS und DFHWS2LS die folgenden XML-Zuordnungen:

- Feste Werte für Elemente
- Substitutionsgruppen
- Abstrakte Datentypen
- <sequence>-Elemente des XML-Schemas können in <choice>-Elementen verschachtelt sein.

DFHSC2LS und DFHWS2LS stellen erweiterte Unterstützung für die folgenden XML-Zuordnungen bereit:

- Abstrakte Elemente
- <choice>-Elemente des XML-Schemas

Zuordnungsebene 2.1 und höher

Zuordnungsebene 2.1 ist mit einer CICS TS 3.2-Region mit angewendetem APAR PK59794 und höher kompatibel.

Diese Zuordnungsebene umfasst eine bessere Kontrolle über die Verarbeitung von Variableninhalten mit dem neuen Parameter **INLINE-MAXOCCURS-LIMIT** und den neuen Werten des Parameters **CHAR-VARYING**.

Auf Zuordnungsebene 2.1 und höher bieten DFHSC2LS und DFHWS2LS die folgende neue und verbesserte Unterstützung für XML-Zuordnungen:

- <any>-Element des XML-Schemas
- xsd:anyType-Typ
- Tolerierung von abstrakten Elementen
- Parameter **INLINE-MAXOCCURS-LIMIT**
- Attribut minOccurs

Der Parameter **INLINE-MAXOCCURS-LIMIT** gibt an, ob variabel wiederkehrende Listen inline zugeordnet werden. Weitere Informationen zur Inline-Zuordnung von variabel wiederkehrenden Inhalten finden Sie unter Variablenarrays von Elementen.

Unterstützung für das Attribut minOccurs wird für die Elemente <sequence>, <choice> und <all> des XML-Schemas erweitert. Wenn minOccurs="0" , verarbeitet der CICS-Assistent diese Elemente so, als wäre das Attribut minOccurs="0" auch ein Attribut aller zugehörigen untergeordneten Elemente.

Auf Zuordnungsebene 2.1 und höher unterstützen DFHLS2SC und DFHLS2WS die folgenden XML-Zuordnungen:

- FILLER-Felder in COBOL und PL/I werden ignoriert.
- Der Wert COLLAPSE für den Parameter **CHAR-VARYING**
- Der Wert BINARY für den Parameter **CHAR-VARYING**

FILLER-Felder in COBOL und PL/I werden ignoriert. Sie werden nicht im generierten XML-Schema angezeigt und eine entsprechende Lücke wird zur Laufzeit in den Datenstrukturen beibehalten.

COLLAPSE sorgt dafür, dass CICS nachgestellte Leerzeichen in Textfeldern ignoriert.

BINARY bietet Unterstützung für Binärfelder. Dieser Wert ist nützlich, wenn Sie COBOL in ein XML-Schema konvertieren. Diese Option ist nur in SBCS-Zeichenarrays verfügbar und ermöglicht das Zuordnen des Arrays zu xsd:base64Binary-Feldern fester Länge statt zu xsd:string-Feldern.

Zuordnungsebene 1.2 und höher

Zuordnungsebene 1.2 ist mit einer CICS TS 3.1-Region und höher kompatibel.

Mehr Kontrolle über die Umsetzung von Zeichen- und Binärdaten zur Laufzeit ist mit diesen zusätzlichen Parametern in den Stapelverarbeitungstools verfügbar:

- **CHAR-VARYING**
- **CHAR-VARYING-LIMIT**
- **CHAR-MULTIPLIER**
- **DEFAULT-CHAR-MAXLENGTH**

Wenn Sie entscheiden, den Parameter **CHAR-MULTIPLIER** in DFHSC2LS oder DFHWS2LS zu verwenden, gelten die folgenden Regeln, nachdem der Wert dieses Parameters zur Berechnung des Platzes verwendet wurde, der für Zeichendaten erforderlich ist.

- DFHSC2LS und DFHWS2LS stellen die folgenden Zuordnungen bereit:
 - Zeichendatentypen variabler Länge, die eine maximale Länge von mehr als 32.767 Bytes haben, werden einem Container zugeordnet. Mit dem Parameter **CHAR-VARYING-LIMIT** können Sie ein niedrigeres Limit festlegen. In der Sprachstruktur wird ein 16-Byte-Feld erstellt, in dem der Name des Containers gespeichert wird. Zur Laufzeit werden die Zeichendaten in einem Container gespeichert, und der Containername wird in die Sprachstruktur gestellt.
 - Zeichendatentypen variabler Länge, die eine maximale Länge von weniger als 32.768 Bytes haben, werden einer VARYING-Struktur für alle Sprachen außer C/C++ und Enterprise PL/I zugeordnet. In C/C++ werden diese Datentypen auf null endenden Zeichenfolgen zugeordnet und in Enterprise PL/I werden diese Datentypen VARYINGZ-Strukturen zugeordnet. Sie können den Parameter **CHAR-VARYING** verwenden, um die Art der Zuordnung von Zeichendaten variabler Länge auszuwählen.
 - Binärdaten variabler Länge, die eine maximale Länge von weniger als 32.768 Bytes haben, werden einer VARYING-Struktur für alle Sprachen zugeordnet. Wenn die maximale Länge größer-gleich 32.768 Bytes ist, werden die Daten einem Container zugeordnet. In der Sprachstruktur wird ein 16-Byte-Feld erstellt, in dem der Name des Containers gespeichert wird. Zur Laufzeit werden die Binärdaten in einem Container gespeichert, und der Containername wird in die Sprachstruktur gestellt.

Wenn in Ihrem XML-Schema Zeichendatentypen enthalten sind, denen keine Länge zugeordnet wurde, können Sie mithilfe des Parameters **DEFAULT-CHAR-MAXLENGTH** in DFHWS2LS oder DFHSC2LS eine Standardlänge zuweisen.

DFHLS2SC und DFHLS2WS stellen die folgenden Zuordnungen bereit:

- Zeichenfelder werden einem xsd:string-Datentyp zugeordnet und können als Felder fester Länge oder als auf null endende Zeichenfolgen zur Laufzeit verarbeitet werden. Mit dem Parameter **CHAR-VARYING** können Sie die Art der Verarbeitung von Zeichendaten variabler Länge zur Laufzeit für alle Sprachen außer PL/I auswählen.
- Base64Binary-Datentypen werden auch einem Container zugeordnet, wenn die maximale Länge der Daten größer ist als 32.767 Bytes oder wenn die Länge nicht definiert ist. Wenn die Länge der Daten 32.767 Bytes oder weniger ist, wird der base64Binary-Datentyp einer VARYING-Struktur für alle Sprachen zugeordnet.

Zuordnungsebene 1.1 und höher

Zuordnungsebene 1.1 ist mit einer CICS TS 3.1-Region und höher kompatibel.

Diese Zuordnungsebene bietet eine verbesserte Zuordnung von XML-Zeichen- und Binärdatentypen, insbesondere bei der Zuordnung von Daten variabler Länge, deren Attribute `maxLength` und `minLength` mit verschiedenen Werten im XML-Schema definiert sind. Daten werden auf die folgenden Arten verarbeitet:

- Zeichen- und Binärdatentypen mit einer festen Länge größer als 16 MB werden einem Container für alle Sprachen außer PL/I zugeordnet. In PL/I werden Zeichen- und Binärdatentypen mit einer festen Länge größer als 32.767 Bytes einem Container zugeordnet. In der Sprachstruktur wird ein 16-Byte-Feld erstellt, in

dem der Name des Containers gespeichert wird. Zur Laufzeit werden die Daten fester Länge in einem Container gespeichert, und der Containername wird in die Sprachstruktur gestellt.

Da Container eine variable Länge haben können, werden Daten fester Länge, die einem Container zugeordnet werden, nicht um Leerzeichen oder Nullen ergänzt bzw. abgeschnitten, um der festen Länge zu entsprechen, die im XML-Schema oder in der Web-Service-Beschreibung angegeben ist. Wenn die Länge der Daten signifikant ist, können Sie entweder Ihre Anwendung so schreiben, dass sie diese prüft oder Sie können die Validierung in der CICS-Region aktivieren. Sowohl die SOAP- als auch die XML-Validierung wirken sich erheblich auf die Leistung aus.

- Die Datentypen `<list>` und `<union>` des XML-Schemas werden Zeichenfeldern zugeordnet.
- Schemadefinierte XML-Attribute werden zugeordnet statt ignoriert. Maximal 255 Attribute sind pro XML-Element zulässig. Weitere Informationen finden Sie unter Unterstützung für XML-Attribute.
- Das Attribut `xsi:nil` wird unterstützt. Weitere Informationen finden Sie unter Unterstützung für XML-Attribute.

Nur Zuordnungsebene 1.1

Zuordnungsebene 1.1 ist mit einer CICS TS 3.1-Region und höher kompatibel.

Diese Zuordnungsebene bietet eine verbesserte Zuordnung von XML-Zeichen- und Binärdatentypen, insbesondere bei der Zuordnung von Daten variabler Länge, deren Attribute `maxLength` und `minLength` mit verschiedenen Werten im XML-Schema definiert sind. Daten werden auf die folgenden Arten verarbeitet:

- Binärdatentypen variabler Länge werden einem Container zugeordnet. In der Sprachstruktur wird ein 16-Byte-Feld erstellt, in dem der Name des Containers gespeichert wird. Zur Laufzeit werden die Binärdaten in einem Container gespeichert, und der Containername wird in die Sprachstruktur gestellt.
- Zeichendatentypen variabler Länge, die eine maximale Länge von mehr als 32.767 Bytes haben, werden einem Container zugeordnet. In der Sprachstruktur wird ein 16-Byte-Feld erstellt, in dem der Name des Containers gespeichert wird. Zur Laufzeit werden die Zeichendaten in einem Container gespeichert, und der Containername wird in die Sprachstruktur gestellt.
- Zeichen- und Binärdatentypen mit einer festen Länge kleiner als 16 MB werden Feldern fester Länge für alle Sprachen außer PL/I zugeordnet. In PL/I werden Zeichen- und Binärdatentypen mit einer festen Länge von 32.767 Bytes oder weniger Feldern fester Länge zugeordnet.
- CICS codiert und decodiert Daten im `hexBinary`-Format, aber nicht im `base64Binary`-Format. `Base64Binary`-Datentypen im XML-Schema werden einem Feld in der Sprachstruktur zugeordnet. Die Größe des Felds wird mithilfe dieser Formel berechnet: $4 \times (\text{ceil}(z / 3))$. Dabei gilt Folgendes:
 - z ist die Länge des Datentyps im XML-Schema.
 - $\text{ceil}(x)$ ist die kleinste ganze Zahl größer als oder gleich x .

Wenn die Länge von z größer als 24.566 Bytes ist, wird die resultierende Sprachstruktur nicht kompiliert. Wenn Sie `base64Binary`-Daten größer als 24.566 Bytes haben, müssen Sie die Zuordnungsebene 1.2 verwenden. Auf Zuordnungsebene 1.2 können Sie die `base64Binary`-Daten einem Container zuordnen, statt ein Feld in der Sprachstruktur zu nutzen.

Nur Zuordnungsebene 1.0

Zuordnungsebene 1.0 ist mit einer CICS TS 3.1-Region und höher kompatibel.

Beachten Sie die folgenden Einschränkungen, die auf neueren Zuordnungsebenen geändert werden:

- DFHSC2LS und DFHWS2LS ordnen Zeichen- und Binärdatentypen im XML-Schema Feldern fester Länge in der Sprachstruktur zu. Sehen Sie sich dieses partielle XML-Schema an:

```
<xsd:element name="example">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="33000"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Dieses partielle XML-Schema wird in einer COBOL-Sprachstruktur ähnlich diesem Beispiel angezeigt:

```
15 example PIC X(33000)
```

- CICS codiert und decodiert Daten im hexBinary-Format, aber nicht im base64Binary-Format. DFHSC2LS und DFHWS2LS ordnen Base64Binary-Daten einem Zeichenfeld fester Länge zu, dessen Inhalte von dem Anwendungsprogramm codiert bzw. decodiert werden müssen.
- DFHSC2LS und DFHWS2LS ignorieren XML-Attribute während der Verarbeitung.
- DFHLS2SC und DFHLS2WS interpretieren Zeichen- und Binärfelder in der Sprachstruktur als Felder fester Länge und ordnen diese Felder XML-Elementen zu, die ein Attribut maxLength aufweisen. Zur Laufzeit werden die Felder in der Sprachstruktur mit Leerzeichen oder Nullen gefüllt, wenn nicht ausreichend Daten verfügbar sind.

Zuordnung einer höheren Programmiersprache zu einem XML-Schema:

Mit den CICS-Assistenten können Sie Zuordnungen zwischen Strukturen einer höheren Programmiersprache und XML-Schemas oder WSDL-Dokumenten generieren. Die CICS-Assistenten generieren auch XML-Schemas oder WSDL-Dokumente aus Datenstrukturen einer höheren Programmiersprache oder umgekehrt.

Die Dienstprogramme DFHSC2LS und DFHLS2SC werden gesammelt als CICS-XML-Assistent bezeichnet. Die Dienstprogramme DFHWS2LS und DFHLS2WS werden gesammelt als CICS-Web-Service-Assistent bezeichnet.

- DFHLS2SC und DFHLS2WS ordnen Strukturen höherer Programmiersprachen XML-Schemas und WSDL-Dokumenten zu.
- DFHSC2LS und DFHWS2LS ordnen XML-Schemas und WSDL-Dokumente Strukturen höherer Programmiersprachen zu.

Die beiden Zuordnungen sind nicht symmetrisch:

- Wenn Sie eine Sprachdatenstruktur mit DFHLS2SC oder DFHLS2WS verarbeiten und dann das resultierende XML-Schema bzw. WSDL-Dokument mit dem ergänzenden Dienstprogramm (DFHSC2LS bzw. DFHWS2LS) verarbeiten, stimmt die finale Datenstruktur wahrscheinlich nicht mit dem Original überein. Die finale Datenstruktur ist aber logisch äquivalent zum Original.
- Wenn Sie ein XML-Schema oder WSDL-Dokument mit DFHSC2LS oder DFHWS2LS verarbeiten und dann die resultierende Sprachstruktur mit dem er-

gänzenden Dienstprogramm (DFHLS2SC bzw. DFHLS2WS) verarbeiten, stimmt das finale XML-Schema oder WSDL-Dokument wahrscheinlich nicht mit dem Original überein.

- In manchen Fällen generieren DFHSC2LS und DFHWS2LS Sprachstrukturen, die nicht von DFHLS2SC und DFHLS2WS unterstützt werden.

Sie müssen Sprachstrukturen, die von DFHLS2SC und DFHLS2WS verarbeitet werden, entsprechend den Regeln der Sprache codieren, die in den von CICS unterstützten Sprachcompilern implementiert sind.

Zuordnung von COBOL zu JSON-Schema:

Das Dienstprogramm DFHLS2JS unterstützt Zuordnungen zwischen COBOL-Datenstrukturen und JSON-Schemadefinitionen.

COBOL-Namen werden entsprechend den folgenden Regeln in JSON-Namen konvertiert:

- Doppelte Namen werden durch Hinzufügen einer oder mehrerer Ziffern eindeutig gemacht.
Zwei Instanzen von year werden zu year und year1.
- Bindestriche werden durch Unterstriche ersetzt. Aufeinanderfolgende Bindestriche werden durch aufeinanderfolgende Unterstriche ersetzt.
Beispiel: current-user--id wird zu current_user__id.
- Segmente von Namen, die durch Bindestriche begrenzt sind und nur Großbuchstaben enthalten, werden in Kleinbuchstaben konvertiert.
Beispiel: CA-REQUEST-ID wird zu ca_request_id.
- Ein Unterstrich wird vor Namen hinzugefügt, die mit einer Ziffer beginnen.
Beispiel: 9A-REQUEST-ID wird zu _9a_request_id.

CICS ordnet COBOL-Datenbeschreibungselemente entsprechend der folgenden Tabelle Schemaelementen zu. COBOL-Datenbeschreibungselemente, die in der Tabelle nicht enthalten sind, werden von DFHLS2JS nicht unterstützt. Es gelten außerdem die folgenden Einschränkungen:

- Datenbeschreibungselemente mit den Ebenennummern 66 und 77 werden nicht unterstützt. Datenbeschreibungselemente mit der Ebenennummer 88 werden ignoriert.
- Die folgenden Klauseln in Datenbeschreibungseinträgen werden nicht unterstützt:
 - REDEFINES
 - RENAMES; d. h. Ebene 66
 - DATE FORMAT
- Die folgenden Klauseln in Datenbeschreibungselementen werden ignoriert:
 - BLANK WHEN ZERO
 - JUSTIFIED
 - VALUE
- Die SIGN-Klausel SIGN TRAILING wird unterstützt. Die SIGN-Klausel SIGN LEADING wird nur unterstützt, wenn in DFHLS2JS eine Zuordnungsebene 1.2 oder höher angegeben ist.
- SEPARATE CHARACTER wird sowohl für SIGN TRAILING- als auch für SIGN LEADING-Klauseln auf Zuordnungsebene 1.2 oder höher unterstützt.
- Die folgenden Phrasen in der USAGE-Klausel werden nicht unterstützt:

OBJECT REFERENCE
 POINTER
 FUNCTION-POINTER
 PROCEDURE-POINTER

- Die folgenden Phrasen in der USAGE-Klausel werden auf Zuordnungsebene 1.2 oder höher unterstützt:
 COMPUTATIONAL-1
 COMPUTATIONAL-2
- Die einzigen PICTURE-Zeichen, die für DISPLAY- und COMPUTATIONAL-5-Datenbeschreibungselemente unterstützt werden, sind 9, S und Z.
- Die PICTURE-Zeichen, die für PACKED-DECIMAL-Datenbeschreibungselemente unterstützt werden, sind 9, S, V und Z.
- Die einzigen PICTURE-Zeichen, die für bearbeitete numerische Datenbeschreibungselemente unterstützt werden, sind 9 und Z.
- Wenn der Parameter MAPPING-LEVEL auf 1.2 oder höher und der Parameter CHAR-VARYING auf NULL festgelegt ist, werden Zeichenarrays einem string-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet.
- Wenn der Parameter MAPPING-LEVEL auf 1.2 oder höher und der Parameter CHAR-VARYING auf BINARY festgelegt ist, werden Zeichenarrays einem string-Element zugeordnet und als binäre Daten verarbeitet.
- Wenn der Parameter MAPPING-LEVEL auf 1.2 oder höher und der Parameter CHAR-VARYING auf COLLAPSE festgelegt ist, werden Leerzeichen am Ende von Zeichenfolgen ignoriert.
- Die Klausel OCCURS DEPENDING ON wird auf Zuordnungsebene 4.0 oder höher unterstützt. Eine komplexe Klausel OCCURS DEPENDING ON wird nicht unterstützt. Dies bedeutet, dass OCCURS DEPENDING ON nur für das letzte Feld einer Struktur unterstützt wird.
- Die Klausel OCCURS INDEXED BY wird auf jeder Zuordnungsebene unterstützt.

COBOL-Datenbeschreibung	JSON-Schemadefinition
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>)	

COBOL-Datenbeschreibung	JSON-Schemadefinition
<pre> PIC S9 DISPLAY PIC S99 DISPLAY PIC S999 DISPLAY PIC S9999 DISPLAY PIC S9(<i>n</i>) DISPLAY PIC S9(<i>n</i>) COMP PIC S9(<i>n</i>) COMP-4 PIC S9(<i>n</i>) COMP-5 PIC S9(<i>n</i>) BINARY </pre>	<pre> "type":"integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i> </pre> <p>Dabei ist <i>n</i> der Höchstwert, der durch ein Muster von '9'-Zeichen dargestellt werden kann.</p>
<pre> PIC 9 DISPLAY PIC 99 DISPLAY PIC 999 DISPLAY PIC 9999 DISPLAY PIC 9(<i>n</i>) DISPLAY PIC 9(<i>n</i>) COMP PIC 9(<i>n</i>) COMP-4 PIC 9(<i>n</i>) COMP-5 PIC 9(<i>n</i>) BINARY </pre>	<pre> "type":"integer", "minimum":0, "maximum": <i>n</i> </pre> <p>Dabei ist <i>n</i> der Höchstwert, der durch ein Muster von '9'-Zeichen dargestellt werden kann.</p>

COBOL-Datenbeschreibung	JSON-Schemadefinition
<pre> PIC S9(m) V9(n) COMP-3 </pre>	<pre> "type": "number", "format": "decimal", "minimum": x , "maximum": y , "multipleOf": z </pre> <p>Dabei gilt Folgendes:</p> <p>x ist der Mindestwert, der durch das Muster von '9'-Zeichen dargestellt werden kann.</p> <p>y ist der Höchstwert, der durch das Muster von '9'-Zeichen dargestellt werden kann.</p> <p>z ist die kleinste verfügbare Einheit = $1/10^n$.</p>
<pre> PIC 9(m) V9(n) COMP-3 </pre>	<pre> "type": "number", "format": "decimal", "minimum": 0, "maximum": y , "multipleOf": z </pre> <p>Dabei gilt Folgendes:</p> <p>y ist der Höchstwert, der durch das Muster von '9'-Zeichen dargestellt werden kann.</p> <p>z ist die kleinste verfügbare Einheit = $1/10^n$.</p>
<pre> PIC S9(15) COMP-3 </pre> <p>Unterstützt auf Zuordnungsebene 3.0, wenn DATETIME=PACKED15</p>	<pre> "type": "string", "format": "date-time" </pre> <p>Das Format der Zeitmarke wird durch RFC3339 definiert.</p>
<pre> PIC S9(m) V9(n) DISPLAY </pre> <p>Unterstützt auf Zuordnungsebene 1.2 und höher</p>	<pre> "type": "number", "format": "decimal", "minimum": x , "maximum": y , "multipleOf": z </pre> <p>Dabei gilt Folgendes:</p> <p>x ist der Mindestwert, der durch das Muster von '9'-Zeichen dargestellt werden kann.</p> <p>y ist der Höchstwert, der durch das Muster von '9'-Zeichen dargestellt werden kann.</p> <p>z ist die kleinste verfügbare Einheit = $1/10^n$.</p>

COBOL-Datenbeschreibung	JSON-Schemadefinition
<p>COMP-1</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert.</p> <p>Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-1-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre>"type": "number", "format": "float"</pre>
<p>COMP-2</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert.</p> <p>Einige extrem große oder kleine Werte sind für doppelte Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-2-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden. .</p>	<pre>"type": "number", "format": "double"</pre>

COBOL-Datenbeschreibung	JSON-Schemadefinition
<p><i>datenbeschreibung</i> OCCURS <i>n</i> TIMES</p>	<p>Auf Zuordnungsebene 4.0 und höher</p> <p>Für Basiselemente:</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { "type": "object", "properties": { <i>name</i> : { <i>datenbeschreibung_JSON</i> } } } "required": [<i>name</i>] </pre> <p>Für Strukturen:</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { <i>datenbeschreibung_JSON</i> } </pre> <p>Dabei ist <i>datenbeschreibung_JSON</i> die JSON-Schemadarstellung der <i>datenbeschreibung</i> in COBOL und <i>name</i> der Name der <i>datenbeschreibung</i> in COBOL.</p> <p>Auf Zuordnungsebene 4.1 und höher</p> <p>TRUNCATE-NULL-ARRAYS = DISABLED</p> <p>Sowohl strukturierte als auch Basisarrays werden wie folgt zugeordnet.</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { <i>datenbeschreibung_JSON</i> } </pre> <p>TRUNCATE-NULL-ARRAYS = ENABLED</p> <p>Basisarrays werden wie oben gezeigt zugeordnet, und strukturierte Arrays wie folgt.</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": 0 "items": { <i>datenbeschreibung_JSON</i> } </pre>

COBOL-Datenbeschreibung	JSON-Schemadefinition
<i>datenbeschreibung</i> OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> Unterstützt auf Zuordnungsebene 4.0	<i>"field-name"</i> : { "type": "array" , "maxItems": <i>m</i> "minItems": <i>n</i> "items":{ ... } } Der Inhalt des Arrayelements hängt vom verwendeten Datentyp ab.
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	Wenn CHAR-OCCURS =STRING: <i>"field-name"</i> : { "type": "string" , "maxLength": <i>n</i> } Dies ist eine Zeichenfolge.

COBOL-Datenbeschreibung	JSON-Schemadefinition
	<p>Wenn CHAR-OCCURS =ARRAY:</p> <p>Auf Zuordnungsebene 4.0 und höher</p> <pre> "field-name" :{ "maxItems": m , "minItems": n , "items":{ "type": "object" , "properties":{ "field-name":{ "type": "string" , "maxLength":1 } }, "required":["field-name"] } } </pre> <p>Dies ist ein Array einzelner Zeichen.</p> <p>Auf Zuordnungsebene 4.1 und höher</p> <p>TRUNCATE-NULL-ARRAYS = <i>DISABLED</i></p> <p>Sowohl strukturierte als auch Basisarrays werden wie folgt zugeordnet.</p> <pre> "type":"array" "maxItems": n "minItems": n "items":{ datenbeschreibung_JSON } </pre> <p>TRUNCATE-NULL-ARRAYS = <i>ENABLED</i></p> <p>Basisarrays werden wie oben gezeigt zugeordnet, und strukturierte Arrays wie folgt.</p> <pre> "type":"array" "maxItems": n "minItems": 0 "items":{ datenbeschreibung_JSON } </pre>

COBOL-Datenbeschreibung	JSON-Schemadefinition
<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>Wenn CHAR-OCCURS =STRING:</p> <pre> "field-name" : { "type": "string" , "maxLength": m "minLength": n } </pre>
<pre> PIC N(n) USAGE NATIONAL </pre> <p>Wenn CHAR-USAGE =NATIONAL: PIC N(n)</p>	<p>Auf Zuordnungsebene 4.0 und höher:</p> <pre> "type": "string", "maxLength": n </pre> <p>Zur Laufzeit füllt CICS das Anwendungsdatenstrukturfeld mit UTF-16-Daten.</p>

Zuordnung von JSON-Schema zu COBOL:

Das Dienstprogramm DFHJS2LS unterstützt Zuordnungen zwischen JSON-Schemas und COBOL-Datenstrukturen.

Die CICS-Assistenten generieren eindeutige, gültige Feldnamen für COBOL-Variablen aus den Schemaelementnamen mithilfe der folgenden Regeln:

1. Reservierten COBOL-Wörtern wird das Präfix X vorangestellt.
Beispiel: DISPLAY wird zu XDISPLAY.
2. Alle Zeichen außer A-Z, a-z, 0-9 oder Bindestrich werden durch X ersetzt.
Beispiel: monthly_total wird zu monthlyXtotal.
3. Wenn das letzte Zeichen ein Bindestrich ist, wird es durch X ersetzt.
Beispiel: ca-request- wird zu ca-requestX.

4. Doppelte Namen in demselben Bereich werden durch das Hinzufügen von ein oder zwei Ziffern zu der zweiten Instanz und zu allen nachfolgenden Instanzen des Namens eindeutig gemacht.

Beispiel: Drei Instanzen von year werden zur year , year1 und year2.

Sollte das oben beschriebene Verhalten unerwünscht sein, kann der Benutzer **MAPPING-OVERRIDES=NO-ARRAY-NAME-INDEXING** als Eingabe für das Dienstprogramm angeben, die das Hinzufügen einer oder mehrerer Ziffern zur zweiten Instanz und allen nachfolgenden Instanzen des Namens inaktiviert.

5. Ein JSON-Schema gibt an, dass eine Variable eine variierende Kardinalität hat, wenn das Element "type" den Wert "array" hat und wenn die Schlüsselwörter "minItems" und "maxItems" nicht angegeben sind oder abweichende Werte aufweisen. Wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat, werden Feldnamen mit den Suffixes "_cont" und "_num" erstellt.

Weitere Informationen finden Sie unter „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.

6. Ein JSON-Schema gibt an, dass eine Variable optional ist, wenn sie nicht in dem "required"-Schlüsselwortarray enthalten ist, das dem einschließenden JSON-Schema vom Typ "object" zugeordnet ist. Bei optionalen Feldern wird ein zusätzliches Feld mit dem Suffix _num zu dem Elementnamen hinzugefügt. Zur Laufzeit ist dies gleich null, um anzugeben, dass der Wert in den JSON-Daten gefehlt hat, und ungleich null, wenn der Wert in den JSON-Daten vorhanden war.
7. Feldnamen sind auf 28 Zeichen begrenzt. Wenn ein generierter Name, einschließlich des Präfix und Suffix, diese Länge überschreitet, wird der Elementname abgeschnitten.

DFHJS2LS ordnet Schematypen zu COBOL-Datenbeschreibungselementen mithilfe der angegebenen Zuordnungsebene in der folgenden Tabelle zu. Beachten Sie die folgenden Punkte:

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL gesetzt ist, werden Zeichendaten mit variabler Länge auf null endenden Zeichenfolgen zugeordnet, und ein zusätzliches Zeichen wird dem Nullabschlusszeichen zugeordnet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf YES gesetzt ist, werden Zeichendaten mit variabler Länge zwei verwandten Elementen zugeordnet: einem Längenfeld und einem Datenfeld.

Beispiel:

```
"textString": {  
  "type": "string",  
  "maxLength": 10000,  
  "minLength": 1  
}
```

Zuordnung zu:

```
15 textString-length PIC S9999 COMP-5 SYNC  
15 textString PIC X(10000)
```

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
Alle: "type": "null" "type": []	Nicht unterstützt

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
"\$schema": "http://json-schema.org/draft-04/schema#"	Dieses Schlüsselwort wird ignoriert, es wird jedoch davon ausgegangen, dass es mit der JSON-Schemaspezifikation 'draft-04' kompatibel ist.
"title": "same text" "description": "more text"	Diese Schlüsselwörter werden ignoriert.
"format": "<vordefinierte werte>"	Das Schlüsselwort "format" wird verwendet, um entweder die generierte Struktur oder den Laufzeitwert zu ändern. Lesen Sie die folgenden Informationen zur unterstützten Verwendung von "format".
"type": "array", "items": {<JSON-unterschema>}, "additionalItems": false, "maxItems": m , "minItems": n	Mehrdimensionale Arrays werden ab Zuordnungsebene 4.3 aufwärts unterstützt, und Arrays mit gemischtem Typ werden nicht unterstützt. "additionalItems" wird als falsch angenommen und es wird kein anderer Wert unterstützt. Wenn sowohl "minItems" als auch "maxItems" vorhanden und gleichwertig sind, wird das Array als feste Kardinalität behandelt. Ist dies nicht der Fall, wird es als variierende Kardinalität behandelt. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.
"type": "array", "uniqueItems": true	"uniqueItems" wird nicht mit JSON-Arrays unterstützt.
"type": "object", "additionalProperties": false, "properties": { ["<elementname>": {<JSON-unterschema> } [,]]* } "required": [["<elementname>" [,]]*]	Das einzige JSON-Objekt, das derzeit unterstützt wird, ist eine feste Gruppe benannter Elemente. Dieses generiert eine Struktur (oder Unterstruktur), die diese Elementnamen verwendet. Für "additionalProperties" wird false angenommen, wenn kein Wert im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist. Alle Elemente im Objekt "properties" werden als "optional" angenommen, wenn sie sich nicht im Array "required" befinden oder wenn kein Array "required" vorhanden ist. Ein optionales Element ("optional") erhält eine variable Ordinalität von null bis X. Dabei ist X entweder 1 oder die maximale Anzahl von Elementen im Array, vorausgesetzt, das Element ist als Array definiert. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
<pre>"type": "object", "additionalProperties": true</pre>	<p>Das Objekt wird wie im vorherigen Beispiel zugeordnet, mit zusätzlichen Feldern, die zusätzliche Eigenschaften unterstützen. Für die Eigenschaft <code>additionalProperties</code> wird 'false' angenommen, wenn sie nicht im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist.</p> <p>Ist diese Option aktiviert, werden bis zu dem im Parameter ADDITIONAL-PROPERTIES-MAX angegebenen Wert Leerzeichen zugeordnet. Die Anzahl der Zeichen in jedem Leerzeichen wird durch den Parameter ADDITIONAL-PROPERTIES-SIZE festgelegt. Jede Eigenschaft wird einem Feld PIC X(z) zugeordnet, wobei z durch den Parameter ADDITIONAL-PROPERTIES-SIZE definiert wird. Ist der Wert von ADDITIONAL-PROPERTIES-MAX größer als 0, werden die Eigenschaften in ähnlicher Weise zugeordnet wie in einem Array, in dem maxItems festgelegt ist.</p> <p>Anmerkung: Es gibt mehrere Möglichkeiten, JSON-Unterstützung in CICS zu konfigurieren, einschließlich der Verwendung von z/OS Connect for CICS. Wenn Sie die ältere CICS-Java-Pipeline-Technologie nutzen, werden Additional Properties nur unterstützt, falls die <code>com.ibm.cics.json.enableAxis2Handlers-JVM</code>-Systemeigenschaft nicht auf 'true' festgelegt ist.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>Keines dieser Schlüsselwörter wird mit JSON-Objekten unterstützt.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression", "minLength": l</pre>	<p>PIC X(z)</p> <p>Dabei basiert der Wert von z auf m , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p>m basiert auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>"pattern"- und "minLength"-Einschränkungen werden über die Sprachstruktur nur in Form eines Kommentars übergeben.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>PIC N(z) USAGE NATIONAL</p> <p>Dabei basiert der Wert von z auf m , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p>m basiert auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*": { "type": "string", "format": "date-time" }</pre>	<p>PIC S9(15) COMP-3</p> <p>Alle unterstützt, wenn DATETIME=PACKED15</p> <p>Beachten Sie, dass "maxLength" und "minLength" nicht für dieses Format unterstützt werden.</p>

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
<pre>"*name*":{ "type":"string", "format":"uri" }</pre>	<p>PIC X(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>PIC N(<i>m</i>) USAGE NATIONAL</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*":{ "type":"string", "format":"base64Binary" }</pre>	<p>PIC X(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"hexBinary" }</pre>	<p>PIC X(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"<vordefiniert>" }</pre>	<p>PIC X(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i> . Ein relevantes Muster ("pattern") wird verwendet und an den Kommentar übergeben.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>PIC N(<i>m</i>) USAGE NATIONAL</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i> . Ein relevantes Muster ("pattern") wird verwendet und an den Kommentar übergeben.</p>
<pre>"type":"boolean"</pre>	<p>PIC X DISPLAY</p> <p>Der Wert x'00' impliziert 'false', x'01' impliziert 'true'.</p>
<pre>"type": "integer", "exclusiveMaximum": true, "exclusiveMinimum": true, "multipleOf": n</pre>	<p>"exclusiveMaximum"- und "exclusiveMinimum"-Einschränkungen werden nur in Form eines Kommentars an die Sprachstruktur übergeben.</p> <p>"multipleOf" wird ignoriert.</p>

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
"type"="integer", minimum=0, maximum=255	Zuordnungsebene 3.0 und niedriger: PIC X DISPLAY Zuordnungsebene 4.0 und höher (oder wenn der Parameter INTEGER-AS-PIC9 angegeben wurde, unabhängig von der Zuordnungsebene): PIC 9(z) COMP-5 SYNC oder PIC 9(z) DISPLAY Dabei ist $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:-128, maximum:127	Zuordnungsebene 3.0 und niedriger: PIC X DISPLAY Zuordnungsebene 4.0 und höher (oder wenn der Parameter INTEGER-AS-PIC9 angegeben wurde, unabhängig von der Zuordnungsebene): PIC S9(z) COMP-5 SYNC oder PIC S9(z) DISPLAY Dabei ist $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:0, maximum; m	PIC 9(z) COMP-5 SYNC oder PIC 9(z) DISPLAY Dabei ist $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:- m , maximum: m -1	PIC S9(z) COMP-5 SYNC oder PIC S9(z) DISPLAY Dabei ist $10^{(z-1)} < m \leq 10^z$
"type": "number", "maximum": m, "minimum": n, "exclusiveMaximum": true, "exclusiveMinimum": true, "multipleOf": n	"maximum"-, "minimum"-, "exclusiveMaximum"- und "exclusiveMinimum"-Einschränkungen werden nur in Form eines Kommentars an die Sprachstruktur übergeben. "multipleOf" wird ignoriert.
"type": "number" "format": "decimal"	PIC 9(p)V9(n) COMP-3 Dabei sind p und n Standardwerte.

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
<pre>"type": "number" "format": "float"</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <ul style="list-style-type: none"> • PIC X(32) <p>Zuordnungsebene 1.2 und höher:</p> <ul style="list-style-type: none"> • COMP-1 <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert.</p> <p>Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-1-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>
<pre>"type": "number" "format": "double"</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <ul style="list-style-type: none"> • PIC X(32) <p>Zuordnungsebene 1.2 und höher:</p> <ul style="list-style-type: none"> • COMP-2 <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert.</p> <p>Einige extrem große oder kleine Werte sind für doppelte Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt COMP-2-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden. .</p>

JSON-Schemaschlüsselwort	COBOL-Datenbeschreibung
<pre> oneOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] anyOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] allOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] </pre>	<p>Die logischen Pfade durch das Array von Optionen werden so zusammengeführt, als ob ein einzelnes Objekt vom Typ { "properties": { "A":..., "B":... } } definiert wäre. Jede dieser JSON-Eigenschaften wird als optional behandelt. Wenn die Unteroptionen widersprüchliche Definitionen für denselben Eigenschaftsnamen enthalten, wird eine Fehlermeldung ausgegeben. Wenn eine einzelne Eigenschaft beispielsweise sowohl als Zeichenfolge (in Pfad A) als auch als Objekt (in Pfad B) definiert ist, wird diese Definition nicht unterstützt und führt zu einer Fehlermeldung.</p> <p>Es ist möglich, dass ein JSON-Schema komplexe logische Strukturen definiert. Die in den komplexen logischen Strukturen implizierten Feinheiten können jedoch bei der Zuordnung zu einer Sprachstruktur verloren gehen. Der Umsetzungsprozess versucht nicht, die kombinatorischen Regeln aus dem Schema umzusetzen. Es interagiert nur mit den Sprachstrukturfeldern, die angeben, ob eine bestimmte JSON-Eigenschaft vorhanden oder nicht vorhanden ist.</p> <p>Wenn die Unteroptionen kompatible Definitionen für denselben Eigenschaftsnamen enthalten, versucht DFHJS2LS, das zugeordnete Muster von Einschränkungen zusammenzuführen, wobei allerdings einige Feinheiten verloren gehen können. Nehmen wir beispielsweise die folgende Definition an:</p> <pre> "A": { "oneOf": [{ "type": "string", "maxLength": 5 }, { "type": "string", "minLength": 7, "maxLength": 8 }] } </pre> <p>„A“ ist entweder als eine Zeichenfolge mit einer Länge von bis zu 5 Zeichen oder als eine Zeichenfolge mit einer Länge zwischen 7 und 8 Zeichen definiert. Die Zusammenführung kann zu Zuordnungen zu einer Zeichenfolge mit einer Länge von 0 bis 8 Zeichen führen, ohne dass erkannt wird, dass die Länge von 6 Zeichen ungültig ist.</p> <p>Die meisten Szenarios mit logischer Zusammensetzung werden einfachen Sprachstrukturen zugeordnet, aber bei einer komplizierten logischen Zusammensetzung werden während des Zuordnungsprozesses möglicherweise Kompromisse geschlossen. Die besten Ergebnisse erhalten Sie, wenn Sie die logische Zusammensetzung in einem JSON-Schema zum Definieren alternativer Deklarationen für dieselbe JSON-Eigenschaft vermeiden.</p>

Anmerkung: CICS kann keine ganzzahligen Werte größer als den maximalen Wert für 'signed long' ($2^{63} - 1$) umsetzen, es sei denn, sie sind in Anführungszeichen eingeschlossen.

Anmerkung: Minimale und maximale Werte, die in dem Schema für numerische Typen angegeben sind, werden nur zum Zuordnen zu einem COBOL-Datentyp verwendet. Die Daten werden während der Laufzeit nicht gegen diese Werte geprüft.

Einige der Schematypen, die in der Tabelle aufgeführt sind, werden dem COBOL-Format COMP-5 SYNC oder DISPLAY zugeordnet, abhängig von den Werten, die ggf. in den Schlüsselwörtern minimum und maximum angegeben sind:

- Für Typen mit Vorzeichen (short , int und long) wird DISPLAY verwendet, wenn Folgendes angegeben ist:

```
"maximum":  
a  
"minimum":  
-a
```

Dabei ist *a* eine aus Neunen bestehende Zeichenfolge.

- Für Typen ohne Vorzeichen (unsignedShort , unsignedInt und unsignedLong) wird DISPLAY verwendet, wenn Folgendes angegeben ist:

```
"maximum":  
a  
"minimum":0
```

Dabei ist *a* eine aus Neunen bestehende Zeichenfolge.

Ist ein anderer Wert oder kein Wert angegeben, wird COMP-5 SYNC verwendet.

Zuordnung von C und C++ zu JSON-Schema:

Das Dienstprogramm DFHLS2JS unterstützt Zuordnungen zwischen C- und C++-Datentypen mit JSON-Schemadefinitionen.

C- und C++-Namen werden entsprechend den folgenden Regeln in JSON-Namen konvertiert:

1. Zeichen, die in JSON-Eigenschaftsnamen nicht gültig sind, werden durch X ersetzt.

Beispiel: monthly-total wird zu monthlyXtotal.

2. Doppelte Namen werden durch Hinzufügen einer oder mehrerer Ziffern eindeutig gemacht.

Zwei Instanzen von year werden zu year und year1.

DFHLS2JS ordnet C- und C++-Datentypen Schemaelementen entsprechend der folgenden Tabelle zu. C- und C++-Typen, die in der Tabelle nicht enthalten sind, werden von DFHLS2JS nicht unterstützt. Das Qualifikationsmerkmal `_Packed` wird für Strukturen unterstützt. Folgende Einschränkungen gelten:

- Headerdateien müssen eine struct-Instanz der höchsten Ebene enthalten.
- Sie können keinen Strukturtyp deklarieren, der sich selbst als Mitglied enthält.
- Die folgenden C- und C++-Datentypen werden nicht unterstützt:

```
decimal  
long double  
wchar_t (nur C++)
```

- Folgendes wird ignoriert, wenn es in der Headerdatei vorhanden ist.

Speicherklassenkennungen:

```
auto  
register  
static  
extern  
mutable
```

Qualifikationsmerkmale

```
const
```

volatile
 _Export (nur C++)
Funktionskennungen
 inline (nur C++)
 virtual (nur C++)

Anfangswerte

- Die Headerdatei darf die folgenden Werte nicht enthalten:
 - Unionen
 - Klassendeklarationen
 - Aufzählungsdatentypen
 - Zeigertypvariablen
 - Schablonendeklarationen
 - Vordefinierte Makros, d. h. Makros mit Namen, die mit zwei Unterstrichen beginnen und enden (__)
 - Die Zeilenfortsetzungssequenz (ein '\' -Symbol, auf das unmittelbar ein Zeilenvorschubzeichen folgt)
 - Deklaratoren für Prototypfunktionen
 - Vorprozessoranweisungen
 - Bitfelder
 - Das Schlüsselwort __cdecl (oder _cdecl) (nur C++)
- Der Anwendungsprogrammierer muss einen 32-Bit-Compiler verwenden, um sicherzustellen, dass einem int-Wert 4 Bytes zugeordnet werden.
- Die folgenden C++-reservierten Schlüsselwörter werden nicht unterstützt:
 - explicit
 - using
 - namespace
 - typename
 - typeid
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL festgelegt ist, werden Zeichenarrays einem string-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf BINARY festgelegt ist, werden Zeichenarrays einem xsd:base64Binary-Element zugeordnet und als binäre Daten verarbeitet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf COLLAPSE festgelegt ist, wird <xsd:whiteSpace value="collapse"/> für Zeichenfolgen generiert.

C- und C++-Datentyp	Schema 'simpleType'
char[z]	"type": "string" "maxLength": z
char16_t[n]	Auf Zuordnungsebene 4.0 und höher: "type": "string" "maxLength": n Zur Laufzeit füllt CICS das Anwendungsdatenstrukturfeld mit UTF-16-Daten.

C- und C++-Datentyp	Schema 'simpleType'
char[8] Unterstützt auf Zuordnungsebene 3.0 und höher, wenn DATETIME=PACKED15	"type":"string" "format":"date-time" Das Format der Zeitmarke wird durch RFC3339 definiert.
char short int long long long	"type":"integer", "minimum":- (n + 1), "maximum": n Dabei ist n der Maximalwert, der durch das Basiselement dargestellt werden kann.
unsigned char unsigned short unsigned int unsigned long unsigned long long	"type":"integer", "minimum":0, "maximum": n Dabei ist n der Maximalwert, der durch das Basiselement dargestellt werden kann.
bool (nur C++)	"type":"boolean"
float Unterstützt auf Zuordnungsebene 1.2 und höher.	"type":"number", "format":"float" Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt Gleitkommatentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.
double Unterstützt auf Zuordnungsebene 1.2 und höher.	"type":"number", "format":"double" Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für doppelte Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt doppelter Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.

C- und C++-Datentyp	Schema 'simpleType'
<i>typename</i> [<i>n</i>]	<p>Für Basiselemente:</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { "type": "object", "properties": { <i>name</i> } }</pre> <p>Für Structs:</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { <i>typ_JSON</i> }</pre> <p>Dabei ist <i>typ_JSON</i> die JSON-Schemadarstellung des C- oder C++-Typs.</p>

Zuordnung von JSON-Schema zu C und C++:

Die DFHJS2LS-Dienstprogramme unterstützen Zuordnungen zwischen den JSON-Schemas und C- und C++-Datentypen.

Die CICS-Assistenten generieren eindeutige, gültige Feldnamen für C- und C++-Variablen aus den Schemaelementnamen mithilfe der folgenden Regeln:

1. Alle Zeichen außer A-Z, a-z, 0-9 oder _ werden durch X ersetzt.
Beispiel: `monthly-total` wird zu `monthlyXtotal`.
2. Wenn das erste Zeichen kein alphabetisches Zeichen ist, wird es durch ein führendes X ersetzt.
Beispiel: `_monthlysummary` wird zu `Xmonthlysummary`.
3. Doppelte Namen in demselben Bereich werden durch das Hinzufügen von ein oder zwei Ziffern zu der zweiten Instanz und zu allen nachfolgenden Instanzen des Namens eindeutig gemacht.
Beispiel: Drei Instanzen von `year` werden zu `year`, `year1` und `year2`.
4. Ein JSON-Schema gibt an, dass eine Variable eine variierende Kardinalität hat, wenn das Element "type" den Wert "array" hat und wenn die Schlüsselwörter "minItems" und "maxItems" nicht angegeben sind oder abweichende Werte aufweisen. Wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat, werden Feldnamen mit den Suffixes "_cont" und "_num" erstellt.
Weitere Informationen finden Sie unter „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.

5. Ein JSON-Schema gibt an, dass eine Variable optional ist, wenn sie nicht in dem "required"-Schlüsselwortarray enthalten ist, das dem einschließenden JSON-Schema vom Typ "object" zugeordnet ist. Bei optionalen Feldern wird ein zusätzliches Feld mit dem Suffix _num zu dem Elementnamen hinzugefügt. Zur Laufzeit ist dies gleich null, um anzugeben, dass der Wert in den JSON-Daten gefehlt hat, und ungleich null, wenn der Wert in den JSON-Daten vorhanden war.
6. Feldnamen sind auf 50 Zeichen begrenzt. Wenn ein generierter Name, einschließlich aller Präfixe und Suffixe, diese Länge überschreitet, wird der Elementname abgeschnitten.

DFHJS2LS ordnet JSON-Schematypwerte C- und C++-Datentypen gemäß der folgenden Tabelle zu. Es gelten außerdem die folgenden Regeln:

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL gesetzt ist, werden Zeichendaten mit variabler Länge auf null endenden Zeichenfolgen zugeordnet, und ein zusätzliches Zeichen wird dem Nullabschlusszeichen zugeordnet.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf YES gesetzt ist, werden Zeichendaten mit variabler Länge zwei verwandten Elementen zugeordnet: einem Längensfeld und einem Datenfeld.

JSON-Schemaschlüsselwort	C- und C++-Datentyp
Alle: "type": "null" "type": []	Nicht unterstützt
"\$schema": "http://json-schema.org/draft-04/schema#"	Dieses Schlüsselwort wird ignoriert, es wird jedoch davon ausgegangen, dass es mit der JSON-Schemaspezifikation 'draft-04' kompatibel ist.
"title": "same text" "description": "more text"	Diese Schlüsselwörter werden ignoriert.
"format": "<vordefinierte werte>"	Das Schlüsselwort "format" wird verwendet, um entweder die generierte Struktur oder den Laufzeitwert zu ändern. Lesen Sie die folgenden Informationen zur unterstützten Verwendung von "format".
<pre> "type": "array", "items": {<JSON-unterschema>}, "additionalItems": false, "maxItems": m , "minItems": n </pre>	<p>Mehrdimensionale Arrays werden ab Zuordnungsebene 4.3 aufwärts unterstützt, und Arrays mit gemischtem Typ werden nicht unterstützt.</p> <p>"additionalItems" wird als 'false' angenommen und es wird kein anderer Wert unterstützt.</p> <p>Wenn sowohl "minItems" als auch "maxItems" vorhanden und gleichwertig sind, wird das Array als feste Kardinalität behandelt. Ist dies nicht der Fall, wird es als variierende Kardinalität behandelt. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.</p>

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre>"type": "array", "uniqueItems": true</pre>	<p>"uniqueItems" wird mit JSON-Arrays nicht unterstützt. Das <JSON-unterschema> muss einen unterstützten Typ ("type") definieren, aber bei diesem Typ kann es sich nicht um ein Array ("array") handeln. Dies ist eine Einschränkung für die generierte Sprachstruktur.</p>
<pre>"type": "object", "additionalProperties": false, "properties": { ["<elementname>": {<JSON-unterschema> } [,]]* } "required": [["<elementname>" [,]]*]</pre>	<p>Das einzige JSON-Objekt, das derzeit unterstützt wird, ist eine feste Gruppe benannter Elemente.</p> <p>Dieses generiert eine Struktur (oder Unterstruktur), die diese Elementnamen verwendet.</p> <p>Für "additionalProperties" wird false angenommen, wenn kein Wert im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist.</p> <p>Alle Elemente im Objekt "properties" werden als "optional" angenommen, wenn sie sich nicht im Array "required" befinden oder wenn kein Array "required" vorhanden ist. Ein optionales Element ("optional") erhält eine variable Ordinalität von null bis X. Dabei ist X entweder 1 oder die maximale Anzahl von Elementen im Array, vorausgesetzt, das Element ist als Array definiert. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.</p>

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre>"type": "object", "additionalProperties": true</pre>	<p>Das Objekt wird wie im vorherigen Beispiel zugeordnet, mit zusätzlichen Feldern, die zusätzliche Eigenschaften unterstützen. Für die Eigenschaft <code>additionalProperties</code> wird 'false' angenommen, wenn sie nicht im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist.</p> <p>Ist diese Option aktiviert, werden bis zu dem im Parameter ADDITIONAL-PROPERTIES-MAX angegebenen Wert Leerzeichen zugeordnet. Die Anzahl der Zeichen in jedem Leerzeichen wird durch den Parameter ADDITIONAL-PROPERTIES-SIZE festgelegt. Jede Eigenschaft wird einem Feld <code>PIC X(z)</code> zugeordnet, wobei <code>z</code> durch den Parameter ADDITIONAL-PROPERTIES-SIZE definiert wird. Ist der Wert von ADDITIONAL-PROPERTIES-MAX größer als 0, werden die Eigenschaften in ähnlicher Weise zugeordnet wie in einem Array, in dem maxItems festgelegt ist.</p> <p>Anmerkung: Es gibt mehrere Möglichkeiten, JSON-Unterstützung in CICS zu konfigurieren, einschließlich der Verwendung von <code>z/OS Connect for CICS</code>. Wenn Sie die ältere <code>CICS-Java-Pipeline-Technologie</code> nutzen, werden <code>Additional Properties</code> nur unterstützt, falls die <code>com.ibm.cics.json.enableAxis2Handlers-JVM-Systemeigenschaft</code> nicht auf 'true' festgelegt ist.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>Keines dieser Schlüsselwörter wird mit JSON-Objekten unterstützt.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression>", "minLength": l</pre>	<p><code>char[z]</code></p> <p>Dabei basiert der Wert von <code>z</code> auf <code>m</code>, ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p><code>m</code> basiert auf dem Schlüsselwort <code>"maxLength"</code> und wird als Zeichenfolge fester Länge behandelt.</p> <p><code>"pattern"</code>- und <code>"minLength"</code>-Einschränkungen werden über die Sprachstruktur nur in Form eines Kommentars übergeben.</p>

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre>"type": "string" "maxLength": m</pre>	<p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>char16_t[z]</p> <p>Dabei basiert der Wert von z auf m , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p>m basiert auf dem Schlüsselwort "maxLength " und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*": { "type": "string", "format": "date-time" }</pre>	<p>char[8]</p> <p>Alle unterstützt, wenn DATETIME=PACKED15</p>
<pre>"*name*": { "type": "string", "format": "uri" }</pre>	<p>char[m]</p> <p>Dabei basiert m auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>char16_t[m]</p> <p>Dabei basiert m auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*": { "type": "string", "format": "base64Binary" }</pre>	<p>char[m]</p> <p>Dabei basiert m auf dem Schlüsselwort "maxLength".</p>
<pre>"*name*": { "type": "string", "format": "hexBinary" }</pre>	<p>char[m]</p> <p>Dabei basiert m auf dem Schlüsselwort "maxLength".</p>

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre>"*name*":{ "type":"string", "format":"<vordefiniert>" }</pre>	<p>char[<i>m</i>]</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i>. Ein relevantes Muster ("pattern") wird verwendet und an den Kommentar übergeben.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>char16_t[<i>m</i>]</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i> . Ein relevantes Muster ("pattern") wird verwendet und an den Kommentar übergeben.</p>
"type":"boolean"	<p>bool (nur C++) short (nur C)</p>
<pre>"type": "integer", "exclusiveMaximum": true, "exclusiveMinimum": true, "multipleOf": n</pre>	<p>"exclusiveMaximum"- und "exclusiveMinimum"-Einschränkungen werden nur in Form eines Kommentars an die Sprachstruktur übergeben.</p> <p>"multipleOf" wird ignoriert.</p>
<pre>"type":"integer", minimum:-128, maximum:127</pre>	signed char
<pre>"type":"integer", minimum:0, maximum:255</pre>	unsigned char
<pre>"type":"integer", minimum:-32768, maximum:32767</pre>	short
<pre>"type":"integer", minimum:0, maximum:65535</pre>	unsigned short
<pre>"type":"integer", minimum:-2147483648, maximum:2147483647</pre>	int
<pre>"type":"integer", minimum:0, maximum:4294967295</pre>	unsigned int
<pre>"type":"integer", minimum:-9223372036854775808, maximum:9223372036854775807</pre>	long long
<pre>"type":"integer", minimum:0, maximum:18446744073709551615</pre>	unsigned long long

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre>"type": "number", "maximum": m, "minimum": n, "exclusiveMaximum": true, "exclusiveMinimum": true, "multipleOf": n</pre>	<p>"maximum"-, "minimum"-, "exclusiveMaximum"- und "exclusiveMinimum"-Einschränkungen werden nur in Form eines Kommentars an die Sprachstruktur übergeben.</p> <p>"multipleOf" wird ignoriert.</p>
<pre>"type": "number" "format": "float"</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <ul style="list-style-type: none"> char[32] <p>Zuordnungsebene 1.2 und höher:</p> <ul style="list-style-type: none"> float(*) <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt Gleitkommatentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>
<pre>"type": "number" "format": "double"</pre>	<p>Zuordnungsebene 1.0 und niedriger:</p> <ul style="list-style-type: none"> char[32] <p>Zuordnungsebene 1.2 und höher:</p> <ul style="list-style-type: none"> double(*) <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für doppelte Datentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt doppelter Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

JSON-Schemaschlüsselwort	C- und C++-Datentyp
<pre> oneOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] anyOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] allOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] </pre>	<p>Die logischen Pfade durch das Array von Optionen werden so zusammengeführt, als ob ein einzelnes Objekt vom Typ { "properties": { "A":..., "B":... } } definiert wäre. Jede dieser JSON-Eigenschaften wird als optional behandelt. Wenn die Unteroptionen widersprüchliche Definitionen für denselben Eigenschaftsnamen enthalten, wird eine Fehlermeldung ausgegeben. Wenn eine einzelne Eigenschaft beispielsweise sowohl als Zeichenfolge (in Pfad A) als auch als Objekt (in Pfad B) definiert ist, wird diese Definition nicht unterstützt und führt zu einer Fehlermeldung.</p> <p>Es ist möglich, dass ein JSON-Schema komplexe logische Strukturen definiert. Die in den komplexen logischen Strukturen implizierten Feinheiten können jedoch bei der Zuordnung zu einer Sprachstruktur verloren gehen. Der Umsetzungsprozess versucht nicht, die kombinatorischen Regeln aus dem Schema umzusetzen. Es interagiert nur mit den Sprachstrukturfeldern, die angeben, ob eine bestimmte JSON-Eigenschaft vorhanden oder nicht vorhanden ist.</p> <p>Wenn die Unteroptionen kompatible Definitionen für denselben Eigenschaftsnamen enthalten, versucht DFHJS2LS, das zugeordnete Muster von Einschränkungen zusammenzuführen, wobei allerdings einige Feinheiten verloren gehen können. Nehmen wir beispielsweise die folgende Definition an:</p> <pre> "A": { "oneOf": [{ "type": "string", "maxLength": 5 }, { "type": "string", "minLength": 7, "maxLength": 8 }] } </pre> <p>„A“ ist entweder als eine Zeichenfolge mit einer Länge von bis zu 5 Zeichen oder als eine Zeichenfolge mit einer Länge zwischen 7 und 8 Zeichen definiert. Die Zusammenführung kann zu Zuordnungen zu einer Zeichenfolge mit einer Länge von 0 bis 8 Zeichen führen, ohne dass erkannt wird, dass die Länge von 6 Zeichen ungültig ist.</p> <p>Die meisten Szenarios mit logischer Zusammensetzung werden einfachen Sprachstrukturen zugeordnet, aber bei einer komplizierten logischen Zusammensetzung werden während des Zuordnungsprozesses möglicherweise Kompromisse geschlossen. Die besten Ergebnisse erhalten Sie, wenn Sie die logische Zusammensetzung in einem JSON-Schema zum Definieren alternativer Deklarationen für dieselbe JSON-Eigenschaft vermeiden.</p>

Anmerkung: CICS kann keine ganzzahligen Werte größer als den maximalen Wert für 'signed long' ($2^{63} - 1$) umsetzen, es sei denn, sie sind in Anführungszeichen eingeschlossen.

Anmerkung: Minimale und maximale Werte, die in dem Schema für numerische Typen angegeben sind, werden nur zum Zuordnen zu einem C- oder C++-Datentyp verwendet. Die Daten werden während der Laufzeit nicht gegen diese Werte geprüft.

Zuordnung von PL/I zu JSON-Schema:

Das Dienstprogramm DFHLS2JS unterstützt Zuordnungen zwischen PL/I-Datenstrukturen und JSON-Schemadefinitionen. Da sich der Enterprise PL/I-Compiler von älteren PL/I-Compilern unterscheidet, werden zwei Sprachoptionen unterstützt: PLI-ENTERPRISE und PLI-OTHER.

PL/I-Namen werden entsprechend den folgenden Regeln in JSON-Namen konvertiert:

1. Zeichen, die in JSON-Eigenschaftsnamen nicht gültig sind, werden durch x ersetzt.
Beispiel: `monthly$total` wird zu `monthlyxtotal`.
2. Doppelte Namen werden durch Hinzufügen einer oder mehrerer Ziffern eindeutig gemacht.
Zwei Instanzen von `year` werden zu `year` und `year1`.

DFHLS2JS ordnet PL/I-Datentypen Schemaelementen entsprechend der folgenden Tabelle zu. PL/I-Typen, die in der Tabelle nicht enthalten sind, werden von DFHLS2JS nicht unterstützt. Es gelten außerdem die folgenden Einschränkungen:

- Datenelemente mit dem Attribut **COMPLEX** werden nicht unterstützt.
- Datenelemente mit dem Attribut **FLOAT** werden auf Zuordnungsebene 1.2 oder höher unterstützt. Enterprise PL/I **FLOAT IEEE** wird nicht unterstützt.
- Die reinen DBCS-Zeichenfolgen **VARYING** und **VARYINGZ** werden auf Zuordnungsebene 1.2 oder höher unterstützt.
- Datenelemente, die als **DECIMAL**(*p* , *q*) angegeben sind, werden nur unterstützt, wenn $p \geq q$.
- Datenelemente, die als **BINARY**(*p* , *q*) angegeben sind, werden nur unterstützt, wenn $q = 0$.
- Wenn das Attribut **PRECISION** für ein Datenelement angegeben ist, wird es ignoriert.
- **PICTURE**-Zeichenfolgen werden nicht unterstützt.
- **ORDINAL**-Datenelemente werden als **FIXED BINARY(7)**-Datentypen behandelt.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf **NULL** festgelegt ist, werden Zeichenarrays einem **string**-Element zugeordnet und als auf null endende Zeichenfolgen verarbeitet. Diese Zuordnung gilt nicht für Enterprise PL/I.
- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf **BINARY** festgelegt ist, werden Zeichenarrays einem **string**-Element zugeordnet und als binäre Daten verarbeitet.

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf COLLAPSE gesetzt ist, werden führende und abschließende Leerzeichen entfernt, und mehrere Leerzeichen werden durch ein einzelnes Leerzeichen ersetzt.

DFHLS2JS implementiert die Padding-Algorithmen von PL/I nicht vollständig, deshalb müssen Sie Padding-Bytes in Ihrer Datenstruktur explizit deklarieren. DFHLS2JS gibt eine Nachricht aus, wenn es erkennt, dass Padding-Bytes fehlen. Jede übergeordnete Struktur muss an einer Doppelwortgrenze beginnen und jedes Byte in der Struktur muss der korrekten Grenze zugeordnet werden. Sehen Sie sich dieses Codefragment an:

```
3 FIELD1 FIXED BINARY(7),
3 FIELD2 FIXED BINARY(31),
3 FIELD3 FIXED BINARY(63);
```

In diesem Beispiel gilt Folgendes:

- FIELD1 ist 1 Byte lang und kann an jeder Grenze ausgerichtet werden.
- FIELD2 ist 4 Bytes lang und muss an einer Vollwortgrenze ausgerichtet werden.
- FIELD3 ist 8 Bytes lang und muss an einer Doppelwortgrenze ausgerichtet werden.

Der Enterprise PL/I-Compiler richtet die Felder in der folgenden Reihenfolge aus:

1. FIELD3 wird zuerst ausgerichtet, weil es die stärksten Anforderungen an die Grenze hat.
2. FIELD2 wird an der Vollwortgrenze unmittelbar vor FIELD3 ausgerichtet.
3. FIELD1 wird an der Bytegrenze unmittelbar vor FIELD3 ausgerichtet.

Schließlich fügt der Compiler drei Padding-Bytes unmittelbar vor FIELD1 ein, so dass die gesamte Struktur an einer Vollwortgrenze ausgerichtet ist.

Da DFHLS2JS keine äquivalenten Padding-Bytes einfügt, müssen Sie diese explizit deklarieren, bevor die Struktur von DFHLS2JS verarbeitet wird. Beispiel:

```
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);
```

Alternativ können Sie die Struktur ändern, um alle Felder als nicht ausgerichtet zu deklarieren und die Anwendung erneut zu kompilieren, die die Struktur verwendet. Weitere Informationen zu den Anforderungen an die strukturelle PL/I-Speicherausrichtung finden Sie unter Enterprise PL/I for z/OS product information.

PL/I-Datenbeschreibung	JSON-Schemadefinition
FIXED BINARY (<i>n</i>)	<pre>"type":"integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i></pre> <p>Dabei ist <i>n</i> der Maximalwert, der durch das Basiselement dargestellt werden kann.</p>

PL/I-Datenbeschreibung	JSON-Schemadefinition
UNSIGNED FIXED BINARY(n) Restriction: Nur Enterprise PL/I	<pre>"type": "integer", "minimum": 0, "maximum": n</pre> <p>Dabei ist n der Maximalwert, der durch das Basiselement dargestellt werden kann.</p>
FIXED DECIMAL(n , m)	<pre>"type": "number", "format": "decimal", "multipleOf": x , "maximum": y , "minimum": - z</pre> <p>Dabei gilt Folgendes:</p> <p>x ist die kleinste verfügbare Einheit = $1 / 10^m$.</p> <p>y ist der maximale Wert, der durch die Kombination von n und m dargestellt werden kann.</p> <p>z ist der maximale Wert, der durch die Kombination von n und m dargestellt werden kann.</p>
FIXED DECIMAL(15) Unterstützt auf Zuordnungsebene 3.0 und höher, wenn DATETIME=PACKED15	<pre>"type": "string", "format": "date-time"</pre> <p>Das Format der Zeitmarke wird von RFC3339 definiert.</p>
BIT(n) Dabei ist n ein Vielfaches von 8. Andere Werte werden nicht unterstützt.	<pre>"type": "string" "maxLength": m</pre> <p>Dabei ist $m = n / 8$</p>
CHARACTER(n) VARYING und VARYINGZ werden auch auf Zuordnungsebene 1.2 und höher unterstützt. Restriction: VARYINGZ wird nur von Enterprise PL/I unterstützt.	<pre>"type": "string" "maxLength": n</pre>
GRAPHIC(n) VARYING und VARYINGZ werden auch auf Zuordnungsebene 1.2 und höher unterstützt. Restriction: VARYINGZ wird nur von Enterprise PL/I unterstützt.	<p>Auf Zuordnungsebene 1.0 und 1.1, wobei $m = 2 * n$:</p> <pre>"type": "string" "maxLength": m</pre> <p>Auf Zuordnungsebene 1.2 oder höher:</p> <pre>"type": "string" "maxLength": n</pre>

PL/I-Datenbeschreibung	JSON-Schemadefinition
<p>WIDECHAR(<i>n</i>)</p> <p>Restriction: Nur Enterprise PL/I</p>	<p>Auf Zuordnungsebene 1.0 und 1.1, wobei $m = 2 * n$:</p> <pre>"type": "string" "maxLength": m</pre> <p>Auf Zuordnungsebene 1.2 oder höher:</p> <pre>"type": "string" "maxLength": n</pre> <p>Auf Zuordnungsebene 4.0 und höher füllt CICS das Anwendungsdatenstrukturfeld mit UTF-16-Daten.</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>Restriction: Nur Enterprise PL/I</p>	<pre>"type": "integer", "minimum": 0, "maximum": 255</pre>
<p>BINARY FLOAT(<i>n</i>)</p> <p>Dabei ist $n \leq 21$.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt BINARY FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre>"type": "number", "format": "float"</pre>

PL/I-Datenbeschreibung	JSON-Schemadefinition
<p>BINARY FLOAT(<i>n</i>)</p> <p>Dabei ist $21 < n \leq 53$.</p> <p>Werte größer als 53 werden nicht unterstützt.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt BINARY FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre>"type": "number", "format": "double"</pre>
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>Dabei ist $n \leq 6$.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre>"type": "number", "format": "float"</pre>

PL/I-Datenbeschreibung	JSON-Schemadefinition
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>Dabei ist $6 < n \leq 16$.</p> <p>Werte größer als 16 werden nicht unterstützt.</p> <p>Unterstützt auf Zuordnungsebene 1.2 und höher.</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>	<pre>"type":"number", "format":"double"</pre>
<p><i>name</i> (<i>n</i>) <i>datenbeschreibung</i></p>	<p>Für Basiselemente:</p> <pre>"type":"array" "maxItems": n "minItems": n "items":{ "type":"object", "properties":{ name : { datenbeschreibung_JSON } } "required":[name] }</pre> <p>Für Datendeklarationen:</p> <pre>"type":"array" "maxItems": n "minItems": n "items":{ datenbeschreibung_JSON }</pre> <p>Dabei ist <i>datenbeschreibung_JSON</i> die JSON-Schemadarstellung der PL/I-Datenbeschreibung.</p>

Zuordnung von JSON-Schema zu PL/I:

Das Dienstprogramm DFHJS2LS unterstützt Zuordnungen zwischen JSON-Schemas und PL/I-Datenstrukturen. Da sich der Enterprise PL/I-Compiler von älteren PL/I-Compilern unterscheidet, werden zwei Sprachoptionen unterstützt: PLI-ENTERPRISE und PLI-OTHER.

Regeln für die Zuordnung von Schemaelementnamen zu PL/I

Die CICS-Assistenten generieren eindeutige und gültige Namen für PL/I-Variablen aus den Schemaelementnamen mithilfe der folgenden Regeln:

1. Alle Zeichen außer A-Z, a-z, 0-9, @, #, _ oder \$ werden ersetzt durch X.

Beispiel: `monthly-total` wird zu `monthlyXtotal`.

Sie können den Parameter **MAPPING-OVERRIDES** verwenden, um anzupassen, wie andere Zeichen behandelt werden. Wenn Sie beispielsweise den Wert **HYPHENS-AS-UNDERSCORES** festlegen, werden alle Bindestriche im JSON-Schema in einen Unterstrich statt in ein X konvertiert. Beispiel: `monthly-total` wird zu `monthly_total`.

2. Doppelte Namen in demselben Bereich werden durch das Hinzufügen von ein oder zwei numerischen Ziffern zu der zweiten Instanz und zu allen nachfolgenden Instanzen des Namens eindeutig gemacht.

Beispiel: Drei Instanzen von `year` werden zu `year`, `year1` und `year2`.

Sollte das oben beschriebene Verhalten unerwünscht sein, kann der Benutzer **MAPPING-OVERRIDES=NO-ARRAY-NAME-INDEXING** als Eingabe für das Dienstprogramm angeben, die das Hinzufügen einer oder mehrerer Ziffern zur zweiten Instanz und allen nachfolgenden Instanzen des Namens inaktiviert.

3. Ein JSON-Schema gibt an, dass eine Variable eine variierende Kardinalität hat, wenn das Element "type" den Wert "array" hat und wenn die Schlüsselwörter "minItems" und "maxItems" nicht angegeben sind oder abweichende Werte aufweisen. Wenn das Schema angibt, dass die Variable eine variierende Kardinalität hat, werden Feldnamen mit den Suffixes "_cont" und "_num" erstellt.

Weitere Informationen finden Sie unter „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.

4. Ein JSON-Schema gibt an, dass eine Variable optional ist, wenn sie nicht in dem "required"-Schlüsselwortarray enthalten ist, das dem einschließenden JSON-Schema vom Typ "object" zugeordnet ist. Bei optionalen Feldern wird ein zusätzliches Feld mit dem Suffix _num zu dem Elementnamen hinzugefügt. Zur Laufzeit ist dies gleich null, um anzugeben, dass der Wert in den JSON-Daten gefehlt hat, und ungleich null, wenn der Wert in den JSON-Daten vorhanden war.
5. Feldnamen sind auf 31 Zeichen begrenzt. Wenn ein generierter Name, einschließlich aller Präfixe und Suffixe, diese Länge überschreitet, wird der Elementname abgeschnitten.

Der resultierende Name ist höchstens 31 Zeichen lang.

Regeln für die Zuordnung von Schematypen zu PL/I

DFHJS2LS ordnet Schematypwerte zu PL/I-Datentypen entsprechend der folgenden Tabelle zu. Beachten Sie außerdem die folgenden Punkte:

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher und der Parameter **CHAR-VARYING** auf NULL gesetzt ist, werden Zeichendaten mit variabler Länge auf null endenden Zeichenfolgen zugeordnet, und ein zusätzliches Zeichen wird dem Nullabschlusszeichen zugeordnet.

- Wenn der Parameter **MAPPING-LEVEL** auf 1.2 oder höher festgelegt und der Parameter **CHAR-VARYING** nicht angegeben ist, werden standardmäßig Zeichendaten mit variabler Länge zu einem VARYINGZ-Datentyp für Enterprise PL/I und einem VARYING-Datentyp für andere PL/I-Versionen zugeordnet.
- Binärdaten mit variabler Länge werden einem VARYING-Datentyp zugeordnet, wenn sie kleiner als 32.768s Byte sind, und einem Container, wenn sie größer als 32.768 Bytes sind.

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
Alle: "type": "null" "type": []	Nicht unterstützt
"\$schema": "http://json-schema.org/draft-04/schema#"	Dieses Schlüsselwort wird ignoriert, es wird jedoch davon ausgegangen, dass es mit der JSON-Schemaspezifikation 'draft-04' kompatibel ist.
"title": "same text" "description": "more text"	Diese Schlüsselwörter werden ignoriert.
"format": "<vordefinierte werte>"	Das Schlüsselwort "format" wird verwendet, um entweder die generierte Struktur oder den Laufzeitwert zu ändern. Lesen Sie die folgenden Informationen zur unterstützten Verwendung von "format".
<div> <div> </div> <div> "array", "items": {<JSON-unterschema>}, "additionalItems": false, "maxItems": m , "minItems": n </div> </div>	<p>Mehrdimensionale Arrays werden ab Zuordnungsebene 4.3 aufwärts unterstützt, und Arrays mit gemischtem Typ werden nicht unterstützt.</p> <p>"additionalItems" wird als falsch angenommen und es wird kein anderer Wert unterstützt.</p> <p>Wenn sowohl "minItems" als auch "maxItems" vorhanden und gleichwertig sind, wird das Array als feste Kardinalität behandelt. Ist dies nicht der Fall, wird es als variierende Kardinalität behandelt. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.</p>
"type": "array", "uniqueItems": true	"uniqueItems" wird mit JSON-Arrays nicht unterstützt. Das <JSON-unterschema> muss einen unterstützten Typ ("type") definieren, aber bei diesem Typ kann es sich nicht um ein Array ("array") handeln. Dies ist eine Einschränkung für die generierte Sprachstruktur.

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre>"type": "object", "additionalProperties": false, "properties": { ["<elementname>": {<JSON-unterschema>} [,]]* } "required": [["<elementname>" [,]]*]</pre>	<p>Das einzige JSON-Objekt, das derzeit unterstützt wird, ist eine feste Gruppe benannter Elemente.</p> <p>Dieses generiert eine Struktur (oder Unterstruktur), die diese Elementnamen verwendet.</p> <p>Für "additionalProperties" wird false angenommen, wenn kein Wert im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist.</p> <p>Alle Elemente im Objekt "properties" werden als "optional" angenommen, wenn sie sich nicht im Array "required" befinden oder wenn kein Array "required" vorhanden ist. Ein optionales Element ("optional") erhält eine variable Ordinalität von null bis X. Dabei ist X entweder 1 oder die maximale Anzahl von Elementen im Array, vorausgesetzt, das Element ist als Array definiert. Siehe „Variable Arrays von Elementen in DFHJS2LS“ auf Seite 304.</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre>"type": "object", "additionalProperties": true</pre>	<p>Das Objekt wird wie im vorherigen Beispiel zugeordnet, mit zusätzlichen Feldern, die zusätzliche Eigenschaften unterstützen. Für die Eigenschaft <code>additionalProperties</code> wird 'false' angenommen, wenn sie nicht im Parameter ADDITIONAL-PROPERTIES-DEFAULT festgelegt ist.</p> <p>Ist diese Option aktiviert, werden bis zu dem im Parameter ADDITIONAL-PROPERTIES-MAX angegebenen Wert Leerzeichen zugeordnet. Die Anzahl der Zeichen in jedem Leerzeichen wird durch den Parameter ADDITIONAL-PROPERTIES-SIZE festgelegt. Jede Eigenschaft wird einem Feld PIC X(z) zugeordnet, wobei z durch den Parameter ADDITIONAL-PROPERTIES-SIZE definiert wird. Ist der Wert von ADDITIONAL-PROPERTIES-MAX größer als 0, werden die Eigenschaften in ähnlicher Weise zugeordnet wie in einem Array, in dem maxItems festgelegt ist.</p> <p>Anmerkung: Es gibt mehrere Möglichkeiten, JSON-Unterstützung in CICS zu konfigurieren, einschließlich der Verwendung von z/OS Connect for CICS. Wenn Sie die ältere CICS-Java-Pipeline-Technologie nutzen, werden <code>AdditionalProperties</code> nur unterstützt, falls die <code>com.ibm.cics.json.enableAxis2Handlers</code>-JVM-Systemeigenschaft nicht auf 'true' festgelegt ist.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>Keines dieser Schlüsselwörter wird mit JSON-Objekten unterstützt.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression>", "minLength": l</pre>	<p><code>char[z]</code></p> <p>Dabei basiert der Wert von z auf m , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p>m basiert auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>"pattern"- und "minLength"-Einschränkungen werden über die Sprachstruktur nur in Form eines Kommentars übergeben.</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre>"type":"string" "maxLength": m</pre>	<p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>WIDECHAR(<i>z</i>)</p> <p>Dabei basiert der Wert von <i>z</i> auf <i>m</i> , ist aber abhängig von den Einstellungen des Parameters CHAR-VARYING.</p> <p><i>m</i> basiert auf dem Schlüsselwort "maxLength " und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*":{ "type":"string", "format":"date-time" }</pre>	<p>FIXED DECIMAL (15,0)</p> <p>Alle unterstützt, wenn DATETIME=PACKED15</p>
<pre>"*name*":{ "type":"string", "format":"uri" }</pre>	<p>CHAR(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>WIDECHAR(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt.</p>
<pre>"*name*":{ "type":"string", "format":"base64Binary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"hexBinary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength".</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre>"*name*":{ "type":"string", "format":"<vordefiniert>" }</pre>	<p>CHAR(<i>m</i>) Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist entweder <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i>. Ein relevantes Muster ("pattern") wird an den Kommentar übergeben.</p> <p>Wenn CCSID=1200 auf Zuordnungsebene 4.0 und höher:</p> <p>WIDECHAR(<i>m</i>)</p> <p>Dabei basiert <i>m</i> auf dem Schlüsselwort "maxLength" und wird als Zeichenfolge fester Länge behandelt. Und <vordefiniert> ist <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> oder <i>ipv6</i> . Ein relevantes Muster ("pattern") wird verwendet und an den Kommentar übergeben.</p>
<pre>"type":"boolean"</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Andere PL/I-Versionen FIXED BINARY (7)</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I BIT(7) BIT(1)</p> <p>Andere PL/I-Versionen BIT(7) BIT(1)</p> <p>Dabei wird BIT(7) für die Ausrichtung angegeben und BIT(1) enthält den zugeordneten booleschen Wert.</p>
<pre>"type": "integer", "exclusiveMaximum": true, "exclusiveMinimum": true, "multipleOf": n</pre>	<p>"exclusiveMaximum"- und "exclusiveMinimum"-Einschränkungen werden nur in Form eines Kommentars an die Sprachstruktur übergeben.</p> <p>"multipleOf" wird ignoriert.</p>
<pre>"type":"integer", minimum:-128, maximum:127</pre>	<p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Andere PL/I-Versionen FIXED BINARY (7)</p>
<pre>"type":"integer", minimum:0, maximum:255</pre>	<p>Enterprise PL/I UNSIGNED FIXED BINARY (8)</p> <p>Andere PL/I-Versionen FIXED BINARY (8)</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
"type": "integer", minimum: -32768, maximum: 32767	Enterprise PL/I SIGNED FIXED BINARY (15) Andere PL/I-Versionen FIXED BINARY (15)
"type": "integer", minimum: 0, maximum: 65535	Enterprise PL/I UNSIGNED FIXED BINARY (16) Andere PL/I-Versionen FIXED BINARY (16)
"type": "integer", minimum: -2147483648, maximum: 2147483647	Enterprise PL/I SIGNED FIXED BINARY (31) Andere PL/I-Versionen FIXED BINARY (31)
"type": "integer", minimum: 0, maximum: 4294967295	Zuordnungsebene 1.1 und niedriger: Enterprise PL/I UNSIGNED FIXED BINARY (32) Zuordnungsebene 1.2 und höher: Enterprise PL/I CHAR(<i>y</i>) Dabei ist <i>y</i> eine feste Länge kleiner 16 MB. Alle Zuordnungsebenen: Andere PL/I-Versionen BIT (64)

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre>"type": "integer", minimum: -9223372036854775808, maximum: 9223372036854775807</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p>Enterprise PL/I SIGNED FIXED BINARY(63) Anmerkung: Die LIMITS-Compilersteueranweisung kann sich auf die Interpretation dieses Felds durch den PL/I-Compiler auswirken. CICS erwartet eine deklarierte Größe dieses Felds, aber der Compiler optimiert das Feld unter Umständen, indem er es verkleinert, was zu einer Abweichung führt. Vermeiden Sie solche Probleme, indem Sie die Kompilierzeioption LIMITS(FIXEDBIN(63)) verwenden.</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I CHAR(y)</p> <p>Dabei ist <i>y</i> eine feste Länge kleiner 16 MB.</p> <p>Alle Zuordnungsebenen:</p> <p>Andere PL/I-Versionen BIT(64)</p>
<pre>"type": "integer", minimum: 0, maximum: 18446744073709551615</pre>	<p>Zuordnungsebene 1.1 und niedriger:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(64) Anmerkung: Die LIMITS-Compilersteueranweisung kann sich auf die Interpretation dieses Felds durch den PL/I-Compiler auswirken. CICS erwartet eine deklarierte Größe dieses Felds, aber der Compiler optimiert das Feld unter Umständen, indem er es verkleinert, was zu einer Abweichung führt. Vermeiden Sie solche Probleme, indem Sie die Kompilierzeioption LIMITS(FIXEDBIN(63)) verwenden.</p> <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I CHAR(y)</p> <p>Dabei ist <i>y</i> eine feste Länge kleiner 16 MB.</p> <p>Andere PL/I-Versionen BIT(64)</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
"type": "number" "description": "decimal"	FIXED DECIMAL(<i>n</i> , <i>m</i>)
"type": "number" "description": "float"	<p>Zuordnungsebenen 1.0 und 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I DECIMAL FLOAT(6) HEXADEC</p> <p>Andere PL/I-Versionen DECIMAL FLOAT(6)</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>
"type": "number" "description": "double"	<p>Zuordnungsebenen 1.0 und 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Zuordnungsebene 1.2 und höher:</p> <p>Enterprise PL/I DECIMAL FLOAT(16) HEXADEC</p> <p>Andere PL/I-Versionen DECIMAL FLOAT(16)</p> <p>Anmerkung: Die IBM HFP-Datendarstellung (Hexadecimal Floating Point) entspricht nicht genau der IEEE-754-1985-Darstellung, die für JSON verwendet wird. Manche Werte werden möglicherweise von einer Darstellung in die andere nicht exakt konvertiert. Einige extrem große oder kleine Werte sind für Gleitkommatentypen möglicherweise nicht gültig. Manche Werte können ihre Genauigkeit verlieren, wenn sie aus der oder in die HFP-Darstellung konvertiert werden. Wenn präzises Konvertieren wichtig ist, ziehen Sie in Betracht, statt DECIMAL FLOAT-Datentypen Alternativen mit eindeutiger Genauigkeit zu verwenden.</p>

JSON-Schemaschlüsselwort	PL/I-Datenbeschreibung
<pre> oneOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] anyOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] allOf: [{ "properties": { "A": {...} } } , { "properties": { "B": {...} } }] </pre>	<p>Die logischen Pfade durch das Array von Optionen werden so zusammengeführt, als ob ein einzelnes Objekt vom Typ { "properties": { "A":..., "B":... } } definiert wäre. Jede dieser JSON-Eigenschaften wird als optional behandelt. Wenn die Unteroptionen widersprüchliche Definitionen für denselben Eigenschaftsnamen enthalten, wird eine Fehlermeldung ausgegeben. Wenn eine einzelne Eigenschaft beispielsweise sowohl als Zeichenfolge (in Pfad A) als auch als Objekt (in Pfad B) definiert ist, wird diese Definition nicht unterstützt und führt zu einer Fehlermeldung.</p> <p>Es ist möglich, dass ein JSON-Schema komplexe logische Strukturen definiert. Die in den komplexen logischen Strukturen implizierten Feinheiten können jedoch bei der Zuordnung zu einer Sprachstruktur verloren gehen. Der Umsetzungsprozess versucht nicht, die kombinatorischen Regeln aus dem Schema umzusetzen. Es interagiert nur mit den Sprachstrukturfeldern, die angeben, ob eine bestimmte JSON-Eigenschaft vorhanden oder nicht vorhanden ist.</p> <p>Wenn die Unteroptionen kompatible Definitionen für denselben Eigenschaftsnamen enthalten, versucht DFHJS2LS, das zugeordnete Muster von Einschränkungen zusammenzuführen, wobei allerdings einige Feinheiten verloren gehen können. Nehmen wir beispielsweise die folgende Definition an:</p> <pre> "A": { "oneOf": [{ "type": "string", "maxLength": 5 }, { "type": "string", "minLength": 7, "maxLength": 8 }] } </pre> <p>„A“ ist entweder als eine Zeichenfolge mit einer Länge von bis zu 5 Zeichen oder als eine Zeichenfolge mit einer Länge zwischen 7 und 8 Zeichen definiert. Die Zusammenführung kann zu Zuordnungen zu einer Zeichenfolge mit einer Länge von 0 bis 8 Zeichen führen, ohne dass erkannt wird, dass die Länge von 6 Zeichen ungültig ist.</p> <p>Die meisten Szenarios mit logischer Zusammensetzung werden einfachen Sprachstrukturen zugeordnet, aber bei einer komplizierten logischen Zusammensetzung werden während des Zuordnungsprozesses möglicherweise Kompromisse geschlossen. Die besten Ergebnisse erhalten Sie, wenn Sie die logische Zusammensetzung in einem JSON-Schema zum Definieren alternativer Deklarationen für dieselbe JSON-Eigenschaft vermeiden.</p>

Anmerkung: CICS kann keine ganzzahligen Werte größer als den maximalen Wert für 'signed long' ($2^{63} - 1$) umsetzen, es sei denn, sie sind in Anführungszeichen eingeschlossen.

Anmerkung: Minimale und maximale Werte, die in dem Schema für numerische Typen angegeben sind, werden nur zum Zuordnen zu einem PL/I-Datentyp verwendet. Die Daten werden während der Laufzeit nicht gegen diese Werte geprüft.

Variable Arrays von Elementen

XML kann ein Array mit einer variierenden Anzahl von Elementen enthalten. Allgemein lassen sich WSDL-Dokumente und XML-Schemas, die eine variierende Anzahl von Elementen enthalten, nicht effizient einer einzelnen Datenstruktur einer höheren Programmiersprache zuordnen. CICS verwendet containerbasierte Zuordnungen oder Inlinezuordnungen, um eine variierende Anzahl von Elementen in XML zu verarbeiten.

Ein Array mit einer variierenden Anzahl von Elementen wird im XML-Schema mithilfe der Attribute `minOccurs` und `maxOccurs` in der Elementdeklaration dargestellt:

- Das Attribut `minOccurs` gibt an, wie oft das Element mindestens vorkommen kann. Es hat den Wert 0 oder eine beliebige positive ganze Zahl.
- Das Attribut `maxOccurs` gibt an, wie oft das Element höchstens vorkommen kann. Es kann als Wert jede positive ganze Zahl größer-gleich dem Wert des Attributs `minOccurs` haben. Es kann auch den Wert `unbounded` haben, der angibt, dass für die Anzahl der Vorkommen des Elements keine Obergrenze gilt.
- Der Standardwert für beide Attribute ist 1.

In diesem Beispiel wird eine 8-Byte-Zeichenfolge dargestellt, die optional ist. Das heißt, sie kann gar nicht oder einmal in der Anwendungs-XML oder der SOAP-Nachricht vorkommen:

```
<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>
```

Im folgenden Beispiel wird eine 8-Byte-Zeichenfolge dargestellt, die mindestens einmal auftreten muss:

```
<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>
```

Allgemein lassen sich WSDL-Dokumente, die eine variierende Anzahl von Elementen enthalten, nicht effizient einer einzelnen Datenstruktur einer höheren Programmiersprache zuordnen. Für diese Fälle verwendet CICS folglich eine Reihe von verbundenen Datenstrukturen, die in einer Reihe von Containern an das Anwendungsprogramm übergeben werden. Diese Strukturen werden als Eingabe und Ausgabe der Anwendung verwendet:

- Wenn CICS XML in Anwendungsdaten umsetzt, füllt es diese Strukturen mit den Anwendungsdaten, und die Anwendung liest sie.
- Wenn CICS die Anwendungsdaten in XML umsetzt, liest es die Anwendungsdaten in den Strukturen, die von der Anwendung gefüllt wurden.

Das Format dieser Datenstrukturen lässt sich am besten anhand einer Reihe von Beispielen erklären. Die XML kann aus einer SOAP-Nachricht oder aus einer Anwendung stammen. Diese Beispiele verwenden ein Array von einfachen 8-Byte-Feldern. Das Modell unterstützt jedoch Arrays komplexer Datentypen ebenso wie Arrays von Datentypen, die andere Arrays enthalten.

Feste Anzahl von Elementen

Das erste Beispiel stellt ein Element dar, das genau dreimal vorkommt.

```
<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>
```

Da in diesem Beispiel die Anzahl der Vorkommen des Elements vorab bekannt ist, kann es als Array fester Länge in einer einfachen COBOL-Deklaration (oder dem Äquivalent in anderen Sprachen) dargestellt werden:

```
05 component PIC X(8) OCCURS 3 TIMES
```

Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner

Dieses Beispiel stellt ein obligatorisches Element dar, das ein- bis fünfmal vorkommen kann:

```
<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>
```

Die Hauptdatenstruktur enthält eine Deklaration mit zwei Feldern. Wenn CICS die XML in binäre Daten umsetzt, enthält das erste Feld, component-num, die Anzahl von Vorkommen des Elements in der XML und das zweite Feld, component-cont, den Namen eines Containers:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Eine zweite Datenstruktur enthält die Deklaration des Elements selbst:

```
01 DFHWS-component
02 component PIC X(8)
```

Sie müssen den Wert von component-num untersuchen (Wert im Bereich zwischen 1 und 5), um herauszufinden, wie oft das Element vorkommt. Der Elementinhalt befindet sich im Container namens component-cont. Der Container enthält ein Array von Elementen, wobei jedes Element von der Datenstruktur DFHWS-component zugeordnet ist.

Wenn `minOccurs="0"` und `maxOccurs="1"`, ist das Element optional. Um die Datenstruktur in Ihrem Anwendungsprogramm zu verarbeiten, müssen Sie den Wert von `component-num` untersuchen:

- Ist er 0, hat die Nachricht kein Komponentenelement und der Inhalt von `component-cont` ist nicht definiert.
- Ist er 1, befindet sich das Komponentenelement im Container namens `component-cont`.

Der Inhalt des Containers wird von der Datenstruktur `DFHWS-component` zugeordnet.

Anmerkung: Wenn die SOAP-Nachricht aus einem einzelnen wiederkehrenden Element besteht, generiert `DFHWS2LS` zwei Sprachstrukturen. Die Hauptsprachstruktur enthält die Anzahl von Elementen in dem Array und den Namen eines Containers, der das Array von Elementen enthält. Die zweite Sprachstruktur ordnet eine einzelne Instanz des wiederkehrenden Elements zu.

Variierende Anzahl von Elementen auf Zuordnungsebene 2.1 und höher

Auf Zuordnungsebene 2.1 und höher können Sie den Parameter **INLINE-MAXOCCURS-LIMIT** in den CICS-Assistenten verwenden. Der Parameter **INLINE-MAXOCCURS-LIMIT** gibt an, wie die variierende Anzahl von Elementen verarbeitet wird. Bei den Zuordnungsoptionen für eine variierende Anzahl von Elementen handelt es sich containerbasierte Zuordnung, wie unter „Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner“ auf Seite 373 beschrieben, oder um eine Inline-Zuordnung. Der *wert* dieses Parameters kann eine positive ganze Zahl im Bereich von 0 bis 32.767 sein:

- Der Standardwert von **INLINE-MAXOCCURS-LIMIT** ist 1, was sicherstellt, dass optionale Elemente inline zugeordnet werden.
- Der Wert 0 für den Parameter **INLINE-MAXOCCURS-LIMIT** verhindert eine Inline-Zuordnung.
- Wenn `maxOccurs` kleiner-gleich dem Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die Inline-Zuordnung verwendet.
- Wenn `maxOccurs` größer als der Wert von **INLINE-MAXOCCURS-LIMIT** ist, wird die containerbasierte Zuordnung verwendet.

Die Inline-Zuordnung einer variierenden Anzahl von Elementen führt dazu, dass sowohl ein Array (wie in dem obigen Beispiel für die feste Anzahl von Vorkommen) als auch ein Zähler generiert wird. Das Feld `component-num` gibt an, wie viele Instanzen des Elements vorhanden sind, und auf diese wird vom Array verwiesen. Für das Beispiel in „Variierende Anzahl von Elementen auf Zuordnungsebene 2 und kleiner“ auf Seite 373, bei dem **INLINE-MAXOCCURS-LIMIT** kleiner-gleich 5 ist, sieht die generierte Datenstruktur wie folgt aus:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

Das erste Feld, `component-num`, ist identisch mit der Ausgabe für das Beispiel für die containerbasierte Zuordnung im vorherigen Abschnitt. Das zweite Feld enthält ein Array mit der Länge 5, das groß genug ist, um die maximale Anzahl von Elementen, die generiert werden können, zu enthalten.

Die Inline-Zuordnung unterscheidet sich von der containerbasierten Zuordnung, die die Anzahl von Vorkommen des Elements und den Namen des Containers, in dem die Daten enthalten sind, speichert, insofern als dass sie alle Daten in dem aktuellen Container speichert. Durch das Speichern der Daten im aktuellen Container

wird die Leistung im Allgemeinen verbessert, was die Inline-Zuordnung zur präferierten Lösung macht.

Verschachtelte variierende Arrays

Komplexe WSDL-Dokumente und XML-Schemas können variierende wiederkehrende Elemente enthalten, die ihrerseits wiederum variierende wiederkehrende Elemente enthalten können. In diesem Fall erstreckt sich die beschriebene Struktur über die beiden in den Beispielen beschriebenen Ebenen hinaus.

Dieses Beispiel stellt ein optionales Element namens `<component2>` dar, das in einem obligatorischen Element namens `<component1>` verschachtelt ist, wobei das obligatorische Element ein- bis fünfmal vorkommen kann.

```
<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>
```

Die übergeordnete Datenstruktur ist identisch mit früheren Beispielen:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

Die zweite Datenstruktur enthält jedoch die folgenden Elemente:

```
01 DFHWS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Eine Struktur auf dritter Ebene enthält die folgenden Elemente:

```
01 DFHWS-component2
02 component2 PIC X(8)
```

Die Anzahl von Vorkommen des äußersten Elements `<component1>` ist in `component1-num` angegeben.

Der in `component1-cont` benannte Container enthält ein Array mit dieser Anzahl von Instanzen der zweiten Datenstruktur `DFHWS-component1`.

Jede Instanz von `component2-cont` benennt einen anderen Container, der jeweils die Datenstruktur enthält, die von der Struktur auf dritter Ebene, `DFHWS-component2`, zugeordnet wird.

Um diese Struktur zu veranschaulichen, ziehen Sie das XML-Fragment in Betracht, das dem folgenden Beispiel entspricht:

```
<component1><component2>
  string1
</component2></component1>
```

```

<component1><component2>
string2
</component2></component1>
<component1></component1>

```

<component1> kommt dreimal vor. Die ersten beiden Vorkommen enthalten jeweils eine Instanz von <component2>, die dritte Instanz jedoch nicht.

In der übergeordneten Datenstruktur enthält component1-num den Wert 3. Der in component1-cont benannte Container enthält drei Instanzen von DFHWS-component1 :

1. In der ersten Instanz hat component2-num den Wert 1 und der in component2-cont benannte Container enthält *string1*.
2. In der zweiten Instanz hat component2-num den Wert 1 und der in component2-cont benannte Container enthält *string2*.
3. In der dritten Instanz hat component2-num den Wert 0 und der Inhalt von component2-cont ist nicht definiert.

In dieser Instanz wird die gesamte Datenstruktur von vier Containern dargestellt:

- Stammdatenstruktur in Container DFHWS-DATA
- In component1-cont benannter Container
- In den ersten beiden Instanzen von component2-cont benannte zwei Container

Optionale Strukturen und 'xsd:choice'

DFHWS2LS und DFHSC2LS unterstützen die Verwendung von minOccurs und maxOccurs in den Elementen <xsd:sequence>, <xsd:choice> und <xsd:all> nur auf Zuordnungsebene 2.1 und höher, wobei die Attribute minOccurs und maxOccurs auf minOccurs="0" und maxOccurs="1" gesetzt sind.

Die Assistenten generieren Zuordnungen, die diese Elemente behandeln, als seien die in ihnen enthaltenen untergeordneten Elemente optional. Wenn Sie eine Anwendung mit diesen Elementen implementieren, stellen Sie sicher, dass keine ungültigen Kombinationen von Optionen von der Anwendung generiert werden. Jedes dieser Elemente hat ein eigenes Feld count in der generierten Sprachstruktur; diese Felder müssen entweder alle auf "0" oder alle auf "1" gesetzt sein. Jede andere Kombination von Werten ist ungültig, außer mit <xsd:choice>-Elementen.

<xsd:choice>-Elemente geben an, dass nur eine der Optionen in dem Element verwendet werden kann. Dies wird auf allen Zuordnungsebenen unterstützt. Die Assistenten verarbeiten alle diese Optionen in einem <xsd:choice>-Element als ob es sich um ein <xsd:sequence>-Element mit minOccurs="0" und maxOccurs="1" handelt. Stellen Sie beim Implementieren einer Anwendung mithilfe des <xsd:choice>-Elements sicher, dass keine ungültigen Kombinationen von Optionen von der Anwendung generiert werden. Jedes der Elemente hat ein eigenes Feld count in der generierten Sprachstruktur. Von diesen muss genau eines den Wert 1 haben, die anderen müssen alle den Wert 0 haben. Alle anderen Kombinationen von Werten sind ungültig, außer wenn das <xsd:choice>-Element selbst optional ist. In diesem Fall können alle Felder den Wert 0 haben.

Unterstützung für Werte variabler Länge und Leerzeichen

Sie können anpassen, wie Werte variabler Länge und Leerzeichen verarbeitet werden, indem Sie Einstellungen in den CICS-Assistenten nutzen und indem Sie Facetten direkt im XML-Schema hinzufügen.

Typischerweise ordnen der CICS-XML-Assistent und der CICS-Web-Service-Assistent Datenzeichenfolgen Zeichenarrays fester Länge zu. Diese Arrays müssen mit

Leerzeichen oder Nullen gefüllt werden. Das Zuordnen von Werten variabler Länge zu Datenarrays fester Länge kann ineffizient sein und sinnlos Speicherplatz belegen. Wenn die Länge Ihrer Daten variabel ist, wird empfohlen, die Art und Weise anzupassen, in der diese Zuordnungen gehandhabt werden.

Bei einer WSDL-Dokumentkonvertierung aus einer Sprachstruktur in ein XML-Schema oder WSDL-Dokument wird empfohlen, die Facetten `whiteSpace` und `maxLength` in Ihrem XML-Schema anzupassen und den Parameter **CHAR-VARYING-LIMIT** in den Assistenten anzugeben.

Bei einer Konvertierung aus einem XML-Schema oder WSDL-Dokument in eine Sprachstruktur wird empfohlen, einen entsprechenden Wert für den Parameter **CHAR-VARYING** in den Assistenten festzulegen.

Anmerkung: Nullzeichen ('x00') sind in XML-Dokumenten nicht gültig. Alle Nullzeichen aus Anwendungsdaten, die von CICS geparkt werden, werden als Ende einer Zeichenfolge interpretiert und der Wert wird abgeschnitten. Wenn CICS Anwendungsdaten generiert, tut es dies entsprechend des Werts des Parameters **CHAR-VARYING**. Beispiel: Wenn die Option **CHAR-VARYING=NULL** angegeben ist, enden Zeichenfolgen variabler Länge, die von CICS generiert werden, auf ein Nullzeichen.

Werte variabler Länge aus XML zu Sprachstrukturen zuordnen

Verwenden Sie Facetten im XML-Schema oder geben Sie bestimmte Parameter in den CICS-Assistenten an, um anzupassen, wie Zuordnungen zwischen Ihrem XML-Schema oder WSDL-Dokument und der Sprachstruktur gehandhabt werden.

XML-Datentypen können mithilfe von Facetten eingeschränkt werden. Verwenden Sie die Längenfacetten (`length`, `maxLength` und `minLength`) und die Facette `whiteSpace`, um anzupassen, wie Daten variabler Länge in Ihrer XML gehandhabt werden.

length Gibt an, dass die Daten eine feste Länge haben.

maxLength

Gibt die maximale Länge für den Datentyp an. Wenn dieser Wert für einen zeichenfolgebasierten Datentyp nicht festgelegt wird, gibt es keine maximale Länge.

minLength

Gibt die minimale Länge für den Datentyp an. Wenn dieser Wert für einen zeichenfolgebasierten Datentyp nicht festgelegt wird, ist die minimale Länge '0'.

whiteSpace

Gibt an, wie Leerraum um einen Datenwert herum gehandhabt wird. Als Leerraum werden Leerzeichen, Tabulatoren und Zeilenumbrüche bezeichnet. Die Facette `whiteSpace` kann auf `preserve`, `replace` oder `collapse` festgelegt werden:

- Der Wert `preserve` behält alle Leerräume in dem Datenwert bei.
- Bei dem Wert `replace` werden alle Tabulatoren oder Zeilenumbrüche durch die entsprechende Anzahl von Leerzeichen ersetzt.
- Bei dem Wert `collapse` werden führende, nachfolgende und eingebettete Leerräume entfernt und alle Tabulatoren, Zeilenumbrüche und aufeinanderfolgenden Leerzeichen durch einzelne Leerzeichen ersetzt.

Wenn die Facette `whiteSpace` nicht festgelegt ist, wird der Leerraum beibehalten.

Weitere Informationen zu XML-Schemafacetten finden Sie unter XML Schema Part 2: Datatypes Second Edition.

Die folgenden Parameter in den CICS-Assistenten DFHSC2LS und DFHWS2LS können verwendet werden, um die Art und Weise zu ändern, in der Daten variabler Länge aus dem XML-Schema der Sprachstruktur zugeordnet werden. Diese Parameter sind auf Zuordnungsebene 1.2 oder höher verfügbar.

DEFAULT-CHAR-MAXLENGTH

Gibt die Standardarraylänge von Zeichendaten in Zeichen für Zuordnungen an, in deren XML-Schema oder WSDL-Dokument keine Länge eingeschlossen ist. Der Wert dieses Parameters kann eine positive ganze Zahl im Bereich von 1 bis 2.147.483.647 sein.

Es wird jedoch empfohlen, die maximale Zeichenlänge, die DFHSC2LS oder DFHWS2LS verwenden sollen, direkt in Ihrem XML-Schema oder WSDL-Dokument mithilfe der Facette `maxLength` anzugeben. Indem Sie die maximale Länge direkt in dem XML-Schema oder WSDL-Dokument angeben, vermeiden Sie Probleme, die auftreten können, wenn eine globale Standardeinstellung auf alle zeichenfolgebasierten Datentypen angewendet wird.

CHAR-VARYING-LIMIT

Gibt die maximale Größe von Zeichendaten variabler Länge an, die der Sprachstruktur zugeordnet werden. Wenn die Zeichendaten größer sind als der in diesem Parameter angegebene Wert, werden sie einem Container zugeordnet, und der Containername wird in der generierten Sprachstruktur verwendet. Der Wert kann zwischen 0 und dem Standardwert 32.767 Bytes liegen.

CHAR-VARYING

Gibt an, wie die Zeichendaten variabler Länge zugeordnet werden. Wenn Sie diesen Parameter nicht angeben, hängt die Standardzuordnung von der angegebenen Sprache ab. Sie haben folgende Optionen:

- **CHAR-VARYING=NO** gibt an, dass Zeichendaten variabler Länge als Zeichenfolgen mit fester Länge zugeordnet werden.
- **CHAR-VARYING=NULL** gibt an, dass Zeichendaten variabler Länge auf null endenden Zeichenfolgen zugeordnet werden.
- **CHAR-VARYING=YES** gibt an, dass Zeichendaten variabler Länge in PL/I einem CHAR VARYING-Datentyp zugeordnet werden. In den COBOL-, C- und C++-Sprachen werden Zeichendaten variabler Länge einer äquivalenten Darstellung zugeordnet, die zwei verwandte Elemente umfasst: die Datenlänge und die Daten.

CHAR-VARYING=YES führt in der Regel zu einer optimalen Leistung.

Werte variabler Länge aus Sprachstrukturen zu XML zuordnen

Sie können anpassen, wie Zuordnungen zwischen Ihrer Sprachstruktur und dem XML-Schema oder WSDL-Dokument gehandhabt werden. Legen Sie den Parameter **CHAR-VARYING** in DFHLS2SC oder DFHLS2WS auf **COLLAPSE** oder **NULL** fest, um zu ändern, wie diese Zeichenarrays generiert werden.

Das Festlegen der Option **CHAR-VARYING=NULL** weist CICS an, beim Generieren von XML ein Nullzeichen am Ende jedes Zeichenarrays hinzuzufügen.

Das Festlegen der Option **CHAR-VARYING-COLLAPSE** weist CICS an, beim Generieren von XML alle abschließenden Leerzeichen von Zeichenarrays automatisch zu entfernen. Diese Option ist nur auf Zuordnungsebene 2.1 oder höher verfügbar und **CHAR-VARYING-COLLAPSE** ist der Standardwert auf Zuordnungsebene 2.1 oder höher für alle Sprachen außer C und C++. Wenn die XML geparkt wird, werden alle führenden, abschließenden und eingebetteten Leerräume entfernt.

Unterstützung für UTF-16 in Anwendungsdaten

CICS-Web-Services unterstützen die Konvertierung von Anwendungsdaten, die in UTF-16 codiert sind, in XML oder JSON und umgekehrt. Verwenden Sie UTF-16, wenn Sie Daten in mehreren Sprachen speichern und verarbeiten müssen.

SOAP- und JSON-WebServices von CICS unterstützen die Konvertierung von Anwendungsdaten, die in UTF-16 codiert sind, in XML oder JSON und umgekehrt. Unicode ist ein Schema für Codeumsetzung mit variabler Breite, das es Systemen ermöglicht, Daten effizient zu verarbeiten.

UTF-16 ist eine Unicode-Codierung mit variabler Breite, bei der jedes Zeichen durch zwei oder vier Bytes dargestellt wird. CICS-Web-Services unterstützen CCSID 1200 für Anwendungsdaten im Format UTF-16 BE (Big Endian) mit IBM Private Use Area. Dieses Verhalten ist konsistent mit der UTF-16-Unterstützung in allen unterstützten Sprachen.

UTF-16 wird auf Zuordnungsebene 4.0 und höher unterstützt. Sie können mithilfe von Zuordnungseinstellungen in den Assistenten anpassen, wie Anwendungsdaten konvertiert werden. Weitere Informationen zu XML-Zuordnungsebenen finden Sie unter Mapping levels for the CICS assistants. Weitere Informationen zu JSON-Zuordnungsebenen finden Sie unter Mapping levels for the CICS JSON assistants.

Anmerkung: UTF-16 erfordert mehr Verarbeitungszeit und ist weniger speichereffizient als EBCDIC-Codierungen. Hinzu kommt, dass gemischte Codierungstypen zu zusätzlicher Laufzeitverarbeitung führen.

UTF-16 aus einem XML- oder JSON-Schema Sprachstrukturen zuordnen

Die Unterstützung für UTF-16 hängt davon ab, wie Sie den Web-Service erstellen. Die Zuordnung von XML- oder JSON-Schemas zu Sprachstrukturen, auch bekannt als Top-down-Zuordnung, weist die folgenden Merkmale auf. Wenn UTF-16 aktiviert ist, werden alle Textfelder UTF-16-Feldern zugeordnet, während numerische Anzeigedatentypen in COBOL als EBCDIC zugeordnet werden. Für die Verwendung von UTF-16 setzen Sie den CCSID-Parameter von DFHJS2LS, DFHSC2LS oder DFHWS2LS auf 1200.

Beispiel: Wenn das folgende XML-Schemafragment in der WSDL vorhanden ist:

```
<xsd:element name="myString" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="20"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Dann generiert der DFHWS2LS-Assistent das folgende Feld in einer COBOL-Sprachstruktur:

```
myString PIC N(
20
) USAGE NATIONAL
```

Mit dem Parameter CHAR-MULTIPLIER der Web-Service-Assistenten kann die Länge eines Felds angegeben werden, das die Assistenten generieren.

CHAR-MULTIPLIER

Wenn Sie UTF-16 verwenden, sind die einzig gültigen Werte für den Parameter **CHAR-MULTIPLIER** 2 oder 4 , wobei 2 der Standardwert ist.

CHAR-MULTIPLIER = 2 , wobei das Schema die Zeichenfolge maxlength x beschreibt, generiert PIC N(x) . Das Festlegen von **CHAR-MULTIPLIER** = 2 schließt nicht die Verwendung von Ersatzzeichenpaaren in einer UTF-16-Zeichenfolge aus, wirkt sich aber auf die Anzahl von Zeichen aus, die in das Feld passen.

CHAR-MULTIPLIER = 4 generiert PIC N(2x) . Wenn **CHAR-MULTIPLIER** = 4 , wird der Wert zur Laufzeit aufgefüllt, wenn die Zeichenfolge Zeichen umfasst, die in einer einzelnen Codierungseinheit ausgedrückt werden können.

UTF-16 aus Sprachstrukturen XML- oder JSON-Schemas zuordnen

Die Zuordnung aus einer Sprachstruktur zu einem XML- oder JSON-Schema, auch als Bottom-up-Zuordnung bezeichnet, wird auf andere Weise verwaltet als die Top-down-Zuordnung. Wenn eine UTF-16-Zeichenfolge in der Sprachstruktur deklariert wird, werden die Daten von CICS als UTF-16-codiert interpretiert, andernfalls wird angenommen, dass die Daten EBCDIC-codiert sind. Der CCSID-Parameter für DFHLS2JS, DFHLS2SC oder DFHLS2WS gibt die Codierung eines beliebigen EBCDIC-Texts innerhalb der Anwendungsdaten an. Er darf nicht UTF-16 angeben.

Die Datentypen, die als UTF-16-Zeichen interpretiert werden, sind folgende: PIC N (n) in COBOL, WIDECHAR(n) in PL/I und char16_t[n] in C und C++.

Der Parameter CHAR-USAGE des Web-Service-Assistenten kann verwendet werden, um Datentypen anzugeben.

CHAR-USAGE

In COBOL kann der nationale Datentyp, PIC N, für UTF-16- oder DBCS-Daten verwendet werden. Diese Einstellung wird durch die Compileroption NSYMBOL gesteuert. Sie müssen den Parameter **CHAR-USAGE** im Assistenten auf denselben Wert setzen wie die Compileroption NSYMBOL, um sicherzustellen, dass die Daten richtig verarbeitet werden. Dieser Wert wird in der Regel auf CHAR-USAGE=NATIONAL gesetzt, wenn Sie UTF-16 verwenden.

Wenn Sie nationale Datentypen, die UTF-16- und DBCS-Daten in demselben Copybook enthalten, mischen möchten, können Sie die Qualifikationsmerkmale USAGE NATIONAL oder USAGE DISPLAY-1 in einzelnen Feldern verwenden.

Anmerkung: DFHLS2WS, DFHLS2SC und DFHLS2JS bieten keine Unterstützung für die COBOL-Klausel GROUP USAGE NATIONAL.

Web-Service-Provider mit dem Web-Service-Assistenten erstellen

Sie können eine Service-Provider-Anwendung aus einer Web-Service-Beschreibung, die mit WSDL 1.1 oder WSDL 2.0 konform ist, oder aus einer Datenstruktur einer höheren Programmiersprache erstellen. Der CICS-Web-Service-Assistent hilft Ihnen bei der Implementierung Ihrer CICS-Anwendungen in einer Service-Provider-Einstellung.

About this task

Wenn Sie den Assistenten verwenden, um eine CICS-Anwendung als Service-Provider zu implementieren, haben Sie zwei Optionen:

- Beginnen Sie mit einer Web-Service-Beschreibung und verwenden Sie den Assistenten, um die Sprachdatenstrukturen zu generieren.

Verwenden Sie diese Option, wenn Sie einen Service-Provider implementieren, der mit einer vorhandenen Web-Service-Beschreibung konform ist.

- Beginnen Sie mit den Sprachdatenstrukturen und verwenden Sie den Assistenten, um die Web-Service-Beschreibung zu generieren.

Verwenden Sie diese Option, wenn Sie ein vorhandenes Programm als Web-Service zugänglich machen und bereit sind, Aspekte der Programmschnittstellen in der Web-Service-Beschreibung und in den SOAP-Nachrichten verfügbar zu machen.

Sie können die Web-Service-Beschreibung, die Ihrem Service-Provider zugeordnet ist, unter Verwendung eines URI verfügbar machen. Dieser URI hat denselben Pfad wie der URI, der dem WEBSERVICE-Element zugeordnet ist, wobei das Suffix `?wsdl` angehängt ist. Dadurch können Requester in Ihrem Unternehmen oder außerhalb davon die WSDL-Dateien, die Ihren Service-Providern zugeordnet sind, erkennen.

Service-Provider-Anwendung aus einer Web-Service-Beschreibung erstellen

Mit dem CICS-Web-Service-Assistenten können Sie eine Service-Provider-Anwendung aus einer Web-Service-Beschreibung erstellen, die mit WSDL 1.1 oder WSDL 2.0 konform ist.

Before you begin

Bevor Sie eine Service-Provider-Anwendung erstellen können, müssen die folgenden Bedingungen erfüllt sein:

- Ihre Web-Service-Beschreibung muss sich in einer UNIX-Datei unter z/OS befinden und Sie müssen eine geeignete Pipeline im Providermodus in der CICS-Region erstellen.
- Sie müssen in OMVS die Benutzer-ID definieren, unter der DFHWS2LS ausgeführt wird.
- Die Benutzer-ID muss über Leseberechtigungen für z/OS UNIX- und PDS-Bibliotheken und über Schreibberechtigungen für die Verzeichnisse verfügen, die in den Parametern **LOGFILE**, **WSBIND** und **WSDL** angegeben sind.
- Sie müssen der Benutzer-ID ausreichend Speicherplatz zuordnen, damit Java ausgeführt werden kann. Sie können alle unterstützten Versionen von Java verwenden. Standardmäßig verwendet DFHWS2LS die Java-Version, die im Parameter **JAVADIR** angegeben ist.

About this task

Mit dem Web-Service-Assistenten können Sie Sprachstrukturen aus Ihrer WSDL für die Service-Provider-Anwendung erstellen. Sie können auch ein WSDL-Dokument verwenden, das auf einem IBM webSphere Service Registry and Repository-Server (WSRR) gespeichert ist.

Procedure

1. Verwenden Sie das DFHWS2LS-Stapelverarbeitungsprogramm, um eine Web-Service-Bindungsdatei und mindestens eine Sprachdatenstruktur zu generieren. DFHWS2LS enthält eine große Auswahl optionaler Parameter, die Ihnen die Flexibilität bieten, die Bindungsdatei und die Sprachstrukturen zu erstellen, die für Ihre Anwendung erforderlich sind. Ziehen Sie diese Optionen in Betracht, wenn Sie eine vorhandene Anwendung für Web-Services aktivieren:
 - a. **Welchen Mechanismus verwendet CICS, um Daten an das Service-Provider-Anwendungsprogramm zu übergeben?**
 - Sie können Kanäle verwenden und die Daten in Containern übergeben oder Sie können einen Kommunikationsbereich verwenden. Kanäle und Container werden empfohlen. Geben Sie diese mit dem Parameter **PGMINT** an.
 - b. **Welche Sprache soll generiert werden?**
 - DFHWS2LS kann COBOL-, C/C++- oder PL/I-Sprachdatenstrukturen generieren. Geben Sie die Sprache mithilfe des Parameters **LANG** an.
 - c. **Welche Zuordnungsebene soll verwendet werden?**
 - Je höher die Zuordnungsebene, umso mehr Steuerungsmöglichkeiten und Unterstützung stehen Ihnen für die Verarbeitung von Zeichen und Binärdaten zur Laufzeit zur Verfügung. Manche optionalen Parameter sind nur für die höheren Zuordnungsebenen verfügbar. Es wird empfohlen, die höchste verfügbare Zuordnungsebene zu verwenden. Geben Sie die Zuordnungsebene mit dem Parameter **MAPPING-LEVEL** an.
 - d. **Welcher URI soll vom Web-Service-Requester verwendet werden?**
 - Geben Sie einen relativen URI mithilfe des Parameters **URI** an, z. B. URI=/my/test/webservice. Der Wert wird von CICS beim Erstellen der URI-MAP-Ressource verwendet.
 - e. **Unter welcher Transaktion und Benutzer-ID werden die Web-Service-Anforderung und -Antwort ausgeführt?**
 - Sie können eine Aliastransaktion verwenden, um die Anwendung auszuführen und eine Antwort an den Service-Requester zu erstellen. Die Aliastransaktion wird unter der Benutzer-ID angehängt.
 - Geben Sie dies mit den Parametern **TRANSACTION** und **USERID** an. Diese Werte werden beim Erstellen der URIMAP-Ressource verwendet. Wenn Sie keine bestimmte Transaktion verwenden möchten, geben Sie diese Parameter nicht an.
 - f. **Wo wird das WSDL-Dokument gespeichert?**
 - Wenn Sie ein WSDL-Dokument von einem WSRR-Server statt von dem lokalen Dateisystem abrufen möchten, müssen Sie bestimmte Parameter in DFHWS2LS angeben.
 - Sie müssen mindestens den Parameter **WSRR-SERVER** mit der Position des WSRR-Servers und den Parameter **WSRR-NAME** mit dem Namen des WSDL-Dokuments, das Sie von WSRR abrufen möchten, angeben.

- Informationen zu anderen Parametern, die Sie möglicherweise angeben möchten, wenn Sie WSRR verwenden, finden Sie unter „DFHWS2LS: Konvertierung von WSDL in eine höhere Programmiersprache“ auf Seite 539.
- g. **Wenn Sie beabsichtigen, Ihr WSDL-Dokument von einem WSRR-Server abzurufen, möchten Sie dies über eine sichere Verbindung tun?**
- Sie können SSL-Verschlüsselung (Secure Socket Layer) verwenden, indem Sie die entsprechenden Parameter so festlegen, dass sie sicher mit WSRR interagieren. Ein Beispiel finden Sie unter Example of how to use SSL with the web services assistant and WSRR.
 - Wenn Sie DFHWS2LS übergeben, generiert CICS die Web-Service-Bindungsdatei und platziert sie an der im Parameter **WSBIND** angegebenen Position. Die Sprachstrukturen werden in der partitionierten Datei hinzugefügt, die Sie mit dem Parameter **PDSLIB** angegeben haben.
2. Kopieren Sie die generierte Web-Service-Bindungsdatei in das Abholverzeichnis (pickup) der PIPELINE-Ressource im Providermodus, die Sie für Ihre Web-Service-Anwendung nutzen möchten. Sie müssen die Bindungsdatei im Binärmodus kopieren.
 3. Optional: Kopieren Sie die Web-Service-Beschreibung oder die Archivdatei, die eine oder mehrere Web-Service-Beschreibungen enthält, in dasselbe Verzeichnis wie die Web-Service-Bindungsdatei. Die Archivdatei muss eine ZIP-Datei sein und der Dateiname muss mit dem Namen der WSDL-Datei übereinstimmen. Anhand dieser Kopie können Sie sich mit WSDL vertraut machen.
 4. Schreiben Sie ein Service-Provider-Anwendungsprogramm, das die Schnittstelle zu den generierten Sprachstrukturen bildet, und implementieren Sie die erforderliche Geschäftslogik.
 5. Erstellen Sie die WEBSERVICE-Ressource und zwei URIMAP-Ressourcen.
 - Die WEBSERVICE-Ressource schließt die Web-Service-Bindungsdatei in CICS ein und wird zur Laufzeit verwendet.
 - Die erste URIMAP-Ressource stellt CICS die Informationen für die Zuordnung der WEBSERVICE-Ressource zu einem bestimmten URI bereit.
 - Die zweite URIMAP-Ressource stellt CICS die Informationen für die Zuordnung der WSDL-Archivdatei zu einem bestimmten URI bereit.
 - Dieser URI hat denselben Pfad wie der URI, der dem WEBSERVICE-Element zugeordnet ist, wobei das Suffix ?wsdl angehängt ist.
 - Diese URIMAP-Ressource wird so erstellt, dass externe Requester den URI verwenden können, um die WSDL-Archivdatei oder das WSDL-Dokument zu erkennen.
 - Diese URIMAP-Ressource wird nur erstellt, wenn die Web-Service-Beschreibung oder die Archivdatei, die eine oder mehrere Web-Service-Beschreibungen enthält, in dasselbe Verzeichnis wie die Web-Service-Bindungsdatei kopiert wurde.
 - Wenn das Abholverzeichnis eine WSDL-Archivdatei und ein WSDL-Dokument enthält, gibt der URI nur die WSDL in der Archivdatei zurück.
 - Diese Funktion steht nur für Web-Services zur Verfügung, die unter Verwendung der Pipelinescanoperation installiert wurden.

Sie können die Ressourcen auf folgende Arten erstellen:

- a. Indem Sie den Befehl **PIPELINE SCAN** verwenden, um die WEBSERVICE-Ressource und URIMAP-Ressourcen dynamisch zu erstellen.
- b. Indem Sie die Ressourcen selbst definieren. Wenn Sie CICS Explorer verwenden, um eine WEBSERVICE-Ressource in einem CICS-Bundle zu defi-

nieren, können Sie eine Web-Service-Bindungsdatei und ein WSDL-Dokument bzw. eine WSDL-Archivdatei importieren und in dieses Bundle einschließen. Anschließend können Sie die URIMAP-Definitionen zur Unterstützung des Web-Service generieren, und diese in ein Bundle packen. Weitere Hilfe zur Verwendung von CICS Explorer zum Erstellen und Bearbeiten von Ressourcen in CICS-Bundles finden Sie unter Working with bundles in the CICS Explorer product documentation.

Results

Wenn es beim Übergeben von DFHWS2LS zu Fehlern kommt oder wenn die Ressourcen nicht korrekt installiert werden können, finden Sie weitere Informationen unter Diagnosing deployment errors.

Service-Provider-Anwendung aus einer Datenstruktur erstellen

Mit dem Assistenten für CICS-Web-Services können Sie eine Service-Provider-Anwendung aus einer Datenstruktur einer höheren Programmiersprache erstellen.

Before you begin

Bevor Sie eine Service-Provider-Anwendung erstellen, müssen die folgenden Vorbedingungen erfüllt sein:

- Ihre Datenstrukturen einer höheren Programmiersprache müssen die folgenden Kriterien erfüllen:
 - Die Datenstrukturen müssen getrennt vom Quellenprogramm definiert werden, z. B. in einem COBOL-Copybook.
 - Wenn Ihr PL/I- oder COBOL-Anwendungsprogramm für die Eingabe und Ausgabe verschiedene Datenstrukturen verwendet, müssen diese in zwei unterschiedlichen Members in einer partitionierten Datei definiert sein. Wenn dieselbe Struktur für Eingabe und Ausgabe verwendet wird, muss sie in einem einzigen Member definiert sein.
Für C und C++ können sich Ihre Datenstrukturen in demselben Member in einer partitionierten Datei befinden.
- Welche Datenstrukturen Sie verarbeiten, hängt davon ab, ob Sie ein Wrapperprogramm verwenden.
 - Wenn Sie ein Wrapperprogramm verwenden, ist das Copybook die Schnittstelle zum Wrapperprogramm.
 - Wenn Sie kein Wrapperprogramm verwenden, ist das Copybook die Schnittstelle zur Geschäftslogik.
- Die Sprachstrukturen müssen in einer partitionierten Datei verfügbar sein und Sie müssen eine geeignete PIPELINE-Ressource in der CICS-Region erstellen:
 - Sie müssen in OMVS die Benutzer-ID definieren, unter der DFHLS2WS ausgeführt wird.
 - Die Benutzer-ID muss über Leseberechtigungen für z/OS UNIX- und PDS-Bibliotheken und über Schreibberechtigungen für die Verzeichnisse verfügen, die in den Parametern **LOGFILE**, **WSBIND** und **WSDL** angegeben sind.
 - Der Benutzer-ID muss ausreichend Speicher zugeordnet sein, um Java auszuführen. Sie können alle unterstützten Versionen von Java verwenden. Standardmäßig verwendet DFHLS2WS die Java-Version, die im Parameter **JAVADIR** angegeben ist.

Procedure

Führen Sie die folgenden Schritte aus, um eine Service-Provider-Anwendung aus einer Datenstruktur zu erstellen:

1. Wenn die Schnittstelle der Service-Provider-Anwendung Kanäle und viele Container verwendet, erstellen Sie ein Kanalbeschreibungsdokument, das die Schnittstelle in XML beschreibt. Sie müssen das Kanalbeschreibungsdokument in einem geeigneten Verzeichnis unter z/OS UNIX hinzufügen. CICS verwendet dieses Dokument, um eine SOAP-Nachricht aus den Containern in einem Kanal zu erstellen bzw. zu zerlegen. Alternativ können Sie einen (1) Container in einem Kanal verwenden und kein Kanalbeschreibungsdokument erstellen.
 - Weitere Informationen zum Erstellen eines Kanalbeschreibungsdokuments finden Sie unter „Kanalbeschreibungsdokument erstellen“ auf Seite 623.
2. Verwenden Sie das Stapelverarbeitungsprogramm DFHLS2WS, um eine Web-Service-Bindungsdatei und eine Web-Service-Beschreibung aus der Sprachstruktur zu generieren. DFHLS2WS enthält eine große Auswahl optionaler Parameter, die Ihnen die Flexibilität bieten, die Bindungsdatei und die Sprachstrukturen zu erstellen, die für Ihre Anwendung erforderlich sind. Ziehen Sie diese Optionen in Betracht, wenn Sie einen Web-Service für eine vorhandene Anwendung aktivieren:
 - a. **Welchen Mechanismus verwendet CICS, um Daten an das Service-Provider-Anwendungsprogramm zu übergeben?**
 - Sie können Kanäle verwenden und die Daten in Containern übergeben oder Sie können einen Kommunikationsbereich verwenden. Geben Sie den Mechanismus mithilfe des Parameters **PGMINT** an. Wenn Ihre Anwendungsschnittstelle Kanäle und viele Container verwendet, geben Sie den Parameter **REQUEST-CHANNEL** und optional den Parameter **RESPONSE-CHANNEL** an. Sie können diese Parameter nur bei einer Zuordnungsebene 3.0 oder höher verwenden.
 - b. **Welche Version von Web-Service-Beschreibung (WSDL-Dokument) möchten Sie generieren?**
 - CICS generiert Beschreibungen, die kompatibel sind mit WSDL 1.1- oder WSDL 2.0-Dokumenten. Wenn die Service-Provider-Anwendung Anforderungen unterstützen soll, die mit beiden Versionen von WSDL kompatibel sind, geben Sie Werte für die Parameter **WSDL_1.1** und **WSDL_2.0** an. Stellen Sie sicher, dass die Dateinamen unterschiedlich sind, wenn mehr als ein WSDL-Parameter verwendet wird. Diese Spezifikation erzeugt zwei Web-Service-Beschreibungen und eine Bindungsdatei.
 - c. **Welche Version des SOAP-Protokolls möchten Sie verwenden?**
 - Sie können die Version mit dem Parameter **SOAPVER** angeben. Wir empfehlen Ihnen, den Wert ALL zu verwenden, der Ihnen die Flexibilität bietet, entweder SOAP 1.1 oder SOAP 1.2 als Bindung für die Web-Service-Beschreibung zu verwenden, obwohl Sie den Web-Service in einer Pipeline installieren müssen, die mit dem SOAP 1.2-Nachrichtenhandler konfiguriert ist. Sie können diesen Parameter nur verwenden, wenn **MINIMUM-RUNTIME-LEVEL** 2.0 oder höher ist.
 - d. **Welche Zuordnungsebene soll verwendet werden?**
 - Je höher die Zuordnungsebene, umso mehr Steuerungsmöglichkeiten und Unterstützung stehen Ihnen für die Verarbeitung von Zeichen und Binärdaten zur Laufzeit zur Verfügung. Manche optionalen Parameter sind nur für die höheren Zuordnungsebenen verfügbar. Es wird empfohlen, die höchste verfügbare Zuordnungsebene im Parameter **MAPPING-LEVEL** anzugeben.

- e. **Welcher URI soll vom Web-Service-Requester verwendet werden?**
 - Geben Sie einen absoluten URI mithilfe des Parameters **URI** an, z. B. **URI** = `http://www.example.org:80/my/test/webservice`. Der relative Teil dieser Adresse, `/my/test/webservice`, wird beim Erstellen der URIMAP-Ressource verwendet. Der vollständige URI wird als Element `<soap:address>` in der Web-Service-Beschreibung verwendet. Diese Syntax gilt sowohl für HTTP- als auch für WebSphere MQ-URIs.
- f. **Möchten Sie Ihr WSDL-Dokument in einem IBM WebSphere Service Registry and Repository (WSRR) veröffentlichen?**
 - Wenn Sie Ihr WSDL-Dokument in einem WSRR veröffentlichen möchten, müssen Sie den Parameter **WSRR-SERVER** in DFHLS2WS angeben. Weitere Informationen zu den Parametern, die Sie bei der Verwendung von WSRR angeben können, finden Sie unter „DFHLS2WS: Konvertierung einer höheren Programmiersprache in WSDL“ auf Seite 522.
- g. **Wenn Sie beabsichtigen, Ihr WSDL-Dokument auf einem WSRR-Server zu veröffentlichen, möchten Sie dies über eine sichere Verbindung tun?**
 - Sie können SSL-Verschlüsselung (Secure Socket Layer) verwenden, indem Sie die entsprechenden Parameter so festlegen, dass sie sicher mit WSRR interagieren. Ein Beispiel finden Sie unter Example of how to use SSL with the web services assistant and WSRR.
 - Wenn Sie DFHLS2WS übergeben, generiert CICS die Web-Service-Bindungsdatei und platziert sie an der im Parameter **WSBIND** angegebenen Position. Die generierte Web-Service-Beschreibung wird an der Position hinzugefügt, die Sie im Parameter **WSDL**, **WSDL_1.1** oder **WSDL_2.0** angegeben haben.
 - Wenn Sie die WSRR-Parameter in DFHLS2WS verwendet haben, wird Ihr WSDL-Dokument auf dem WSRR-Server veröffentlicht, den Sie angegeben haben.
3. Überprüfen Sie die generierte Web-Service-Beschreibung und nehmen Sie alle erforderlichen Anpassungen vor. Weitere Informationen finden Sie unter „Generierte Web-Service-Beschreibungsdokumente anpassen“ auf Seite 625.
4. Kopieren Sie die Web-Service-Bindungsdatei in das Abholverzeichnis der Pipeline im Providermodus, die Sie für Ihre Web-Service-Anwendung nutzen möchten. Sie müssen die Web-Service-Bindungsdatei im Binärmodus kopieren.
5. Optional: Kopieren Sie die Web-Service-Beschreibung oder die Archivdatei, die eine oder mehrere Web-Service-Beschreibungen enthält, in dasselbe Verzeichnis wie die Web-Service-Bindungsdatei. Die Archivdatei muss eine ZIP-Datei sein und der Dateiname muss mit dem Namen der WSDL-Datei übereinstimmen. Anhand dieser Kopie können Sie sich mit WSDL vertraut machen.
6. Verwenden Sie den Befehl **PIPELINE SCAN**, um die WEBSERVICE-Ressource und zwei URIMAP-Ressourcen dynamisch zu erstellen. Die WEBSERVICE-Ressource schließt die Web-Service-Bindungsdatei in CICS ein und wird zur Laufzeit verwendet.
 - Die erste URIMAP-Ressource stellt CICS die Informationen für die Zuordnung der WEBSERVICE-Ressource zu einem bestimmten URI bereit.
 - Die zweite URIMAP-Ressource stellt CICS die Informationen für die Zuordnung der WSDL-Archivdatei oder des WSDL-Dokuments zu einem bestimmten URI bereit.
 - Dieser URI hat denselben Pfad wie der URI, der dem WEBSERVICE-Element zugeordnet ist, wobei das Suffix `?wsdl` angehängt ist.
 - Diese URIMAP-Ressource wird so erstellt, dass externe Requester den URI verwenden können, um die WSDL-Archivdatei oder das WSDL-Dokument zu erkennen.

- Diese URIMAP-Ressource wird nur erstellt, wenn die Web-Service-Beschreibung oder die Archivdatei, die eine oder mehrere Web-Service-Beschreibungen enthält, in dasselbe Verzeichnis wie die Web-Service-Bindungsdatei kopiert wurde.
- Wenn das Abholverzeichnis eine WSDL-Archivdatei und ein WSDL-Dokument enthält, gibt der URI nur die WSDL in der Archivdatei zurück.
- Diese Funktion steht nur für Web-Services zur Verfügung, die unter Verwendung der Pipelinescanoperation installiert wurden.

Alternativ können Sie die Ressourcen selbst definieren. Dies wird jedoch nicht empfohlen.

Results

Wenn Sie die CICS-Ressourcen erfolgreich erstellt haben, ist die Erstellung Ihrer Service-Provider-Anwendung abgeschlossen.

Wenn es beim Übergeben von DFHLS2WS zu Fehlern kommt oder wenn die Ressourcen nicht korrekt installiert werden können, finden Sie weitere Informationen unter [Diagnosing deployment errors](#).

What to do next

Machen Sie die Web-Service-Beschreibung für jeden verfügbar, der einen Web-Service-Requester entwickeln muss, der auf Ihren Service zugreifen wird.

Kanalbeschreibungsdocument erstellen

Erstellen Sie ein Kanalbeschreibungsdocument, wenn Ihre Service-Provider-Anwendung eine Kanalschnittstelle mit vielen Containern verwendet.

About this task

Verwenden Sie einen XML-Editor, um das Kanalbeschreibungsdocument zu erstellen. Das Schema für die Kanalbeschreibung wird `channel.xsd` genannt und befindet sich im Verzeichnis `/usr/lpp/cicsts/cicsts52/schemas/channel` (wobei `/usr/lpp/cicsts/cicsts52` das Standardinstallationsverzeichnis für CICS-Dateien ist).

Procedure

1. Erstellen Sie ein XML-Dokument mit einem `<channel>`-Element und dem Namensbereich des CICS-Kanals.

```
<channel name="myChannel"
xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```

2. Fügen Sie ein `<container>`-Element für jeden Container hinzu, den die Anwendungsprogrammierschnittstelle in dem Kanal verwendet. Sie müssen die einzelnen Container mit Namens-, Typ- und Verwendungsattributen beschreiben. Im folgenden Beispiel sind sechs Container mit verschiedenen Attributwerten dargestellt:

```
<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>

```

Das Strukturelement gibt an, dass der Inhalt in einer Sprachstruktur definiert ist, die sich in einem Member einer partitionierten Datei befindet.

3. Speichern Sie das XML-Dokument unter z/OS UNIX.

Kanalschema

Das Kanalbeschreibungsdokument muss dem folgenden Schema entsprechen:

```

<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. Dieses Element stellt einen CICS-Kanal dar.
2. Dieses Element stellt einen CICS-Container innerhalb des Kanals dar.
3. Eine Struktur kann nur mit Bitmodus-Containern verwendet werden. Das Attribut 'location' gibt die Position einer Datei an, die den Inhalt des Containers abbildet. Das Attribut 'structure' kann in C und C++ verwendet werden, um den Namen der Struktur anzugeben.

What to do next

Führen Sie DFHLS2WS aus, um die Zuordnungen und das WSDL-Dokument für die Web-Service-Provider-Anwendung zu erstellen. DFHLS2WS fügt die Zuordnungen für den Kanal in der Reihenfolge in das WSDL-Dokument ein, in der die Container im Kanalbeschreibungsdokument angegeben sind.

Generierte Web-Service-Beschreibungsdokumente anpassen

Die Web-Service-Beschreibungsdokumente, die von DFHLS2WS generiert werden, enthalten automatisch generierte Inhalte, die Sie möglicherweise ändern möchten, bevor Sie sie veröffentlichen. Die Anpassung von WSDL-Dokumenten kann dazu führen, dass die Web-Service-Bindungsdatei neu generiert wird und in einigen Fällen ein Wrapperprogramm geschrieben wird.

About this task

Führen Sie diese Schritte aus, um generierte Web-Service-Beschreibungsdokumente anzupassen:

Procedure

1. Wenn Sie Unterstützung für HTTPS zugänglich machen möchten oder mithilfe von WebSphere MQ kommunizieren möchten, verwenden Sie den Parameter **URI** in DFHLS2WS zum Festlegen eines absoluten URI. Wenn Sie den Parameter **URI** nicht verwendet haben, müssen Sie die Elemente `<wsdl:service>` und `<wsdl:binding>` am Ende des WSDL-Dokuments ändern. Die generierte WSDL enthält Kommentare, die Sie beim Vornehmen der Änderungen unterstützen. Das Ändern dieser Elemente führt nicht dazu, dass die Web-Service-Bindungsdatei neu generiert werden muss.
2. Wenn Sie die Netzadresse Ihres Web-Service bereitstellen möchten, verwenden Sie den Parameter **URI** in DFHLS2WS zum Festlegen eines absoluten URI. Wenn Sie den Parameter **URI** nicht verwendet haben, fügen Sie die Details zu `soap:address` im Element `wsdl:service` hinzu.
 - a. Wenn Sie ein HTTP-basiertes Protokoll verwenden, ersetzen Sie *mein_server* durch den TCP/IP-Hostnamen Ihrer CICS-Region und *mein_port* durch die Portnummer der TCPIPService-Ressource.
 - b. Wenn Sie WebSphere MQ als Transportprotokoll verwenden, ersetzen Sie *meine_warteschlange* durch den Namen der entsprechenden Warteschlange.

Sie können diese Änderungen vornehmen, ohne dass Änderungen an der Web-Service-Bindungsdatei erforderlich sind.

Wenn Sie den Portnamen und den Namensbereich ändern, ohne die WSBInd-Datei neu zu generieren, sind die Überwachungsdaten möglicherweise ab Laufzeitebene 2.1 falsch.

3. Überlegen Sie, ob die automatisch generierten Namen in dem WSDL-Dokument für Ihre Zwecke geeignet sind. Sie können diese Werte umbenennen:
 - Der Wert `targetNamespace` des WSDL-Dokuments
 - Der Wert `targetNamespace` des XML-Schemas in dem WSDL-Dokument
 - Der Name von `<wsdl:portType>`
 - Der Name von `<wsdl:operation>`
 - Der Name von `<wsdl:binding>`
 - Der Name von `<wsdl:service>`
 - Die Namen der Felder in den XML-Schemas im WSDL-Dokument.

Diese Werte bilden einen Teil der programmgesteuerten Schnittstelle, für die Sie ein Clientprogramm codieren. Wenn die generierten Namen nicht ausreichend aussagekräftig sind, kann dies die Wartung Ihres Anwendungscodes im Laufe der Zeit schwierig gestalten. Verwenden Sie die Parameter **REQUEST-NAMESPACE** und **RESPONSE-NAMESPACE** in DFHLS2WS, um den Wert `targetNamespace` des XML-Schemas zu ändern, und den Parameter **WSDL-NAMESPACE**, um den Wert `targetNamespace` des WSDL-Dokuments zu ändern.

Wenn Sie einen dieser Werte ändern, müssen Sie DFHWS2LS verwenden, um die Web-Service-Bindungsdatei neu zu generieren. Die erstellten Sprachstrukturen sind nicht identisch mit Ihren vorhandenen Sprachstrukturen, aber sie sind kompatibel mit Ihrer vorhandenen Anwendung, deshalb sind keine Anwendungsänderungen erforderlich. Sie können jedoch die neuen Sprachstrukturen ignorieren und die neue Web-Service-Bindungsdatei mit den ursprünglichen Strukturen verwenden.

4. Überlegen Sie, ob die Kommunikationsbereichsfelder in den XML-Schemas geeignet sind. Möglicherweise erscheint es Ihnen sinnvoll, alle Felder, die für einen Web-Service-Cliententwickler nicht hilfreich sind, zu entfernen:
 - Felder, die nur für Ausgabewerte verwendet werden, können aus dem Schema entfernt werden, das die Eingabedatenstrukturen zuordnet.
 - Füllfelder.
 - Automatisch generierte Annotationen.

Wenn Sie eine dieser Änderungen vornehmen, müssen Sie die Web-Service-Bindungsdatei mithilfe von DFHWS2LS neu generieren. Die neuen Sprachstrukturen, die generiert werden, sind nicht mit den ursprünglichen Sprachstrukturen kompatibel, deshalb müssen Sie ein Wrapperprogramm schreiben, um Daten aus der neuen Darstellung der alten Darstellung zuzuordnen. Dieses Wrapperprogramm muss einen Befehl **EXEC CICS LINK** an das Zielanwendungsprogramm absetzen und die zurückgegebenen Daten dann zuordnen.

Diese Anpassungen sind aufwändig, resultieren aber in den aussagekräftigsten programmgesteuerten Schnittstellen für Ihre Web-Service-Cliententwickler.

5. Wenn Sie das generierte WSDL-Dokument mit DFHWS2LS verarbeiten möchten, um neue Sprachstrukturen zu erstellen, entscheiden Sie, ob die Annotationen in dem WSDL-Dokument beibehalten werden sollen. Die Annotationen überschreiben die normalen Zuordnungsregeln, wenn DFHWS2LS die Sprachstrukturen generiert. Wenn Sie die Zuordnungsregeln überschreiben, stellen Sie sicher, dass die generierten Sprachstrukturen mit der Version kompatibel sind, die von DFHLS2WS verwendet wurde. Wenn Sie die Standardzuordnungsregeln verwenden möchten, um die Sprachstrukturen zu erstellen, entfernen Sie die Annotationen.

Results

Wenn Sie Ihr angepasstes WSDL-Dokument auf einem IBM WebSphere Service Registry and Repository-Server (WSRR) veröffentlichen möchten, müssen Sie dies manuell über die WSRR-Schnittstelle vornehmen. Weitere Informationen zu WSRR finden Sie unter WebSphere Service Registry and Repository.

Example

Ein Beispiel eines WSDL-Dokuments finden Sie unter *An example of the generated WSDL document*.

SOAP-Fehler senden

In einem Service-Provider können Sie die CICS-API verwenden, um einen SOAP-Fehler an einen Web-Service-Requester zu senden. Der Fehler kann von der Service-Provider-Anwendung oder von einem Programm zur Headerverarbeitung in der Pipeline ausgegeben werden.

Before you begin

Um die API zu verwenden, muss die Service-Provider-Anwendung Kanäle und Container verwenden. Wenn die Anwendung Kommunikationsbereiche verwendet, schreiben Sie ein Wrapperprogramm, das Kanäle und Container verwendet, um die SOAP-Fehlernachricht zu erstellen. Sie können die API in einem Programm zur Headerverarbeitung nur verwenden, wenn sie direkt von einem von CICS bereitgestellten SOAP-Nachrichtenhandler aufgerufen wird.

About this task

Unter Umständen möchten Sie einen SOAP-Fehler an den Web-Service-Requester ausgeben, beispielsweise wenn Ihre Anwendungslogik die Anforderung nicht erfüllen kann oder wenn es ein zugrunde liegendes Problem mit der Anforderungsnachricht gibt. Beachten Sie, dass CICS einen SOAP-Fehler nicht als Fehler ausgibt, deshalb findet eine normale Nachrichtenantwort-Pipelineverarbeitung statt und keine Fehlerbehandlung. Wenn Sie Transaktionen rückgängig machen wollen, müssen Sie das Anwendungsprogramm verwenden.

Procedure

1. Verwenden Sie in Ihrem Programm den -Befehl **EXEC CICS SOAPFAULT CREATE** , um einen SOAP-Fehler zu senden.
2. Fügen Sie die Option **CLIENT** oder **SERVER** zum Befehl hinzu. Diese Option gibt an, wo das Problem aufgetreten ist, auf Client- oder auf Serverseite.
 - **CLIENT** gibt an, dass das Problem in der empfangenen Anforderungsnachricht vorliegt.
 - **SERVER** gibt an, dass das Problem auftritt, wenn die Anforderungsnachricht von CICS verarbeitet wird. Das Problem kann in einem Anwendungsprogramm auftreten, beispielsweise kann es nicht möglich sein, die Anforderung zu erfüllen, oder es gibt ein zugrunde liegendes Problem, das während der Pipelineverarbeitung auftritt.
3. Fügen Sie die Option **FAULTSTRING** und die zugehörige Länge in der Option **FAULTSTRLEN** hinzu, um zusammenzufassen, warum der Fehler von dem Service-Provider ausgegeben wurde. Der Inhalt dieser Option ist in XML. Alle von der Anwendung bereitgestellten Daten müssen sich in einem Format befinden, das sich für die direkte Aufnahme in ein XML-Dokument eignet. Die Anwendung muss möglicherweise einige Zeichen als XML-Entitäten angeben. Beispiel: Wenn das Zeichen **<** an einer anderen Stelle als dem Beginn eines XML-Tags verwendet wird, muss die Anwendung es in **<** ändern. Das folgende Beispiel zeigt eine falsche **FAULTSTRING**-Option:

```
dcl msg_faultString char(*) constant('Error: Value A
< Value B');
```

Korrekt ist diese **FAULTSTRING**-Option:

```
dcl msg_faultString char(*) constant('Error: Value A
&lt; Value B');
```

Tip: Um keine XML-Entitäten zu verwenden, können Sie die Daten in ein XML-CDATA-Konstrukt packen. XML-Prozessoren parsen keine Zeichendaten in diesem Konstrukt. Mithilfe dieser Methode können Sie die folgende FAULTSTRING-Option angeben:

```
dc1 msg_faultString char(*)
constant('<![CDATA[Error: Value A < Value B]]>');
```

4. Codieren Sie die Option **DETAIL** und die zugehörige Länge in der Option **DETAILENGTH**, um detailliert anzugeben, warum der Fehler von dem Service-Provider ausgegeben wurde. Der Inhalt dieser Option ist in XML. Für die Optionen **DETAIL** und **FAULTSTRING** gelten dieselben Anweisungen.
5. Optional: Wenn Sie die API aus einem Programm zur Headerverarbeitung aufrufen, definieren Sie das Programm in der Pipeline-Konfigurationsdatei. Das Programm zur Headerverarbeitung wird im Element `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` oder `<cics_soap_1.2_handler_java>` definiert.

Results

Wenn Ihr Programm diesen Befehl ausgibt, erstellt CICS die SOAP-Fehlerantwortnachricht mit der entsprechenden SOAP-Version. Wenn Ihre Service-Provider-Anwendung den Befehl ausgibt, muss sie keine SOAP-Antwort erstellen und sie nicht in den Container **DFHRESPONSE** stellen. Die Pipeline verarbeitet den SOAP-Fehler über die Nachrichtenhandler und sendet die Antwort an den Web-Service-Provider.

Example

Der Befehl **SOAPFAULT CREATE** verfügt über eine Reihe von Optionen, um Ihnen die Flexibilität zu geben, angemessen auf einen Web-Service-Requester zu antworten. Im Folgenden finden Sie ein einfaches Beispiel eines abgeschlossenen Befehls, der einen SOAP-Fehler erstellt, der sowohl für SOAP 1.1 als auch für SOAP 1.2 verwendet werden kann:

```
EXEC CICS SOAPFAULT CREATE CLIENT
DETAIL(
  nachrichtendetail
)
DETAILENGTH(length(
  nachrichtendetail
))
FAULTSTRING(
  nachrichten-fehlerzeichenfolge
)
FAULTSTRLEN(length(
  nachrichten-fehlerzeichenfolge
));
```

Sie können *nachrichtendetail* und *nachrichten-fehlerzeichenfolge* mit den folgenden Werten codieren:

```
dc1 nachrichtendetail char(*)
constant('<ati:ExampleFault xmlns="http://www.example.org/faults"
xmlns:ati="http://www.example.org/faults">Detaillierte
Fehlernachricht</ati:ExampleFault>');
dc1 nachrichten-fehlerzeichenfolge char(*) constant('Something went wrong');
```

Web-Service-Requester mit dem Web-Service-Assistenten erstellen

Sie können eine Service-Requester-Anwendung aus einer Web-Service-Beschreibung erstellen, die mit WSDL 1.1 oder WSDL 2.0 kompatibel ist. Der CICS-Web-Service-Assistent hilft Ihnen bei der Implementierung Ihrer CICS-Anwendungen in einer Service-Requester-Einstellung.

Before you begin

Ihre Web-Service-Beschreibung muss sich in einer Datei unter z/OS UNIX befinden oder sie muss auf einem IBM WebSphere Services Registry and Repository-Server (WSRR) veröffentlicht sein, und eine Pipeline im Requestermodus muss in der CICS-Region installiert sein.

Sie müssen der Benutzer-ID ausreichend Speicherplatz zuordnen, damit die ID Java ausführen kann. Sie können alle unterstützten Versionen von Java verwenden. Standardmäßig verwendet DFHWS2LS die im Parameter **JAVADIR** angegebene Java-Version.

About this task

Wenn Sie den CICS-Web-Service-Assistenten verwenden, um eine CICS-Anwendung als Service-Requester zu implementieren, müssen Sie mit einer Web-Service-Beschreibung beginnen und die Sprachdatenstrukturen daraus generieren.

Procedure

1. Verwenden Sie das DFHWS2LS-Stapelverarbeitungsprogramm, um eine Web-Service-Bindungsdatei und mindestens eine Sprachdatenstruktur zu generieren. Ziehen Sie diese Optionen beim Erstellen einer Service-Requester-Anwendung aus einer Web-Service-Beschreibung in Betracht:
 - Welche Zuordnungsebene soll verwendet werden? Je höher die Zuordnungsebene, umso mehr Steuerungsmöglichkeiten und Unterstützung stehen Ihnen für die Verarbeitung von Zeichen und Binärdaten zur Laufzeit zur Verfügung. Manche optionalen Parameter sind nur für die höheren Zuordnungsebenen verfügbar. Es wird empfohlen, die höchste verfügbare Zuordnungsebene zu verwenden.
 - Welche Codepage soll bei der Umsetzung von Daten zur Laufzeit verwendet werden? Wenn Sie eine bestimmte Codepage für Ihre Anwendung verwenden möchten, die sich von der Codepage für die CICS-Region unterscheidet, verwenden Sie den Parameter **CCSID**.
 - Möchten Sie ein Subset der Operationen unterstützen, die in der Web-Service-Beschreibung deklariert sind? Wenn Sie über eine sehr lange Web-Service-Beschreibung verfügen und Ihre Service-Requester-Anwendung nur eine bestimmte Anzahl von Operationen unterstützen soll, verwenden Sie den Parameter **OPERATION**, um die gewünschten Operationen aufzulisten. Die einzelnen Operationen müssen durch Leerzeichen getrennt werden. Die Groß-/Kleinschreibung muss beachtet werden.
 - Wo wird das WSDL-Dokument gespeichert? Wenn das WSDL-Dokument, das als Eingabe für DFHWS2LS verwendet werden soll, auf einem WSRR-Server gespeichert ist, können Sie es abrufen, indem Sie DFHWS2LS mit bestimmten angegebenen Parametern ausführen. Verwenden Sie den Parameter **WSRR-SERVER**, um die Position des WSRR-Servers anzugeben, und den Parameter **WSRR-NAME**, um den Namen des WSDL-Dokuments anzugeben, das Sie abrufen möchten. Informationen zu anderen Parametern in DFHWS2LS, die

Sie möglicherweise für die Interaktion mit WSRR verwenden möchten, finden Sie unter „DFHWS2LS: Konvertierung von WSDL in eine höhere Programmiersprache“ auf Seite 539.

- Wenn Sie das WSDL-Dokument von einem WSRR-Server abrufen möchten, soll dies über eine sichere Verbindung erfolgen? Sie können SSL-Verschlüsselung (Secure Socket Layer) mit dem Web-Service-Assistenten verwenden, um sicher mit WSRR zu interagieren. Ein Beispiel finden Sie unter Example of how to use SSL with the web services assistant and WSRR.

Geben Sie keine Parameter wie **PROGRAM**, **URI**, **TRANSACTION** und **USERID** an, wenn Sie DFHWS2LS verwenden. Diese Parameter gelten nur für eine Service-Provider-Anwendung und wenn sie angegeben werden, führen sie dazu, dass eine Web-Service-Bindungsdatei im Providermodus erstellt wird. Zusätzlich zu der Web-Service-Bindungsdatei generiert das Programm eine Sprachdatenstruktur.

2. Überprüfen Sie die Protokolldatei, um festzustellen, ob Probleme aufgetreten sind, als DFHWS2LS die Bindungsdatei und die Sprachstrukturen generiert hat. CICS bietet möglicherweise keine Unterstützung für manche Elemente oder Optionen in der Web-Service-Beschreibung. Wenn Warnungen oder Fehlermeldungen ausgegeben werden, lesen Sie die bereitgestellten Anweisungen und ergreifen Sie die entsprechenden Maßnahmen. Möglicherweise müssen Sie das Stapelverarbeitungsprogramm erneut ausführen.
3. Kopieren Sie die Web-Service-Bindungsdatei in das Abholverzeichnis der Pipeline im Requestermodus, die Sie für Ihre Web-Service-Anwendung nutzen möchten.
4. Stellen Sie sicher, dass die PIPELINE-Ressource für Service-Requester-Anwendungen konfiguriert ist. Der Wert des Parameters **MODE** zeigt an, ob die Pipeline Requester- oder Provider-Web-Service-Anwendungen unterstützt.
5. Stellen Sie sicher, dass das korrekte SOAP-Protokoll in der Pipeline für Ihren Web-Service unterstützt wird. Der Parameter **SOAPLEVEL** gibt an, welche Version unterstützt wird. Im Service-Requestermodus muss die Bindung des Web-Service mit der SOAP-Version übereinstimmen, die in der Pipeline unterstützt wird. Sie können einen Web-Service mit einer SOAP 1.1-Bindung nicht in einer Service-Requester-Pipeline installieren, die SOAP 1.2 unterstützt.
6. Stellen Sie sicher, dass das konfigurierte Zeitlimit für die Pipeline für Ihre Service-Requester-Anwendung geeignet ist. Das Zeitlimit wird als Wert des Attributs **RESPWAIT** in der PIPELINE-Ressource angezeigt. Wenn für die Pipeline kein Zeitlimit angegeben ist, wird der Standardwert für den Transport verwendet.
 - Das Standardzeitlimit für HTTP beträgt 10 Sekunden.
 - Das Standardzeitlimit für WebSphere MQ beträgt 60 Sekunden.

Jede Transaktion in der CICS-Region hat ein Dispatcherzeitlimit. Wenn dieser Wert kleiner ist als der Standardwert für eines der Protokolle, wird das Zeitlimit mit dem Dispatcher erreicht.

7. Optional: Kopieren Sie die Web-Service-Beschreibung in dasselbe Abholverzeichnis wie die Web-Service-Bindungsdatei, sodass Sie die Validierung für den Web-Service zur Laufzeit aktivieren können.
8. Erstellen Sie die WEBSERVICE-Ressource. Diese Ressource schließt die Web-Service-Bindungsdatei in CICS ein und wird zur Laufzeit verwendet.

Sie können dies auf folgende Arten tun:

- a. Indem Sie den Befehl **PIPELINE SCAN** verwenden, um die WEBSERVICE-Ressource dynamisch zu erstellen.

- b. Indem Sie die Ressource selbst definieren. Wenn Sie CICS Explorer verwenden, um eine WEBSERVICE-Ressource in einem CICS-Bundle zu definieren, können Sie eine Web-Service-Bindungsdatei und ein WSDL-Dokument bzw. eine WSDL-Archivdatei importieren und in dieses Bundle einschließen. Anschließend können Sie URIMAP-Definitionen zur Unterstützung des Web-Service generieren, und diese in ein Bundle packen. Weitere Hilfe zur Verwendung von CICS Explorer zum Erstellen und Bearbeiten von Ressourcen in CICS-Bundles finden Sie unter Working with bundles in the CICS Explorer product documentation.
9. Schreiben Sie ein Wrapperprogramm, das Sie statt Ihrer Kommunikationslogik verwenden können. Verwenden Sie die in Schritt 1 generierte Sprachdatenstruktur, um Ihr Wrapperprogramm zu schreiben. Verwenden Sie den Befehl **EXEC CICS INVOKE SERVICE** in Ihrem Wrapperprogramm, um mit dem Web-Service zu kommunizieren. Der Befehl enthält die folgenden Optionen:
 - Den URI des Web-Service
 - Die Operation, für die der Web-Service aufgerufen wird

Wenn Sie den Web-Service aufrufen, können Sie eine URIMAP-Ressource angeben, die die Informationen zu dem URI des Web-Service enthält. Sie können diese Informationen direkt im Befehl INVOKE SERVICE angeben, anstatt eine URIMAP-Ressource zu verwenden. Aber wenn Sie eine URIMAP-Ressource verwenden, müssen Sie Ihre Anwendungen nicht erneut kompilieren, wenn sich der URI eines Service-Providers ändert. Mit einer URIMAP-Ressource können Sie auch Verbindungspooling implementieren, wobei CICS die Clientverbindung nach der Verwendung geöffnet hält, sodass sie von der Anwendung für nachfolgende Anforderungen bzw. von einer anderen Anwendung, die denselben Service aufruft, wiederverwendet werden kann. Der Befehl PIPELINE SCAN erstellt keine URIMAP-Ressourcen für einen Service-Requester, daher müssen Sie die URIMAP-Ressource unter Berücksichtigung der Anweisungen unter Creating a URIMAP resource for CICS as a HTTP client selbst definieren.

Results

Wenn Sie die CICS-Ressourcen erfolgreich erstellt haben, ist die Erstellung Ihrer Service-Requester-Anwendung abgeschlossen.

Web-Service mithilfe von Tools erstellen

Statt der JCL des Web-Service-Assistenten können Sie IBM Developer for Z verwenden oder Ihr eigenes Java-Programm schreiben, um die erforderlichen Dateien in CICS zu schreiben.

Procedure

1. Sie haben zwei Möglichkeiten:
 - Verwenden Sie das Tool IBM Developer for Z, um eine Web-Service-Bindungsdatei und die Web-Service-Beschreibung bzw. die Sprachstrukturen zu erstellen. Weitere Informationen zu diesem Tool finden Sie unter IBM Developer for z Systems.
 - Schreiben Sie Ihr eigenes Java-Programm unter Verwendung der bereitgestellten API, um den Web-Service-Assistenten aufzurufen. Diese API wird in dem Web services assistant Javadoc information in the Reference -> Application development-Javadoc beschrieben. Es enthält Kommentare, die die Klassen erläutern, und Beispielcode wird bereitgestellt, um zu zeigen, wie Sie den

Web-Service-Assistenten aufrufen können. Das Javadoc enthält außerdem eine vollständige Liste der erforderlichen JAR-Dateien und ihrer Position in z/OS UNIX.

Sie können Ihr Java-Programm auf der z/OS-, Windows- oder Linux-Plattform ausführen. Wenn Sie das Programm unter Windows oder Linux ausführen, übertragen Sie die generierte Web-Service-Bindungsdatei in ein passendes Abholverzeichnis im Binärmodus mithilfe von FTP oder einem äquivalenten Prozess.

2. Optional: Wenn Sie eine Web-Service-Beschreibung aus einer Sprachstruktur generieren, überprüfen Sie die Datei und nehmen Sie alle erforderlichen Anpassungen vor. Weitere Informationen finden Sie unter „Generierte Web-Service-Beschreibungsdokumente anpassen“ auf Seite 625.
3. Implementieren Sie die generierte Web-Service-Bindungsdatei in einem geeigneten Pipelineabholverzeichnis.
4. Optional: Kopieren Sie die Web-Service-Beschreibung in das Abholverzeichnis der Pipeline, sodass Sie den Web-Service validieren können, um zu überprüfen, ob er wie erwartet funktioniert.
5. Wenn Sie mit einer Web-Service-Beschreibung begonnen haben, schreiben Sie ein Service-Provider- oder Service-Requester-Anwendungsprogramm, das als Schnittstelle mit den generierten Sprachstrukturen dienen soll.
6. Führen Sie den Befehl **PIPELINE SCAN** aus, um die erforderlichen CICS-Ressourcen zu erstellen.

Eigene XML-fähige Web-Service-Anwendungen erstellen

Wenn Sie sich entscheiden, die von CICS bereitgestellten Datenzuordnungen nicht zu verwenden, können Sie stattdessen Ihre eigenen XML-fähigen Datenanwendungen schreiben. Sie haben zwei Möglichkeiten. Sie können entweder den Parameter **XML-ONLY** in DFHWS2LS verwenden oder eine eigene Anwendung schreiben, ohne die Tools zu verwenden. Der Parameter **XML-ONLY** stellt die unkomplizierteste Methode dar, den CICS-Pipelineprozess so zu konfigurieren, dass die XML-Daten zur Verarbeitung an die Anwendung übergeben werden.

About this task

Das Schreiben Ihrer eigenen XML-fähigen Anwendungen beinhaltet das Schreiben von Code zum Parsen und Generieren von XML-Dokumenten. Eine Möglichkeit, Ihre eigene XML-fähige Anwendung zu schreiben, sieht die Verwendung der Anweisungen XML PARSE und XML GENERATE in COBOL vor. Eine andere Möglichkeit, Ihre eigenen XML-fähigen Anwendungen zu schreiben, sieht die Verwendung anderer IBM Tools vor. Sie können beispielsweise das Tool IBM Developer for Z verwenden, um COBOL-XML-Konvertierungsprogramme zu generieren, die von Ihren Anwendungen aufgerufen werden können.

XML-fähige Service-Provider-Anwendungen erstellen

Ihre XML-fähige Service-Provider-Anwendung muss mit den Containern arbeiten, die an sie übergeben werden, und die Datenkonvertierung zwischen der XML und der Programmiersprache steuern.

About this task

Die folgenden Schritte führen Sie durch die Erstellung Ihrer XML-fähigen Anwendung, einschließlich der Entscheidung, ob CICS-Tools verwendet werden sollen.

Procedure

1. Entscheiden Sie, ob eine Web-Service-Bindungsdatei für Ihre XML-fähige Anwendung mithilfe von DFHWS2LS generiert werden soll. Der Vorteil des Generierens einer Web-Service-Bindungsdatei besteht darin, dass Sie CICS-Services wie die Validierung verwenden können, um Ihren Web-Service und die CICS-Überwachung mithilfe von globalen Benutzerexits zu testen.
 - Wenn Sie eine Web-Service-Bindungsdatei generieren möchten, führen Sie DFHWS2LS aus, geben Sie den Parameter **XML-ONLY** an und legen Sie **MINIMUM-RUNTIME-LEVEL** auf 2.1 oder höher fest. Die Web-Service-Bindungsdatei ermöglicht dem Anwendungsprogramm, direkt mit den Inhalten des Containers DFHWS-BODY zu arbeiten. Im Übrigen weist die generierte Bindungsdatei dieselben Implementierungsmerkmale und dasselbe Laufzeitverhalten auf wie eine Datei, die ohne den Parameter **XML-ONLY** generiert wurde, einschließlich des Parsens der XML während der SOAP-Nachrichtenbehandlung. Um dieses Parsen zu verhindern, dürfen Sie SOAP message handlers nicht in Ihrer Pipelinekonfigurationsdatei angeben.
 - Wenn Sie keine Web-Service-Bindungsdatei verwenden möchten, konfigurieren Sie Ihre Service-Provider-Pipeline so, dass die Web-Service-Anforderung Ihre XML-fähige Anwendung erreicht. Sie können entweder den Terminal-Handler in der Pipelinekonfigurationsdatei für die Verwendung Ihres XML-fähigen Anwendungsprogramms konfigurieren oder Sie können einen Nachrichtenhandler erstellen, der dynamisch zu Ihrer Anwendung wechselt, abhängig von dem URI, der in der Pipeline empfangen wird.
2. Schreiben Sie Ihre Anwendung für die Verarbeitung der Web-Service-Anforderung, die in den folgenden Containern gespeichert wird:

DFHWS-BODY

Die Inhalte des SOAP-Hauptteils für eine eingehende SOAP-Anforderung, wenn die Pipeline einen von CICS bereitgestellten SOAP-Nachrichtenhandler enthält.

DFHREQUEST

Die vollständige Anforderung, einschließlich des Envelopes für eine SOAP-Anforderung, die aus der Pipeline empfangen wird.

DFHWS-XMLNS

Eine Liste mit Name/Wert-Paaren, die Namensbereichspräfixe zu Namensbereichen zuordnet, für den XML-Inhalt der Anforderung.

DFHWS-SOAPACTION

Der SOAPAction-Header, der der SOAP-Nachricht im Container DFHWS-BODY zugeordnet ist.

Wenn Sie API-Befehle für die Arbeit mit den Containern codieren, geben Sie nicht die Option **CHANNEL** an, weil alle Container dem aktuellen Kanal zugeordnet sind (d. h. dem Kanal, der an das Programm übergeben wurde). Wenn Sie den Namen des Kanals wissen müssen, verwenden Sie den Befehl **EXEC CICS ASSIGN CHANNEL**.

3. Optional: Ihre Anwendung kann auch zusätzliche Container verwenden, die für Nachrichtenhandler in der Pipeline verfügbar sind, ebenso wie andere Container, die die Nachrichtenhandler im Rahmen ihrer Verarbeitung erstellen. Eine vollständige Liste der Container finden Sie unter Containers used in the pipeline.
4. Wenn Ihre Anwendung die Anforderung verarbeitet hat, erstellen Sie eine Web-Service-Antwort mithilfe der folgenden Container:

DFHRESPONSE

Die vollständige Antwortnachricht, die an die Pipeline übergeben wer-

den soll. Verwenden Sie diesen Container, wenn Sie nicht SOAP für Ihre Nachrichten verwenden oder wenn Sie die vollständige SOAP-Nachricht, einschließlich des Envelopes, in Ihrem Programm erstellen wollen, statt den von CICS bereitgestellten SOAP-Nachrichtenhandler zu verwenden.

Wenn Sie einen SOAP-Hauptteil im Container DFHWS-BODY bereitstellen, wird DFHRESPONSE ignoriert.

DFHWS-BODY

Bei einer ausgehenden SOAP-Antwort die Inhalte des SOAP-Hauptteils. Stellen Sie diesen Container bereit, wenn der Terminal-Handler Ihrer Pipeline ein von CICS bereitgestellter SOAP-Nachrichtenhandler ist. Der Nachrichtenhandler erstellt die vollständige SOAP-Nachricht, die den Hauptteil enthält.

Ihr Programm muss diesen Container erstellen, auch wenn Anforderung und Antwort identisch sind. Tut es dies nicht, gibt CICS einen internen Serverfehler aus.

Sie können auch einen beliebigen anderen Container verwenden, um Informationen zu übergeben, die Ihre Pipeline für die Verarbeitung der ausgehenden Antwort benötigt.

Wenn Ihr Web-Service keine Antwort zurückgibt, müssen Sie den Container DFHNORESPONSE zurückgeben, um anzuzeigen, dass es keine Antwort gibt. Die Inhalte des Containers sind nicht wichtig, weil der Nachrichtenhandler nur prüft, ob der Container vorhanden ist oder nicht.

5. Erstellen Sie eine URIMAP-Ressource. Wenn Sie den Parameter **XML-ONLY** verwenden und einen Wert für den Parameter **URI** von DFHWS2LS angegeben haben, wird die URIMAP während des PIPELINE SCAN-Prozesses automatisch für Sie erstellt.

XML-fähige Service-Requester-Anwendungen erstellen

Ihre XML-fähigen Web-Service-Requester-Anwendung steuert die Datenkonvertierung zwischen der XML und der Programmiersprache und füllt die Steuercontainer in der Pipeline.

Before you begin

Sie können Ihre eigene XML-fähige Service-Requester-Anwendung unter Verwendung des Parameters **XML-ONLY** in DFHWS2LS schreiben oder Sie können auf den Einsatz dieses und anderer Tools verzichten. Die unkomplizierteste Methode, Ihre eigene XML-fähige Service-Requester-Anwendung zu schreiben, besteht darin, den Parameter **XML-ONLY** in DFHWS2LS zu verwenden. Der Parameter **XML-ONLY** ist nur auf Laufzeitebene 2.1 und höher verfügbar.

About this task

Die Verwendung des Parameters **XML-ONLY** führt dazu, dass eine WSBInd-Datei generiert wird, die CICS anweist, dass die Anwendung direkt mit den Inhalten des Containers DFHWS-BODY zusammenarbeiten wird. Die generierte WSBInd-Datei muss in einer Pipeline im Requestermodus installiert werden, um eine WEBSERVICE-Ressource im Requestermodus zu erstellen. Die Anwendung muss XML für den Hauptteil der Web-Service-Anforderung generieren und sie im Container DFHWS-BODY speichern. Sie muss anschließend den Befehl **EXEC CICS INVOKE SERVICE** absetzen. Die ausgehende Nachricht wird an den Web-Service-Provider ge-

sendet. Der Hauptteil der Antwortnachricht befindet sich nach Abschluss des Befehls ebenfalls im Container DFHWS-BODY.

Die XML der Antwortnachrichten wird während der SOAP-Nachrichtenbehandlung geparkt. Um dieses Parsen zu verhindern, dürfen Sie SOAP message handlers nicht in Ihrer Pipelinekonfigurationsdatei angeben.

XML-fähige Requester-Anwendungen können SOAP-Fehlernachrichten von der fernen Anwendung im Providermodus zurück empfangen. In diesem Fall ist die Requester-Anwendung verantwortlich für die Interpretation des SOAP-Fehlers und für dessen Unterscheidung von einer regulären Antwortnachricht. Wenn der Befehl **INVOKE SERVICE** mit einem **XML-ONLY-WEBSERVICE** verwendet wird, legt CICS den Antwortcode nicht fest, der üblicherweise verwendet wird, um anzugeben, dass ein SOAP-Fehler empfangen wurde.

Wenn Sie Ihre eigene XML-fähige Service-Requester-Anwendung schreiben, ohne die Option **XML-ONLY** anzugeben, führen Sie die folgenden Schritte aus:

Procedure

1. Erstellen Sie einen Kanal und füllen Sie ihn mit Containern. Die Steuercontainer müssen alle im CHAR-Modus gefüllt werden. Geben Sie die folgenden Informationen in den einzelnen Containern an:

DFHWS-PIPELINE

Der Name der PIPELINE-Ressource für die ausgehende Anforderung.

DFHWS-URI

Der URI des Ziel-Web-Service.

DFHWS-BODY

Bei einer ausgehenden SOAP-Anforderung die Inhalte des SOAP-Hauptteils. Stellen Sie diesen Container bereit, wenn die Pipeline einen von CICS bereitgestellten SOAP-Nachrichtenhandler enthält. Der Nachrichtenhandler erstellt die vollständige SOAP-Nachricht mit dem Hauptteil.

DFHREQUEST

Die vollständige Anforderungsnachricht, die an die Pipeline übergeben werden soll. Verwenden Sie diesen Container, wenn Sie nicht SOAP für Ihre Nachrichten verwenden oder wenn Sie die vollständige SOAP-Nachricht, einschließlich des Envelopes, in Ihrem Programm erstellen wollen. Die Pipeline darf keinen von CICS bereitgestellten SOAP-Nachrichtenhandler enthalten, um zu vermeiden, dass doppelte SOAP-Header in der ausgehenden Nachricht gesendet werden.

Wenn Sie einen SOAP-Hauptteil im Container DFHWS-BODY bereitstellen, muss DFHREQUEST leer sein. Wenn Sie Inhalte sowohl in DFHWS-BODY als auch in DFHREQUEST bereitstellen, verwendet CICS DFHREQUEST.

DFHWS-XMLNS

Eine Liste mit Name/Wert-Paaren, die Namensbereichspräfixe zu Namensbereichen zuordnet, für den XML-Inhalt der Anforderung.

DFHWS-SOAPACTION

Der SOAPAction-Header, der der SOAP-Nachricht hinzugefügt werden soll, die im Container DFHWS-BODY angegeben wird.

Tip: Wenn Sie den Container DFHWS-NOABEND zum Kanal hinzufügen, werden keine Abbrüche aus DFHPIRT heraus ausgegeben, wenn DFHPIRT aufgerufen wird. Dies ist nützlich, wenn Sie ein C/C++-Programm ausführen, weil Sie Fehler über den Container DFHERROR verarbeiten können.

2. Stellen Sie eine Verknüpfung zum Programm DFHPIRT her. Verwenden Sie diesen Befehl:

```
EXEC CICS LINK PROGRAM(DFHPIRT) CHANNEL(  
    benutzerkanal  
)
```

Dabei ist *benutzerkanal* der Kanal, der Ihre Container enthält. Die ausgehende Nachricht wird von den Nachrichtenhandlern und Programmen zur Headerverarbeitung in der Pipeline verarbeitet und an den Web-Service-Provider gesendet.

3. Rufen Sie die Container ab, die die Web-Service-Antwort aus demselben Kanal enthalten. Die Antwort vom Web-Service-Provider kann eine Erfolgsmeldung oder ein SOAP-Fehler sein. Die Web-Service-Requester-Anwendung muss beide Typen von Antworten vom Service-Provider verarbeiten können. Die vollständige Antwort ist in den folgenden Containern enthalten:

DFHRESPONSE

Die vollständige Antwort, einschließlich des Envelopes für eine SOAP-Antwort, empfangen vom Web-Service-Provider.

DFHWS-BODY

Wenn die Pipeline einen von CICS bereitgestellten SOAP-Nachrichtenhandler enthält, die Inhalte des SOAP-Hauptteils.

DFHERROR

Fehlerinformationen aus der Pipeline.

Anmerkung: In manchen Fällen wird DFHWS-BODY möglicherweise nicht aktualisiert. Sie müssen DFHRESPONSE auf einen SOAP-Fehler überprüfen.

Java mit Web-Services verwenden

Sie können Java zum Erstellen von Web-Service-Anwendungen verwenden. Zum Erstellen dieser Anwendungen werden andere Verfahren verwendet als bei anderen Programmiersprachen.

Für die meisten Nicht-Java-Programmiersprachen verwenden Sie die Web-Service-Assistenten, um Anwendungen zu aktivieren. Wenn Sie die Web-Service-Assistenten verwenden, konvertiert CICS die Daten aus dem Web-Service in ein für die Anwendung geeignetes Formular und platziert es in einem Container oder in einem Kommunikationsbereich. Sie können den Web-Service-Assistenten mit Java-Anwendungen verwenden, allerdings stellen die folgenden Tasks passendere Methoden zum Erstellen von Java-Web-Services für Java-Anwendungen bereit.

Web-Service im Java-Providermodus auf einem Axis2-JVM-Server implementieren

Sie können eine Axis2-Anwendung als Web-Service im Providermodus in CICS implementieren. Diese Anwendungen werden in der Regel mit JAX-WS generiert und können in einer Java-aktivierten Pipeline gehostet werden.

Sie können Java-Anwendungen aus den folgenden Gründen mithilfe dieser Methode implementieren:

- Sie verfügen über bereits getätigte Investitionen, die Axis2-Handler-Schnittstellen verwenden.
- Sie möchten eine CICS-Pipelinekonfiguration verwenden.

Anmerkung: Anwendungen im Axis2-Stil verwenden keine WEBSERVICE-Ressourcen. Sie interagieren mit CICS mithilfe des Axis2-Programmiermodells und können deshalb Teile der CICS-Web-Service-Unterstützung nicht verwenden. Die folgenden Services werden für Anwendungen im Axis2-Stil nicht unterstützt.

- SOAPFAULT CREATE
- WSACONTEXT GET
- DFHWS-OPERATION container
- DFHWS-MEP container
- DFHWS-USERID container
- DFHWS-TRANID container
- Options for securing SOAP messages

Before you begin

Sie müssen über eine Java-Anwendung verfügen, die für die Implementierung in Axis2 geeignet ist, z. B. eine POJO-Anwendung mit JAX-WS. Für diese Task wird die folgende POJO-Anwendung als Beispiel verwendet:

```
/**
 * Einfaches Beispiel
 */
@javax.jws.WebService(targetNamespace = "com.ibm.cics.example", name = "pojoExample")
public class TestAxis2
{
    public String getMessage(String input)
    {
        return "CICS got this: '" + input + "'";
    }
}
```

Diese Anwendung gibt den XML-Namensbereich an, der zum Generieren der WSDL verwendet wird, und einen Namen, der dem Web-Service zugeordnet werden soll.

Der Java-Code für diese Anwendung muss kompiliert und der JAX-WS-Generator muss ausgeführt werden, um die Anwendung in eine JAR-Datei namens TestAxis2.jar zu packen. Geben Sie dafür den folgenden Code aus:

```
javac TestAxis2.java
ws-gen -cp . TestAxis2 -wsdl
jar -cvf TestAxis2.jar *
```

Der JAX-WS-Generator erstellt außerdem ein WSDL-Dokument und die Bindungen, die von Axis2 verwendet werden.

About this task

Zum Implementieren eines Axis2-Web-Service müssen Sie die Pipelineinfrastruktur für Ihre Web-Services erstellen. Wenn Sie die Pipeline erstellt haben, können Sie Ihre Web-Services erstellen. Sie können die erstellte Pipeline für so viele Web-Services wiederverwenden, wie Sie benötigen. Die folgenden Schritte beschreiben, wie die Pipeline und die Web-Services erstellt werden.

Anmerkung: Im Rahmen dieser Task wird keine WEBSERVICE-Ressource erstellt oder installiert.

Procedure

1. Erstellen Sie die Pipelineinfrastruktur.
 - a. Erstellen Sie eine Web-Service-Infrastruktur für eine Java-Pipeline. Weitere Informationen finden Sie unter Creating the CICS infrastructure for a SOAP service provider.
 - b. Erstellen Sie ein Axis2-Repository. Erstellen Sie dazu eine Kopie des bereitgestellten Repositories in `$CICS_HOME/lib/pipeline/repository`.
 - c. Fügen Sie das Element `<repository>` zu Ihrer Pipelinekonfigurationsdatei hinzu. Dieses Element muss den Namen des Axis2-Repositorys angeben, das Sie erstellt haben.
 - d. Erstellen und aktivieren Sie eine PIPELINE-Ressource.
2. Wiederholen Sie die folgenden Schritte für jeden Web-Service, der dieser Pipeline zugeordnet ist, um den Web-Service zu erstellen.
 - a. Implementieren Sie die Axis2-Anwendung im Axis2-Repository. Beispielsweise muss die in dem Beispiel erstellte JAR-Datei in einem Verzeichnis namens `servicejars` im Repository-Verzeichnis implementiert werden. Sie müssen dieses Verzeichnis erstellen, wenn es noch nicht vorhanden ist.
 - b. Definieren und installieren Sie eine URIMAP-Ressource für den Web-Service. Die URIMAP-Ressource muss den URI und die PIPELINE-Ressource angeben, die dem Web-Service zugeordnet sind. Der URI muss der Axis2-Namenskonvention für URIs folgen. Die Axis2-Standardnamenskonvention lautet: `/ name_des_service Service. name_des_ports Port/ suffix`. Dabei sind `name_des_service` der Name des Web-Service in der WSDL, `name_des_ports` der Name des Ports in der WSDL und `suffix` ein optionales Suffix, das Sie definieren können. Im vorherigen Beispiel könnte die folgende URIMAP-Ressource verwendet werden:

```
Urimap : EXAMPLE
Group : EXAMPLE
Status : Enabled
USAge : Pipeline
SCHEME : HTTP
Port : No
HOST : *
Path : /TestAxis2Service.pojoExamplePort/example/TestAxis2
Transaction : CPIH
PIpeline : EXAMPLE
```

In diesem Beispiel wird angenommen, dass die verwendete PIPELINE-Ressource EXAMPLE heißt.

What to do next

Testen Sie, ob Ihre Web-Services ordnungsgemäß ausgeführt werden.

Java-Web-Service erstellen, der XML generiert und parst

Sie können Java-Anwendungen erstellen, die selbst XML parsen und generieren. Diese Anwendungen sind mit XML-fähigen Anwendungen konsistent, die in anderen Programmiersprachen geschrieben sind, aber sie profitieren von der Verwendung von Java-Standardtechnologien zur Verarbeitung der XML.

Procedure

1. Erstellen Sie eine XML-ONLY WEBSERVICE-Ressource. Weitere Informationen finden Sie unter „XML-fähige Service-Requester-Anwendungen erstellen“ auf Seite 634 oder „XML-fähige Service-Provider-Anwendungen erstellen“ auf Seite 632.
2. Schreiben Sie einen Java-Web-Service, der XML für den Hauptteil der SOAP-Nachricht parsen und generieren kann. Sie können verschiedene Tools verwenden, z. B. die Java 6 JAXB-Bibliothek (Java Architecture for XML Binding), um einen Java-Web-Service mit diesen Funktionalitäten zu erstellen.
3. Optional: Wenn Sie mit einer Provider-Pipeline arbeiten und die Funktionalität, eine SOAP-Fehlernachricht an den Requester zurückzugeben, hinzufügen möchten, verwenden Sie die JCICS-Klasse 'SoapFault', um den Befehl **EXEC CICS SOAPFAULT CREATE** auszugeben.
4. Optional: Wenn Sie mit einer Requester-Pipeline arbeiten, verwenden Sie die JCICS-Klasse 'Service', um den Befehl **EXEC CICS INVOKE SERVICE** auszugeben.

Java-Web-Service mit COBOL-Schnittstelle erstellen

Sie können Java-Anwendungen erstellen, die mit CICS interagieren, indem Sie dieselben Verfahren verwenden, die auch in anderen Programmiersprachen zum Einsatz kommen. Um diese Anwendungen zu erstellen, müssen Sie Java-Code schreiben oder generieren, der strukturierte Kommunikationsbereiche oder containerartige Daten erstellt.

Procedure

1. Verwenden Sie DFHWS2LS, um COBOL-Sprachstrukturen für den Web-Service zu erstellen.
2. Schreiben Sie einen Java-Web-Service, der COBOL-Sprachstrukturen generiert und parst. Weitere Informationen zu Tools, mit denen Java-Programme auf vorhandene CICS-Anwendungsdaten zugreifen können, sowie Links zu Beispielen für die Erstellung eines Java-Web-Service, der COBOL-Sprachstrukturen generieren und parsen kann, finden Sie unter Interacting with structured data from Java.
3. Optional: Wenn Sie mit einer Provider-Pipeline arbeiten und die Funktionalität, eine SOAP-Fehlernachricht an den Requester zurückzugeben, hinzufügen möchten, verwenden Sie die JCICS-Klasse 'SoapFault', um den Befehl **EXEC CICS SOAPFAULT CREATE** auszugeben.
4. Optional: Wenn Sie mit einer Requester-Pipeline arbeiten, verwenden Sie die JCICS-Klasse 'Service' als Schnittstelle zur CICS SERVICE-API, und geben Sie den Befehl **EXEC CICS INVOKE SERVICE** aus.

JAX-WS-Web-Service im Requestermodus implementieren

Sie können eine JAX-WS-Anwendung als Web-Service im Requestermodus in CICS erstellen. Diese Anwendungen verwenden jedoch nicht den Befehl **EXEC CICS INVOKE**, sondern interagieren stattdessen mit den fernen Web-Services mithilfe von JAX-WS.

Before you begin

Es muss ein JVM-Server konfiguriert sein, um OSGi zu unterstützen. Weitere Informationen finden Sie unter Setting up a JVM server.

About this task

Der Vorteil der Implementierung einer JAX-WS-Anwendung als Web-Service im Requestermodus besteht darin, dass Sie eine plattformunabhängige Web-Service-Requester-Anwendung erstellen, die zEnterprise Application Assist Processor (zAAP) verwendet. Die Verwendung von zAAP kann die Kosten von Transaktionen reduzieren. Weitere Informationen finden Sie in der IBM Redbooks-Veröffentlichung: zSeries Application Assist Processor (zAAP) Implementation.

Procedure

1. Erstellen Sie eine Web-Service-Requester-Anwendung in Java und verwenden Sie eine passende API, z. B. die Java-API für XML-Web-Services (JAX-WS), um den fernen Web-Service aufzurufen.
2. Optional: Wenn Sie JAX-WS verwenden, um einen fernen Web-Service zu starten, müssen Sie auch JAX-WS verwenden, um die SOAP-Nachrichten zu generieren, die Netzkommunikation zu steuern und die SOAP-Antwort zu verarbeiten.
3. Implementieren Sie Ihre Java-Anwendung und installieren Sie sie auf dem JVM-Server.

What to do next

Testen Sie, dass Ihre Web-Services korrekt starten.

Java-Web-Service im Providermodus auf einem Liberty-JVM-Server implementieren

Sie können eine Webanwendung als Web-Service im Providermodus auf einem Liberty-JVM-Server implementieren. Diese Anwendungen werden mithilfe der Java-Standards JAX-WS und JAXB erstellt. Dieses Thema bezieht sich nur auf Liberty im integrierten Modus von CICS.

About this task

CICS TS V5.3 enthält das aktuelle WebSphere Application Server-Liberty-Profil (WLP), das Funktionen für die Java-API für XML-Web-Services (JAX-WS) und für die Java-Architektur für die XML-Bindung (JAXB) bereitstellt. Zusammen ermöglichen diese Technologien es Ihnen, SOAP-Web-Services in Java als Teil einer CICS-Anwendung zu schreiben. Der folgende Artikel zeigt, wie Eclipse konfiguriert wird, wie ein Web-Service-Beispielprojekt getestet wird, wie das Beispiel in CICS implementiert wird, wie das Beispiel so geändert wird, dass es JCICS verwendet, und wie es mit dem Web-Service-Explorer getestet wird. Weitere Informationen finden Sie im CICS DevCenter-Artikel JAX-WS web service sample for Liberty.

Sie können Java-Anwendungen aus den folgenden Gründen mithilfe dieser Methode implementieren:

- Sie möchten Web-Services in Java erstellen.
- Sie haben komplexe WSDL-Dokumente, deren Verarbeitung sich mit den CICS-Web-Service-Assistenten schwierig gestalten würde.
- Sie möchten die Verarbeitung der Web-Service-Anwendung an zEnterprise Application Assist Processor (zAAP) auslagern.

Anmerkung: Webanwendungen, die auf einem Liberty-JVM-Server implementiert sind, verwenden nicht die Ressourcen WEBSERVICE oder TCPIPSERVICE. Sie in-

teragieren mit Webanforderungen mithilfe des Liberty-HTTP-Listeners und können deshalb nicht die Funktionen der CICS-Web-Service-Unterstützung nutzen.

SOAP-Nachrichten überprüfen

Bei Verwendung der CICS-Web-Services können Sie angeben, dass die SOAP-Nachrichten überprüft werden müssen, um sicherzustellen, dass sie mit dem Schema konform sind, das in der Web-Service-Beschreibung enthalten ist. Sie können Anwendungen sowohl im Provider- als auch im Requestermodus überprüfen.

Before you begin

Während der Entwicklung und des Tests Ihrer Web-Service-Implementierung hilft eine vollständige Validierung dabei, Probleme beim Nachrichtenaustausch zwischen einem Service-Requester und einem Service-Provider zu erkennen. Allerdings bringt die vollständige Validierung ein hohes Maß an Mehraufwand mit sich, deshalb wird nicht empfohlen, Nachrichten in einer umfassend getesteten Produktionsanwendung zu validieren.

CICS verwendet ein Java-Programm, um SOAP-Nachrichten zu validieren. Deshalb muss die Java-Unterstützung in Ihrer CICS-Region aktiviert sein, damit Sie SOAP-Nachrichten validieren können.

About this task

Die SOAP-Nachricht wird validiert, bevor sie in eine Anwendungsdatenstruktur umgesetzt wird und wenn eine SOAP-Nachricht aus der Anwendungsdatenstruktur generiert wird. Die SOAP-Nachricht wird mithilfe des XML-Schemas in der WSDL validiert und gegen die Umsetzungsanforderungen von CICS validiert. Sie können die WSDL-Datei verwenden, die im Attribut **WSDLFILE** der WEBSERVICE-Ressource angegeben ist, oder eine WSDL-Datei, die in der .zip-Datei enthalten ist, die wiederum im Attribut **ARCHIVEFILE** der WEBSERVICE-Ressource angegeben ist. Wenn beide Attribute angegeben sind, wird die WSDL-Datei in der Archivdatei, die im Attribut **ARCHIVEFILE** angegeben ist, verwendet.

Wenn die Validierung inaktiviert ist, verwendet CICS das Java-Programm nicht. CICS validiert SOAP-Nachrichten nur in dem Maße, in dem es erforderlich ist, um zu bestätigen, dass sie korrekt formatierte XML enthalten, und um sie umzusetzen. Aus diesem Grund kann es sein, dass eine SOAP-Nachricht zwar erfolgreich validiert wird, dann aber in der Laufzeitumgebung fehlschlägt und umgekehrt.

Procedure

1. Richten Sie einen OSGi-JVM-Server in der CICS-Region ein. Eine SOAP-Validierung mithilfe von DFHPIVAL kann nur in einem OSGi-Framework ausgeführt werden, nicht in einer Axis2- oder Liberty-Profil-JVM.
 - a. Installieren Sie den JVM-Beispielservers DFH\$JVMS in der Gruppe DFH\$OSGI oder erstellen Sie Ihren eigenen JVM-Server. Weitere Informationen finden Sie unter Setting up a JVM server.
 - b. Wenn Sie einen eigenen JVM-Server erstellt haben, ändern Sie die Programmdefinition DFHPIVAL in der Gruppe DFHPIVAL, um auf den Namen der JVMSERVER-Ressource zu verweisen. Die DFHPIVAL-Definition ist nicht gesperrt und kann bearbeitet werden. Standardmäßig referenziert die Definition DFH\$JVMS.
2. Stellen Sie sicher, dass Ihrer WEBSERVICE-Ressource eine Web-Service-Beschreibung zugeordnet ist. Diese Zuordnung wird für WEBSERVICE-Ressour-

cen erstellt, die automatisch erstellt werden, wenn eine WSDL-Datei oder eine .zip-Datei, die mindestens eine WSDL-Datei enthält, während eines Pipeline-Scans im Abholverzeichnis der Pipeline vorhanden ist.

Für WEBSERVICE-Definitionen, die mit der Onlineressourcendefinition erstellt werden, wird die Web-Service-Beschreibung mit dem Attribut WSDLFILE angegeben.

3. Aktivieren Sie die Web-Service-Validierung, indem Sie das Attribut **VALIDATION=YES** der WEBSERVICE-Ressource angeben. Sie können angeben, ob eine Validierung erforderlich ist, wenn Sie die Ressource definieren, und Sie können diese Einstellung ändern, nachdem die Ressource installiert wurde.

Results

Überprüfen Sie das Systemprotokoll, um zu sehen, ob die SOAP-Nachricht gültig ist. Die Nachricht DFHPI1002 gibt an, dass die SOAP-Nachricht erfolgreich validiert wurde, und die Nachricht DFHPI1001 gibt an, dass die Validierung fehlgeschlagen ist.

What to do next

Inaktivieren Sie die Validierung, wenn Sie sie nicht länger benötigen.

Ungültige und nicht initialisierte von Anwendungen bereitgestellte Daten verarbeiten

Vorgehensweise zum Tolerieren von ungültigen Daten, die während der Umsetzung zwischen JSON und Anwendungsdaten gefunden werden.

Das Aktivieren einer vorhandenen Anwendung für einen Service führt dazu, dass CICS die Inhalte des Kommunikationsbereichs oder der Container erkennt, die von dieser Anwendung verwendet werden. Die WSBInd-Datei, die in CICS implementiert wird, enthält Informationen zu dem Datenformat, einschließlich der Namen, Position und des Datentyps jedes einzelnen Felds. CICS verwendet diese Informationen, um die Umsetzung zwischen der XML-/JSON-Datendarstellung und den Anwendungsdaten zu vereinfachen.

Wenn die Anwendungsdaten nicht mit den Informationen konsistent sind, die in der WSBInd-Datei gespeichert sind, berichtet CICS ein Problem und kann die Daten nicht umsetzen. Beispiel: Eine Anwendungsschnittstelle wird geändert, aber die WSBInd-Datei wird nicht neu generiert und nicht erneut implementiert. Diese Art von Problem resultiert oft in der Fehlernachricht DFHPI1010.

Eine subtilere Variante dieser Bedingung kann in Szenarios auftreten, in denen die Anwendung Felder absichtlich nicht initialisiert hat. Ein signiertes gepacktes Dezimalfeld kann beispielsweise niedrige Werte (Nullbytes) enthalten. Wenn CICS dieses Feld verarbeitet, kann nicht festgestellt werden, ob der ungültige Wert absichtlich nicht initialisiert wurde. CICS zeigt die Fehlernachricht DFHPI1010 an und die Datenumsetzung schlägt fehl. Anwendungen können Felder aus diversen Gründen nicht initialisieren. Beispiele:

- Eine Sprachstruktur kann sowohl die Eingabe- als auch die Ausgabedatenformate beschreiben, aber das nicht initialisierte Feld diene nur zur Eingabe.
- Die Anwendung kann einen Fehler feststellen, einen Antwortcode festlegen und zurückkehren, ohne die anderen Ausgabefelder initialisiert zu haben.

- Bedingte Logik in der Anwendung kann unter Umständen steuern, ob das Feld als aussagekräftig betrachtet wird.

Die ideale Antwort auf dieses Szenario ist, die Anwendung zu ändern, um sicherzustellen, dass keine ungültigen Werte zur Verarbeitung an CICS übergeben werden. Eine alternative Möglichkeit ist, die Option DATA-SCREENING des Assistenten zum Zeitpunkt der Generierung der WSBIND-Datei auf DISABLED zu setzen. Diese Option bewirkt, dass CICS von der Anwendung bereitgestellte fehlerhafte Werte toleriert und dass solche Werte durch einen Standardwert, typischerweise Null, ersetzt werden. Diese Optionen können die Serviceaktivierung vorhandener Anwendungen einfacher gestalten, aber die Fehlererkennung erschweren, und müssen deshalb umsichtig verwendet werden. Wenn das Datenscreening inaktiviert ist, ist es wahrscheinlicher, dass die von der Anwendung generierten Datenfehler von CICS nicht erkannt werden.

Eine ähnliche Bedingung kann auftreten, wenn von Anwendungen generierte Arrays teilweise nicht gefüllt werden. Beispiel: Bei einem Array von 1000 Datensätzen initialisiert die Anwendung möglicherweise die ersten zehn Datensätze. CICS generiert eine XML- oder JSON-Darstellung der Daten und füllt alle 1000 Felder, obwohl 990 davon leer sind. Die ideale Antwort auf dieses Problem ist die Einführung einer Klausel OCCURS DEPENDING ON in der Anwendung, um genau anzugeben, wie viele Datensätze als gefüllt angenommen werden. Eine Alternative ist die Verwendung der Option TRUNCATE=NULL-ARRAYS der Assistenten, wenn die WSBIND-Datei generiert wird. Diese Option weist CICS an, zu versuchen, nicht initialisierte Daten zu erkennen, und das Array an diesem Punkt abzuschneiden. Die Verwendung dieser Option kann zu eleganten JSON/XML-Daten führen, die aus mehrdeutigen und nicht initialisierten Anwendungsdaten generiert wurden, birgt aber das Risiko eines zufälligen Datenverlusts, wenn der Array-Inhalt nicht von nicht initialisierten Daten unterscheidbar ist.

Die beste Lösung besteht darin, dass Anwendungen eindeutige Daten erzeugen, die vollständig mit der Sprachstruktur konsistent sind, die diese Daten beschreibt. Die Verwendung der Optionen TRUNCATE=NULL-ARRAYS=ENABLED und DATA-SCREENING=DISABLED kann dazu führen, dass CICS ungeeignete von Anwendungen bereitgestellte Daten toleriert, birgt aber ein gewisses Risiko und eine gewisse Unsicherheit für den Prozess.

Das folgende Beispiel veranschaulicht, wie diese Optionen verwendet werden.

Beispiel 1: Tolerierung von Dezimalfeldern

Testen der automatischen Korrektur ungültiger Daten in einem Dezimalfeld.

About this task

Dieses Szenario zeigt, wie der Standardwert 0 in Dezimalfeldern, die ungültige Daten enthalten, automatisch festgelegt werden kann.

Procedure

1. Generieren Sie das JSON-Schema und erforderliche Artefakte für CICS, um zwischen JSON und COBOL-Anwendungsdaten umzusetzen.
 - a. Bereiten Sie das Copybook vor.

```

03 BAD-DATA.
05 NORMAL-NUM PIC 9(2).
05 NORMAL-CHAR PIC X(3).
05 PZONED-DECIMAL PIC S9(4) DISPLAY.
05 NZONED-DECIMAL PIC S9(4) DISPLAY.
05 UZONED-DECIMAL PIC 9(4) DISPLAY.
05 NORMAL-CHAR2 PIC X(3).
05 PBINARY PIC S9(4) BINARY.
05 NBINARY PIC S9(4) COMP.
05 UBINARY PIC 9(4) COMP.
05 NORMAL-NUM2 PIC 9(3).
05 PPACKED-DECIMAL PIC S9(4) COMP-3.
05 NPACKED-DECIMAL PIC S9(4) COMP-3.
05 UPACKED-DECIMAL PIC 9(4) COMP-3.
05 NORMAL-NUM3 PIC 9(2).
05 NORMAL-CHAR3 PIC X(3).
05 FLOAT-ZONED PIC S9(4)V99.
05 FLOAT-PACKED PIC S9(4)V99 COMP-3.

```

- b. Übergeben Sie die JCL. Setzen Sie DATA-SCREENING auf DISABLED, um die Funktion zu aktivieren.

```

//GENJSON JOB ('abrechnungsdaten',name),CLASS=M,REGION=0M,
// MSGCLASS=A,NOTIFY=&SYSUID
//JCLLIB JCLLIB ORDER=ZZZZ.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=ZZZZ.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. Überprüfen Sie, dass die generierten JSON-Schemas (Anforderung und Antwort) und die WSBind-Dateien erfolgreich erstellt wurden.
2. Erstellen Sie das Anwendungsprogramm.
- a. Weisen Sie im COBOL-Programm allen Feldern, außer Dezimalzahlen, einen Wert zu.

```

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. Definieren Sie CICS-Ressourcen.
- a. Definieren und installieren Sie das COBOL-Programm.
- b. Definieren und installieren Sie TCPIPSERVICE.
- c. Definieren und installieren Sie basierend auf Ihrem Parser PIPELINE.

4. Testen Sie die Anwendung.
 - a. Führen Sie PIPELINE SCAN aus. Verwenden Sie für den in der WSbind-Datei beschriebenen Web-Service INSERVICE.
 - b. Senden Sie die JSON-Anforderung.

```
{"BADDATA10operation":{"bad_data":{"normal_num":12}}}
```

- c. Überprüfen Sie die Antwort.

Results

Nicht initialisierten Feldern werden Standardwerte zugewiesen.

```
{
  "BADDATA10operationResponse": {
    "bad_data": {
      "normal_num": 11,
      "normal_char": "AAA",
      "pzoned_decimal": 0,
      "nzoned_decimal": 0,
      "uzoned_decimal": 0,
      "normal_char2": "BBB",
      "pbinary": 0,
      "nbinary": 0,
      "ubinary": 0,
      "normal_num2": 222,
      "ppacked_decimal": 0,
      "npacked_decimal": 0,
      "upacked_decimal": 0,
      "normal_num3": 0,
      "normal_char3": "CCC",
      "float_zoned": 0,
      "float_packed": 0
    }
  }
}
```

Kapitel 4. Unterstützung für den Schutz von Web-Services

CICS Transaction Server for z/OS bietet Unterstützung für eine Reihe verwandter Technologien zum Schutz von SOAP- und JSON-Nachrichten.

Manche dieser Technologien sind als Teil des HTTP-Protokolls verfügbar und gleichermaßen anwendbar auf SOAP und JSON. Einige verwenden die Spezifikation 'Web Services Security (WSS): SOAP Message Security 1.0' und stehen nur für SOAP zur Verfügung. Informationen zu den gemeinsam genutzten TCP/IP- und HTTP-Sicherheitsoptionen finden Sie unter Security for TCP/IP clients und Security for CICS web support.

Informationen zu SAML-Zusicherungen finden Sie unter Configuring CICS for SAML.

Sicherheit von SOAP-Web-Services

In 'Web Services Security (WSS): SOAP Message Security 1.0' wird die Verwendung von *Sicherheitstoken* und *digitalen Signaturen* zum Schutz und zur Authentifizierung von SOAP-Nachrichten beschrieben. Weitere Informationen finden Sie in der Spezifikation Web Services Security: SOAP Message Security 1.0. Web Services Security: SOAP Message Security 1.0

Web Services Security schützt die *Privatsphäre* und *Integrität* von SOAP-Nachrichten, indem Nachrichten vor nicht autorisierter Offenlegung geschützt und nicht autorisierte und nicht erkannte Änderungen verhindert werden. WSS stellt diesen Schutz durch die digitale Unterzeichnung und Verschlüsselung von XML-Elementen in der Nachricht bereit. Elemente, die geschützt werden können, sind der Hauptteil bzw. beliebige Elemente im Hauptteil oder der Header. Sie können verschiedenen Elementen in der SOAP-Nachricht unterschiedliche Stufen von Schutz gewähren.

Die WS-Trust-Spezifikation (Web Services Trust Language) erweitert Web Services Security durch Bereitstellung eines Frameworks für das Anfordern und Ausgeben von Sicherheitstoken sowie für das Verwalten von Vertrauensbeziehungen zwischen Web-Service-Requestern und -Providern. Diese Erweiterung der Authentifizierung von SOAP-Nachrichten ermöglicht es Web-Services, Sicherheitstoken verschiedener Typen über einen vertrauenswürdigen Dritten zu validieren und auszutauschen. Dieser Dritte wird als *Sicherheitstokenservice* (STS) bezeichnet. Weitere Informationen zur Web Services Trust Language finden Sie in der Spezifikation Web Services Trust Language.

CICS Transaction Server for z/OS bietet Unterstützung für diese Spezifikationen mithilfe eines von CICS bereitgestellten Sicherheitshandlers in der Pipeline.

- Bei ausgehenden Nachrichten unterstützt CICS die digitale Signatur und Verschlüsselung des ganzen SOAP-Hauptteils. CICS kann auch ein Benutzernamens-token gegen ein Sicherheitstoken eines anderen Typs über einen STS austauschen.
- Bei eingehenden Nachrichten unterstützt CICS die Verschlüsselung und digitale Signatur des Hauptteils bzw. von Elementen des Hauptteils und Headers der Nachricht. CICS kann auch Sicherheitstoken mit einem STS austauschen und validieren.

CICS stellt auch eine separate vertrauenswürdige Clientschnittstelle bereit, damit Sie mit einem STS interagieren können, ohne den CICS-Sicherheitshandler zu verwenden.

Anmerkung: Web Services Security ist möglicherweise nicht konform mit SP800-131A. Web Services Security wird durch Hinzufügen eines Handlers in der Pipeline konfiguriert und CICS hat keine Kontrolle über die Verarbeitung in einem vom Benutzer geschriebenen Handler. Wenn Sie digitale Signaturen verwenden, können Sie nur die Algorithmen `dsa-sha1` und `rsa-sha1` angeben. Diese Algorithmen sind nicht konform mit SP800-131A. Der Triple-DES-Verschlüsselungsalgorithmus mit zwei Schlüsseln, mit dem ein SOAP-Hauptteil verschlüsselt werden kann, ist ebenfalls nicht konform.

Voraussetzungen für Web Services Security

Um Web Services Security zu implementieren, müssen Sie diese Updates in Ihrer CICS-Region anwenden: Installieren Sie das IBM XML Toolkit for z/OS v1.10, wenden Sie APAR OA14956 an und fügen Sie drei Bibliotheken zur DFHRPL-Verkettung hinzu.

About this task

Führen Sie die folgenden Schritte aus, bevor Sie Web Services Security implementieren:

Procedure

1. Installieren Sie das kostenlose IBM XML Toolkit for z/OS v1.10. Sie können es von der folgenden Site herunterladen: <http://www.ibm.com/servers/eserver/zseries/software/xml/>. Sie müssen Version 1.10 installieren. Spätere Versionen funktionieren nicht mit der Web Services Security-Unterstützung in CICS.
2. Wenden Sie ICSF APAR OA14956 an, wenn es nicht bereits in z/OS installiert ist.
3. Fügen Sie die folgenden Bibliotheken zur DFHRPL-Verkettung hinzu:
 - `hlq.SIXMLOD1`, wobei `hlq` das übergeordnete Qualifikationsmerkmal des XML Toolkit ist.
 - `hlq.SCEERUN`, wobei `hlq` das übergeordnete Qualifikationsmerkmal der Language Environment ist.
 - `hlq.SDFHWSLD`, wobei `hlq` das übergeordnete Qualifikationsmerkmal der CICS-Installation ist, z. B. `CICSTS55`.

Die ersten beiden Bibliotheken enthalten DLLs, die zur Laufzeit für den Sicherheitshandler erforderlich sind. `IXM4C57` wird vom XML Toolkit bereitgestellt und befindet sich in `'hlq.SIXMLOD1'`. `C128N` wird von der Language Environment-Laufzeit bereitgestellt und befindet sich in `'hlq.SCEERUN'`.

Die Bibliothek `'hlq.SDFHWSLD'` ermöglicht CICS die Suche nach den Web Services Security-Modellen `DFHWSSE1` und `DFHWSXXX`.

4. Möglicherweise müssen Sie den Wert des Systeminitialisierungsparameters **EDSALIM** erhöhen. Die drei geladenen DLLs erfordern ca. 15 MB EDSA-Speicher.

Results

Wenn die Bibliotheken nicht angegeben wurden, wird die folgende Nachricht angezeigt:

CEE3501S Das Modul `modulname` wurde nicht gefunden.

Sicherheit von SOAP-Web-Services planen

Sie können entscheiden, welche die beste Methode ist, Ihre Web-Services zu sichern. CICS unterstützt eine Reihe von Optionen, darunter einen konfigurierbaren Sicherheitsnachrichtenhandler und eine separate vertrauenswürdige Clientschnittstelle.

About this task

CICS implementiert Web Services Security (WS-Security oder WSS) auf Pipelineebene statt für jeden einzelnen Web-Service. Beantworten Sie die folgenden Fragen, um zu entscheiden, wie Sie Sicherheit implementieren möchten.

Procedure

1. Ist die Leistung Ihrer Pipelineverarbeitung wichtig? Die Verwendung von WSS zum Sichern Ihrer Web-Services wirkt sich deutlich auf die Leistung aus.
Der Hauptvorteil der Implementierung von WSS besteht darin, dass Sie durch Verschlüsseln eines Teils einer SOAP-Nachricht die Nachricht über eine Kette von Zwischenknoten senden können, die möglicherweise alle einen legitimen Grund haben, den SOAP-Header zu prüfen, um Entscheidungen hinsichtlich der Weiterleitung oder Verarbeitung zu treffen. Aber diese Zwischenknoten dürfen nicht den Inhalt der Nachricht anzeigen. Indem Sie nur solche Abschnitte verschlüsseln, die vertraulich sein müssen, realisieren Sie die folgenden Vorteile:
 - Sie erzeugen keinen zusätzlichen Systemaufwand für das Verschlüsseln und Entschlüsseln an jedem Knoten in einer Kette von Zwischenprozessen.
 - Sie können eine vertrauliche Nachricht über ein öffentliches Netz von nicht vertrauenswürdigen Knoten senden, wobei nur der letzte Empfänger der Daten diese lesen kann.Als Alternative zu WSS können Sie SSL verwenden, um den gesamten Datenstrom zu verschlüsseln.
2. Wenn Sie WSS verwenden möchten, welche Sicherheitsstufe benötigen Sie? Die Optionen reichen von einer Basisauthentifizierung, bei der der Nachrichtenheader einen Benutzernamen und ein Kennwort enthält, bis zu einer Kombination von digitalen Signaturen und Verschlüsselung in der Nachricht. Die Optionen, die der CICS-Sicherheitshandler unterstützt, werden unter „Optionen zum Sichern von SOAP-Nachrichten“ auf Seite 650 beschrieben.
3. Erfüllt der von CICS bereitgestellte Sicherheitshandler Ihre Anforderungen? Wenn Sie eine erweiterte Sicherheitsverarbeitung nutzen möchten, müssen Sie Ihren eigenen angepassten Sicherheitshandler schreiben. Dieser Handler muss die erforderliche Authentifizierung von Nachrichten ausführen, entweder direkt mit RACF oder mithilfe eines Sicherheitstokenservice, und die Verarbeitung von digitalen Zertifikaten und verschlüsselten Elementen abwickeln. Ausführliche Informationen finden Sie unter „Angepassten Sicherheitshandler schreiben“ auf Seite 666.
4. Enthält Ihre Pipeline einen MTOM-Handler? Wenn Sie planen, sowohl den MTOM-Handler als auch den Sicherheitshandler in Ihrer Pipelinekonfigurationsdatei zu aktivieren, werden alle MIME Multipart- oder Related-Nachrichten im Kompatibilitätsmodus verarbeitet, weil der Sicherheitshandler die XOP-Elemente im Hauptteil der Nachricht nicht parsen kann. Diese Verarbeitung kann sich ebenfalls auf die Leistung der Pipelineverarbeitung auswirken.

Optionen zum Sichern von SOAP-Nachrichten

CICS unterstützt sowohl das Signieren als auch das Verschlüsseln von SOAP-Nachrichten. Daher können Sie die Sicherheitsstufe auswählen, die für die in der SOAP-Nachricht gesendeten oder empfangenen Daten am besten geeignet ist.

Das Signieren oder Verschlüsseln von SOAP-Nachrichten wird für Axis2-Web-Service-Java-Anwendungen im Providermodus oder für Provider-Web-Services, die unter Verwendung des Axis2-Nachrichtenkontexts an die Pipeline angehängt werden, nicht unterstützt.

Sie haben folgende Optionen:

Sichere Authentifizierung

In Service-Provider-Pipelines kann CICS ein Benutzernamenstoken im SOAP-Nachrichtenheader als vertrauenswürdig akzeptieren. Dieser Typ von Sicherheitstoken enthält in der Regel einen Benutzernamen und ein Kennwort, in diesem Fall ist jedoch kein Kennwort erforderlich. CICS vertraut dem angegebenen Benutzernamen und platziert ihn im Container DFHWS-USERID, und die Nachricht wird in der Pipeline verarbeitet.

In Service-Requester-Pipelines kann CICS ein Benutzernamenstoken ohne das Kennwort im SOAP-Nachrichtenheader an den Service-Provider senden.

Basisauthentifizierung

Im Service-Providermodus kann CICS ein Benutzernamenstoken im SOAP-Nachrichtenheader für die Authentifizierung in eingehenden SOAP-Nachrichten akzeptieren. Dieser Typ von Sicherheitstoken enthält einen Benutzernamen und ein Kennwort. CICS überprüft das Benutzernamenstoken mithilfe eines externen Sicherheitsmanagers, z. B. RACF. Ist dies erfolgreich, wird der Benutzername im Container DFHWS-USERID platziert und die SOAP-Nachricht wird in der Pipeline verarbeitet. Wenn CICS das Benutzernamenstoken nicht überprüfen kann, wird eine SOAP-Fehlernachricht an den Service-Requester zurückgegeben.

Benutzernamenstoken, die Kennwörter enthalten, werden im Service-Requestermodus oder in ausgehenden SOAP-Nachrichten nicht unterstützt.

HTTP-Basisauthentifizierung

Im Service-Providermodus kann CICS Basisauthentifizierungsinformationen über ein HTTP-Protokoll akzeptieren. Der Service-Requester verwendet eine URIMAP-Definition, um anzugeben, dass Berechtigungsnachweise (Informationen zur Identifizierung von Benutzern) von dem globalen Benutzerexit, XWBAUTH, erfasst werden können. XWBAUTH übergibt diese Informationen an CICS auf Anforderung, und CICS sendet die Informationen in einem HTTP-Berechtigungsheader an den Service-Provider.

Erweiterte Authentifizierung

In Service-Provider- und Service-Requester-Pipelines können Sie Sicherheitstoken mit einem Sicherheitstokenservice (STS) zu Authentifizierungszwecken überprüfen oder austauschen. Diese Authentifizierung ermöglicht CICS, Nachrichten mit Sicherheitstoken im Nachrichtenheader, die normalerweise nicht unterstützt werden, zu akzeptieren und zu senden, z. B. Kerberos-Tokens oder SAML-Zusicherungen.

Für eine eingehende Nachricht können Sie auswählen, ob ein Sicherheitstoken überprüft oder ausgetauscht werden soll. Wenn die Anforderung ist, das Sicherheitstoken auszutauschen, muss CICS ein Benutzernamenstoken

vom STS zurück empfangen. Für eine ausgehende Nachricht können Sie ein Benutzernamenstoken nur für gegen ein Sicherheitstoken austauschen.

Signieren mit X.509-Zertifikaten

Im Service-Provider- und Service-Requestermodus können Sie ein X.509-Zertifikat im SOAP-Nachrichtenheader bereitstellen, um den Hauptteil der SOAP-Nachricht zur Authentifizierung zu signieren. Dieser Typ von Sicherheitstoken wird als *binäres Sicherheitstoken* bezeichnet. Um binäre Sicherheitstoken von eingehenden SOAP-Nachrichten zu akzeptieren, muss der öffentliche Schlüssel, der dem Zertifikat zugeordnet ist, in einen externen Sicherheitsmanager, z. B. RACF, importiert werden und dem Schlüsselring zugeordnet werden, der im Systeminitialisierungsparameter **KEYRING** angegeben ist. Für ausgehende SOAP-Nachrichten generieren und veröffentlichen Sie den öffentlichen Schlüssel für die gewünschten Empfänger. Die Integrated Cryptographic Service Facility (ICSF) wird zum Generieren von öffentlichen Schlüsseln verwendet.

Verwenden Sie nicht die folgenden Zeichen, wenn Sie die Bezeichnung angeben, die einem digitalen X.509-Zertifikat zugeordnet ist:

< > ; ! =

Sie können auch ein zweites X.509-Zertifikat in den Header einschließen und es mithilfe des ersten Zertifikats unterzeichnen. Mit diesem zweiten Zertifikat können Sie die Arbeit in CICS unter der Benutzer-ID ausführen, die dem zweiten X.509-Zertifikat zugeordnet ist. Das Zertifikat, mit dem Sie die SOAP-Nachricht unterzeichnen, muss einer vertrauenswürdigen Benutzer-ID zugeordnet sein und über eine Ersatzberechtigung verfügen, um zu bestätigen, dass die Arbeit unter einer anderen ID, der *bestätigten ID*, ausgeführt wird, ohne dass die vertrauenswürdige Benutzer-ID dieser Identität zugeordnet wird.

Verschlüsselung

Im Service-Provider- und Service-Requestermodus können Sie den SOAP-Nachrichtenhauptteil mithilfe eines symmetrischen Algorithmus wie Triple DES oder AES verschlüsseln. Bei einem symmetrischen Algorithmus wird derselbe Schlüssel verwendet, um Daten zu verschlüsseln und zu entschlüsseln. Dies wird als *symmetrischer Schlüssel* bezeichnet. Er wird dann in die Nachricht eingeschlossen und mithilfe einer Kombination aus dem öffentlichen Schlüssel des gewünschten Empfängers und dem asymmetrischen Schlüsselverschlüsselungsalgorithmus RSA 1.5 verschlüsselt. Diese Verschlüsselung bietet eine höhere Sicherheit, da der asymmetrische Algorithmus komplex ist und sich der symmetrische Schlüssel sich nur schwer entschlüsseln lässt. Sie erzielen aber eine bessere Leistung, da der Großteil der SOAP-Nachricht mit dem symmetrischen Algorithmus verschlüsselt ist, der sich schneller entschlüsseln lässt.

Für eingehende SOAP-Nachrichten können Sie ein Element im SOAP-Hauptteil verschlüsseln und dann den SOAP-Hauptteil als Ganzes verschlüsseln. Diese Art von Verschlüsselung kann sich insbesondere für ein Element eignen, das sensible Daten enthält. Wenn CICS eine SOAP-Nachricht mit zwei Ebenen von Verschlüsselung empfängt, entschlüsselt CICS automatisch beide Ebenen. Diese Art der Verschlüsselung wird für ausgehende SOAP-Nachrichten nicht unterstützt.

CICS bietet keine Unterstützung für eingehende SOAP-Nachrichten, die nur ein verschlüsseltes Element im Nachrichtenheader und keine verschlüsselten Elemente im SOAP-Hauptteil enthalten.

Signieren und Verschlüsseln

Im Service-Provider- und Service-Requestermodus können Sie auswählen, dass eine SOAP-Nachricht sowohl signiert als auch verschlüsselt wird. CICS unterzeichnet immer zunächst den SOAP-Hauptteil und verschlüsselt ihn dann. Der Vorteil dieser Methode besteht darin, dass Sie sowohl Nachrichtenschutz als auch Nachrichtenintegrität erzielen.

ICRX-basierte Identitätsweitergabe

Im Service-Providermodus können Sie ein nicht authentifiziertes ICRX-Identitätstoken (Extended Identity Context Reference) unter denselben Umständen verwenden wie ein nicht authentifiziertes WS-Security-Benutzer-ID-Token. Ein ICRX-Identitätstoken ist eine z/OS-ID, die einer Benutzer-ID zugeordnet wird. CICS löst das ICRX-Identitätstoken in eine Benutzer-ID auf und platziert eine Kopie im Container DFHWS-ICRX. CICS füllt außerdem den Container DFHWS-USERID. Weitere Informationen zu ICRX-Identitätstoken finden Sie unter Identity propagation and distributed security.

Authentifizierung mit einem Sicherheitstokenservice

CICS kann mit einem Sicherheitstokenservice (STS) wie Tivoli Federated Identity Manager zusammenarbeiten, um eine erweiterte Authentifizierung von Web-Services bereitzustellen.

Ein Sicherheitstokenservice (STS) fungiert als vertrauenswürdiger Drittanbieter zum Aushandeln von Vertrauensbeziehungen zwischen einem Web-Service-Requester und einem Web-Service-Provider. Auf ähnliche Weise wie eine Zertifizierungsstelle in einem SSL-Handshake garantiert der STS, dass der Requester und der Provider den Berechtigungsnachweisen, die in der Nachricht bereitgestellt werden, vertrauen können. Diese Vertrauensbeziehung wird durch den Austausch von Sicherheitstoken dargestellt. Ein STS kann diese Sicherheitstoken ausgeben, austauschen und validieren und so die Vertrauensbeziehungen einrichten, damit Web-Services aus unterschiedlichen vertrauenswürdigen Domänen erfolgreich miteinander kommunizieren können. Weitere Details finden Sie in der Web Services Trust Language-Spezifikation.

CICS fungiert als vertrauenswürdiger Client und kann zwei Typen von Web-Service-Anforderungen an einen STS senden. Der erste Typ von Anforderung ist die Validierung des Sicherheitstokens im WS-Security-Nachrichtenheader, der zweite Typ von Anforderung ist der Austausch des Sicherheitstokens gegen ein andersartiges. Diese Anforderungen ermöglichen CICS, Nachrichten zu senden und zu empfangen, die verschiedene Sicherheitstoken einer breiten Auswahl von vertrauenswürdigen Domänen enthalten, darunter SAML-Zusicherungen und Kerberos-Token.

Sie können entweder den CICS-Sicherheitshandler konfigurieren, um zu definieren, wie CICS mit einem STS interagiert, oder Sie können Ihren eigenen Nachrichtenhandler schreiben, um eine separat bereitgestellte vertrauenswürdige Clientschnittstelle zu verwenden. Unabhängig von der Methode, für die Sie sich entscheiden, verwenden Sie SSL zum Sichern der Verbindung zwischen CICS und dem STS.

Vorgehensweise des Sicherheitshandlers zum Aufrufen des STS

Der CICS-Sicherheitshandler verwendet die Informationen in der Pipelinekonfigurationsdatei, um eine Web-Service-Anforderung an den Sicherheitstokenservice (STS) zu senden. Der Typ der gesendeten Anforderung hängt von der Aktion ab, die der STS ausführen soll.

In einer Service-Provider-Pipeline

In einer Service-Provider-Pipeline unterstützt der Sicherheitshandler zwei Typen von Aktionen, abhängig von der Art, wie Sie den Sicherheitshandler konfigurieren:

- Senden Sie eine Anforderung an den STS, um die erste Instanz eines Sicherheitstokens oder das erste Sicherheitstoken eines bestimmten Typs im WS-Security-Header der eingehenden Nachricht zu validieren.
- Senden Sie eine Anforderung an den STS, um die erste Instanz eines Sicherheitstokens oder das erste Sicherheitstoken eines bestimmten Typs im WS-Security-Header der eingehenden Nachricht gegen ein Sicherheitstoken zu tauschen, das CICS lesen kann.

Der Sicherheitshandler erstellt dynamisch eine Pipeline, um die Web-Service-Anforderung an den STS zu senden. Diese Pipeline ist solange vorhanden, bis eine Antwort vom STS empfangen wird. Anschließend wird sie gelöscht. Ist die Anforderung erfolgreich, gibt der STS ein Identitätstoken oder den Gültigkeitsstatus des Tokens zurück. Der Sicherheitshandler platziert die RACF-ID, die aus dem Token abgerufen wurde, im Container DFHWS-USERID.

Wenn der STS einen Fehler feststellt, gibt er einen SOAP-Fehler an den Sicherheitshandler zurück. Der Sicherheitshandler gibt dann einen Fehler an den Web-Service-Requester zurück.

In einer Service-Requester-Pipeline

In einer Service-Requester-Pipeline kann der Sicherheitshandler nur den Austausch eines Tokens mit dem STS anfordern. Die Pipelinekonfigurationsdatei definiert, welchen Typ von Token der STS an den Sicherheitshandler zurückgibt.

Wenn die Anforderung erfolgreich ist, wird die RACF-ID im Container DFHWS-USERID platziert und das Token wird in den ausgehenden Nachrichtenheader eingeschlossen. Wenn der STS einen Fehler feststellt, gibt er einen SOAP-Fehler an den Sicherheitshandler zurück. Der Sicherheitshandler gibt dann einen Fehler über die Pipeline an die Web-Service-Requester-Anwendung zurück.

Der Sicherheitshandler kann nur einen Typ von Aktion vom STS für die Pipeline anfordern. Er kann außerdem nur einen Typ von Token für eine ausgehende Anforderungsnachricht austauschen und ist beschränkt auf das Verarbeiten des ersten Tokens im WS-Security-Nachrichtenheader - entweder die erste Instanz oder die erste Instanz eines bestimmten Typs. Diese Optionen decken die gängigsten Szenarios für die Verwendung eines STS ab, stellen Ihnen jedoch möglicherweise nicht die Verarbeitungsmöglichkeiten bereit, die für die Verarbeitung eingehender und ausgehender Nachrichten erforderlich sind.

Wenn Sie spezifischere Verarbeitungsmöglichkeiten bereitstellen möchten, um viele Tokens in den eingehenden Nachrichtenheadern verarbeiten oder um mehrere Typen von Tokens für ausgehende Nachrichten austauschen zu können, verwenden Sie die vertrauenswürdige Clientschnittstelle. In dieser Schnittstelle können Sie ei-

nen angepassten Nachrichtenhandler erstellen, um Ihre eigene Web-Service-Anforderung an den STS zu senden.

Vertrauenswürdige Clientschnittstelle

Die vertrauenswürdige Clientschnittstelle ermöglicht es Ihnen, direkt mit einem Sicherheitstokenservice (STS) statt mit dem Sicherheitshandler zu interagieren. Auf diese Weise haben Sie die Flexibilität, eine fortschrittlichere Verarbeitung von Token als die vom Sicherheitshandler angebotene zu nutzen.

Die vertrauenswürdige Clientschnittstelle ist eine Erweiterung des von CICS bereitgestellten Programms DFHPIRT. Dieses Programm wird üblicherweise verwendet, um eine Pipeline zu starten, wenn eine Web-Service-Requester-Anwendung nicht mithilfe des CICS-Web-Service-Assistenten implementiert wurde. Es kann aber auch als vertrauenswürdige Clientschnittstelle mit dem STS fungieren.

Sie können die vertrauenswürdige Clientschnittstelle aufrufen, indem Sie eine Verknüpfung mit DFHPIRT aus einem Nachrichtenhandler- oder Headerverarbeitungsprogramm herstellen, wobei Sie einen Kanal namens DFHWSTC-V1 und eine Gruppe von Sicherheitscontainern übergeben. Dank dieser Container haben Sie die Flexibilität, eine Validierungs- oder Ausgabeaktion vom STS anzufordern, den Typ des auszutauschenden Tokens auszuwählen und das entsprechende Token aus dem Nachrichtenheader zu übergeben. DFHPIRT erstellt dynamisch eine Pipeline, setzt eine Web-Service-Anforderung aus den Sicherheitscontainern zusammen und sendet sie an den STS.

DFHPIRT wartet auf die Antwort vom STS und übergibt diese im Container DFHWS-RESTOKEN zurück an den Nachrichtenhandler. Wenn der STS einen Fehler feststellt, gibt er einen SOAP-Fehler zurück. DFHPIRT stellt den Fehler in den Container DFHWS-STSFault und gibt ihn an das Verknüpfungsprogramm in der Pipeline zurück.

Sie können die vertrauenswürdige Clientschnittstelle verwenden, ohne den Sicherheitshandler in Ihren Service-Provider- und Service-Requester-Pipelines zu aktivieren, oder Sie können sie zusätzlich zum Sicherheitshandler einsetzen.

SOAP-Nachrichten unterzeichnen

Bei eingehenden Nachrichten unterstützt CICS digitale Signaturen für Elemente im SOAP-Hauptteil und in SOAP-Headerblöcken. Bei ausgehenden Nachrichten unterzeichnet CICS alle Elemente im SOAP-Hauptteil.

Eine SOAP-Nachricht ist ein XML-Dokument, das aus einem <Envelope>-Element besteht, das wiederum ein optionales <Header>-Element und ein obligatorisches <Body>-Element enthält.

Die Spezifikation 'WSS: SOAP Message Security' lässt zu, dass der Inhalt des <Header>-Elements und des <Body>-Elements auf Elementebene signiert werden. Das heißt, dass in einer bestimmten Nachricht einzelne Elemente signiert oder nicht signiert sein können bzw. dass sie mit unterschiedlichen Signaturen oder mithilfe unterschiedlicher Algorithmen signiert werden können. Beispiel: In einer SOAP-Nachricht, die in einer Onlineeinkaufsanwendung verwendet wird, ist es sinnvoll, Elemente zu signieren, die den Empfang einer Bestellung bestätigen, da diese Elemente Rechtscharakter haben können. Um jedoch den Systemaufwand für das Signieren der gesamten Nachricht zu vermeiden, können andere Informationen gefahrlos nicht signiert werden.

- Signierte Elemente im <Header>.
- Signierte Elemente im SOAP-Hauptteil (<Body>). Wenn der Handler so konfiguriert ist, dass er einen signierten Hauptteil erwartet, weist CICS alle SOAP-Nachrichten zurück, in denen der Hauptteil nicht signiert ist, und gibt einen SOAP-Fehler aus.

Signaturalgorithmen

Algorithmus	URI
Digital Signature Algorithm mit Secure Hash Algorithm 1 (DSA mit SHA1)	http://www.w3.org/2000/09/xmlsig#dsa-sha1
Nur für eingehende SOAP-Nachrichten unterstützt.	
Rivest-Shamir-Adleman-Algorithmus mit Secure Hash Algorithm 1 (RSA mit SHA1)	http://www.w3.org/2000/09/xmlsig#rsa-sha1

Dieses Beispiel zeigt eine SOAP-Nachricht, die von CICS signiert wurde.

Kapitel 4. Unterstützung für den Schutz von Web-Services 655

```

<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/> 2
<ds:DigestValue>QORZEA+gpafluShspHxhrrjaFlXE=</ds:DigestValue> 3
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>drDH0XESiyN6YJm27mfK1ZMG4Q4IsZqQ9N9V6kEnw2lk7aM3if77XNFnyKS4deglbC3ga11kkaFJ 4
p4jLOmYRqgycDPpqPm+UEu7mzfHRQGe7H0EnFqZpikNqZK5FF6fvYlv2JgTDPwrOSYXmhzwegUDT
1TVjOvuUgYrFya03pw=</ds:SignatureValue>

<ds:KeyInfo>
<wsse:SecurityTokenReference>
<wsse:Reference URI="#x509cert00"
Value="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"/> 5
</wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>
</wsse:Security>
</SOAP-ENV:Header>
<SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="TheBody">
<getVersion xmlns="http://msgsec.wssecfvt.ws.ibm.com"/>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

1. Das binäre Sicherheitstoken enthält die base64binary-Codierung des X.509-Zertifikats. Diese Codierung schließt den öffentlichen Schlüssel ein, den der gewünschte Empfänger der SOAP-Nachricht verwendet, um die Signatur zu verifizieren.
2. Der Algorithmus, der während des Hashing-Prozesses verwendet wird, um den Message-Digest zu generieren.
3. Der Wert des Message-Digest.
4. Der Digest-Wert wird dann mit dem privaten Schlüssel des Benutzers verschlüsselt und hier als Signaturwert eingeschlossen.
5. Referenziert das binäre Sicherheitstoken, das den öffentlichen Schlüssel enthält, mit dem die Signatur verifiziert wird.

CICS-Unterstützung für verschlüsselte SOAP-Nachrichten

Bei eingehenden Nachrichten kann CICS alle verschlüsselten Elemente im SOAP-Hauptteil entschlüsseln ebenso wie verschlüsselte SOAP-Headerblöcke, in denen der Hauptteil ebenfalls verschlüsselt ist. Bei ausgehenden Nachrichten verschlüsselt CICS den gesamten SOAP-Hauptteil.

Eine SOAP-Nachricht ist ein XML-Dokument, das aus einem <Envelope>-Element besteht, das wiederum ein optionales <Header>-Element und ein obligatorisches <Body>-Element enthält.

Die Spezifikation 'WSS: SOAP Message Security' ermöglicht, dass ein Teil des Inhalts des <Header>-Elements und der gesamte Inhalt des <Body>-Elements auf Elementebene verschlüsselt werden. Das heißt, dass in einer bestimmten Nachricht einzelne Elemente verschiedene Ebenen der Verschlüsselung aufweisen können und mithilfe unterschiedlicher Algorithmen verschlüsselt werden können. In einer SOAP-Nachricht, die in einer Onlineeinkaufsanwendung verwendet wird, ist es sinnvoll, die Kreditkartendetails einer Person zu verschlüsseln, um sicherzustellen, dass sie vertraulich bleiben. Um jedoch den Systemaufwand für die Verschlüsselung der gesamten Nachricht zu vermeiden, können einige Informationen mithilfe eines weniger sicheren (aber schnelleren) Algorithmus verschlüsselt werden, und andere Informationen können ganz unverschlüsselt bleiben.

Bei eingehenden Nachrichten kann der von CICS bereitgestellte Sicherheitsnachrichtenhandler einzelne Elemente im SOAP-Hauptteil (<Body>) und Elemente im SOAP-<Header> entschlüsseln, wenn der SOAP-Hauptteil ebenfalls verschlüsselt ist. Der Sicherheitsnachrichtenhandler entschlüsselt immer die folgenden Elemente:

- Elemente, die er im <Header>-Element findet, in der Reihenfolge, in der er sie findet.

- Bei ausgehenden Nachrichten unterstützt der Sicherheitsnachrichtenhandler nur die Verschlüsselung des Inhalts des SOAP-Hauptteils (<Body>). Er verschlüsselt keine Elemente im <Header>-Element. Wenn der Sicherheitsnachrichtenhandler das Element <Body> verschlüsselt, werden alle Elemente im Hauptteil mit demselben Algorithmus und demselben Schlüssel verschlüsselt. Der Algorithmus und die Informationen zu dem Schlüssel finden Sie in den Konfigurationsinformationen zum Handler.

CICS unterstützt die Verschlüsselungsalgorithmen, die für die XML Encryption-Spezifikation erforderlich sind. Jeder Algorithmus ist durch einen Universal Resource Identifier (URI) angegeben.

Algorithmus	URI
Triple Data Encryption Standard-Algorithmus (Triple DES)	http://www.w3.org/2001/04/xmlenc#tripledes-cbc
Advanced Encryption Standard-Algorithmus (AES) mit einer Schlüssellänge von 128 Bits	http://www.w3.org/2001/04/xmlenc#aes128-cbc
Advanced Encryption Standard-Algorithmus (AES) mit einer Schlüssellänge von 192 Bits	http://www.w3.org/2001/04/xmlenc#aes192-cbc
Advanced Encryption Standard-Algorithmus (AES) mit einer Schlüssellänge von 256 Bits	http://www.w3.org/2001/04/xmlenc#aes256-cbc

Dieses Beispiel einer SOAP-Nachricht wurde von CICS verschlüsselt.

Kapitel 4. Unterstützung für den Schutz von Web-Services 657

```

<wsse:SecurityTokenReference>
  <wsse:Reference URI="#x509cert00"
    ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"/> 3
</wsse:SecurityTokenReference>
</ds:KeyInfo>
<xenc:CipherData>
  <xenc:CipherValue>M6bDQtJrvX0pEjAEIcf6bq6MP3ySmB4TQ0a/B5U1Qj1vWjD56V+GRJbF7ZCES5ojwCjHRVKW1ZB5 4
    Mb+aUzSWlsoHzHQ1xc1JchgwCiyIn+E2TbG3R9m0zHD3XQsKTyVaOT1R7VPoMBd1ZLNDIomxjZn2
    p7JfxywXk0bcSLhdZnc=</xenc:CipherValue>
</xenc:CipherData>
<xenc:ReferenceList>
  <xenc:DataReference URI="#Enc1"/>
</xenc:ReferenceList>
</xenc:EncryptedKey>
</wsse:Security>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="Enc1" Type="http://www.w3.org/2001/04/xmlenc#Content">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/> 5
    <xenc:CipherData>
      <xenc:CipherValue>kgvqKnMcgIUn7r11vkFXF0g4SodEd3dxAJo/mVN6ef211B1MZe1g70yjEHf4ZXw1Cd0FebId1nK 6
        rrrksql1Mpw6So7ID8zav+KPQUKGm4+E=</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

1. Das binäre Sicherheitstoken enthält die base64binary-Codierung des X.509-Zertifikats. Diese Codierung enthält den öffentlichen Schlüssel, der zum Verschlüsseln des symmetrischen Schlüssels verwendet wurde.
2. Gibt den Algorithmus an, der zum Verschlüsseln des symmetrischen Schlüssels verwendet wurde.
3. Referenziert das binäre Sicherheitstoken, das den öffentlichen Schlüssel enthält, mit dem der symmetrische Schlüssel verschlüsselt wurde.
4. Der verschlüsselte symmetrische Schlüssel, der zum Verschlüsseln der Nachricht verwendet wurde.
5. Der Verschlüsselungsalgorithmus, der zum Verschlüsseln der Nachricht verwendet wurde.
6. Die verschlüsselte Nachricht.

RACF für Web Services Security konfigurieren

Sie müssen einen externen Sicherheitsmanager wie RACF konfigurieren, um öffentlich-private Schlüsselpaare und X.509-Zertifikate für das Signieren und Verschlüsseln ausgehender SOAP-Nachrichten und für das Authentifizieren und Entschlüsseln signierter und verschlüsselter eingehender SOAP-Nachrichten zu erstellen.

Before you begin

Bevor Sie diese Task ausführen, müssen Sie RACF für die Arbeit mit CICS eingerichtet haben. Geben Sie die Systeminitialisierungsparameter **DFLTUSER**, **KEYRING** und **SEC=YES** in der CICS-Region an, die Ihre Web-Service-Pipelines enthält.

Anmerkung: Mehrere Zertifikate mit demselben definierten Namen in demselben **KEYRING** werden nicht unterstützt.

Procedure

1. So authentifizieren Sie eingehende signierte SOAP-Nachrichten:
 - a. Importieren Sie das X.509-Zertifikat in RACF als ICSF-Schlüssel.
 - b. Hängen Sie das Zertifikat an den im Systeminitialisierungsparameter **KEYRING** angegebenen Schlüsselring mithilfe des Befehls **RADCERT** an.


```
RADCERT ID(userid1)
CONNECT(ID(userid2) LABEL('label-name') RING(ring-name))
```

Dabei gilt Folgendes:

- *userid1* ist die Standard-Benutzer-ID des Schlüsselrings oder ist berechtigt, Zertifikate an den Schlüsselring für andere Benutzer-IDs anzuhängen.
 - *userid2* ist die Benutzer-ID, die Sie dem Zertifikat zuordnen möchten.
 - *label-name* ist der Name des Zertifikats.
 - *ring-name* ist der Name des Schlüsselrings, der im Systeminitialisierungsparameter **KEYRING** angegeben ist.
- c. Optional: Wenn Sie zugesicherte Identitäten verwenden möchten, stellen Sie sicher, dass die dem Zertifikat zugeordnete Benutzer-ID über eine Ersatzberechtigung verfügt, um Arbeit unter anderen Benutzer-IDs ausführen lassen zu können. Stellen Sie außerdem sicher, dass alle zusätzlichen Zertifikate, die in den SOAP-Nachrichtenheader eingeschlossen sind, ebenfalls in RACF importiert werden.

Die SOAP-Nachricht kann ein binäres Sicherheitstoken im Header enthalten, das entweder das Zertifikat einschließt oder eine Referenz auf das Zertifikat enthält. Diese Referenz kann der KEYNAME sein (die Zertifikatsbezeichnung in RACF), eine Kombination von ISSUER und Seriennummer (SERIAL) oder der SubjectKeyIdentifier. CICS kann den SubjectKeyIdentifier nur erkennen, wenn er als Attribut in der Definition des Zertifikats in RACF angegeben wurde.

2. So signieren Sie ausgehende SOAP-Nachrichten:

- a. Erstellen Sie ein X.509-Zertifikat und ein öffentlich/privates Schlüsselpaar mithilfe des folgenden **RACDCERT**-Befehls:

```
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')
```

Dabei ist *userid2* die Benutzer-ID, die Sie dem Zertifikat zuordnen möchten. Verwenden Sie keines der folgenden Zeichen für den *label-name*-Wert des Zertifikats:

< > : ! =

- b. Hängen Sie das Zertifikat an den im Systeminitialisierungsparameter **KEYRING** angegebenen Schlüsselring an. Verwenden Sie den Befehl **RACDCERT**.
- c. Exportieren Sie das Zertifikat und veröffentlichen Sie es für den vorgesehenen Empfänger der SOAP-Nachricht.

Sie können die Pipelinekonfigurationsdatei so bearbeiten, dass CICS automatisch das X.509-Zertifikat im binären Sicherheitstoken des SOAP-Nachrichtenheaders für den beabsichtigten Empfänger enthält, um die Signatur zu validieren.

3. Um eingehende verschlüsselte SOAP-Nachrichten zu entschlüsseln, muss die SOAP-Nachricht den öffentlichen Schlüssel enthalten, der Teil eines Schlüsselpaars ist, in dem der private Schlüssel in CICS definiert ist.
- a. Generieren Sie ein öffentlich/privates Schlüsselpaar und ein Zertifikat in RACF für die Verschlüsselung. Das Schlüsselpaar und das Zertifikat müssen mithilfe von ICSF generiert werden.
- b. Hängen Sie das Zertifikat an den im Systeminitialisierungsparameter **KEYRING** angegebenen Schlüsselring an. Verwenden Sie den Befehl **RACDCERT**.
- c. Exportieren Sie das Zertifikat und veröffentlichen Sie es für den Generator der SOAP-Nachrichten, die Sie entschlüsseln möchten.

Der Generator der SOAP-Nachricht kann dann das Zertifikat importieren, das den öffentlichen Schlüssel enthält, und es zum Verschlüsseln der SOAP-Nachricht verwenden. Die SOAP-Nachricht kann ein binäres Sicherheitstoken im Header enthalten, das entweder den öffentlichen Schlüssel einschließt oder eine Referenz darauf enthält. Diese Referenz kann der KEYNAME, eine Kombination von ISSUER und Seriennummer (SERIAL) oder der SubjectKeyIdentifier sein. CICS kann den SubjectKeyIdentifier nur erkennen, wenn er als Attribut in der Definition des öffentlichen Schlüssels in RACF angegeben wurde.

4. So verschlüsseln Sie ausgehende SOAP-Nachrichten:
 - a. Importieren Sie das Zertifikat, das den öffentlichen Schlüssel enthält, den Sie für die Verschlüsselung verwenden möchten, als ICSF-Schlüssel in RACF. Für den beabsichtigten Empfänger muss der private Schlüssel dem öffentlichen Schlüssel zugeordnet sein, damit die SOAP-Nachricht entschlüsselt werden kann.
 - b. Hängen Sie das Zertifikat an, das den öffentlichen Schlüssel des Schlüsselrings enthält, der im Systeminitialisierungsparameter **KEYRING** angegeben ist. Verwenden Sie den Befehl **RACDCERT**.

CICS verwendet den öffentlichen Schlüssel im Zertifikat, um den SOAP-Hauptteil zu verschlüsseln, und sendet das Zertifikat, das den öffentlichen Schlüssel enthält, als binäres Sicherheitstoken im SOAP-Nachrichtenheader. Der öffentliche Schlüssel ist in der Pipelinekonfigurationsdatei definiert.

What to do next

Diese Konfiguration zum Signieren und Verschlüsseln von ausgehenden Nachrichten setzt voraus, dass das verwendete Zertifikat der Benutzer-ID der CICS-Region gehört. Das Zertifikat muss Eigentum der Benutzer-ID der CICS-Region sein, da RACF nur dem Zertifikatseigner erlaubt, den privaten Schlüssel zu extrahieren, der zum Signieren oder Verschlüsseln verwendet wird.

Wenn CICS eine Nachricht mithilfe eines Zertifikats signieren oder verschlüsseln muss, dessen Eigner es nicht ist, kann ein einzelnes Zertifikat gemeinsam von CICS-Systemen verwendet werden, indem Sie die Anweisungen unter Using an existing certificate that is not owned by the CICS region user ID befolgen.

Web-Services im Providermodus für die Weitergabe von Identitäten konfigurieren

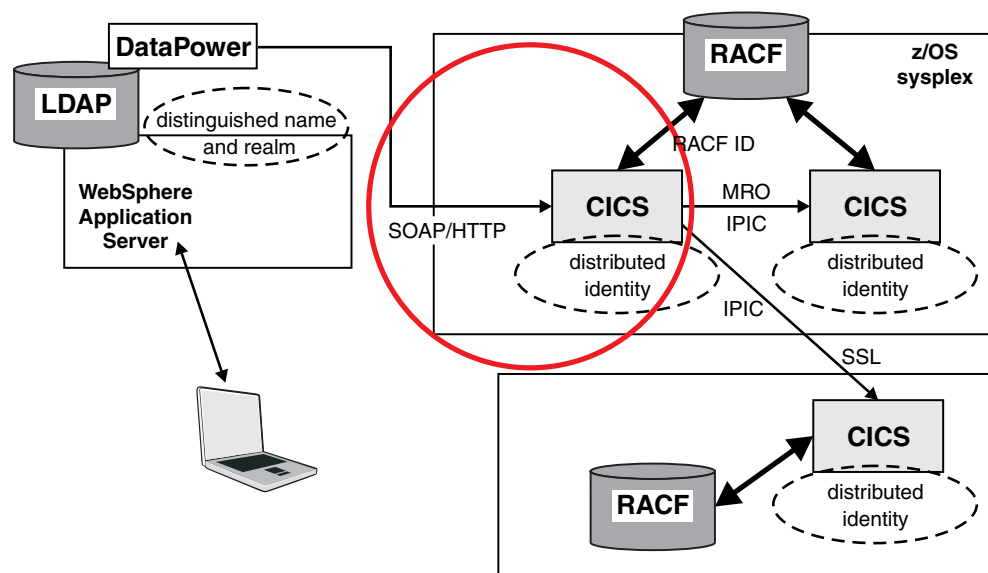
Die Weitergabe von Identitäten mit einer Web-Service-Anforderung hängt von vertrauensbasierten Konfigurationen ab, beispielsweise von der Verwendung einer clientzertifizierten SSL-Verbindung von WebSphere DataPower. In dieser Task konfigurieren Sie eine PIPELINE-Ressource so, dass sie im WS-Security-Header ein ICRX-Identitätstoken erwartet, das von einem vertrauenswürdigen Client gesendet wurde.

Before you begin

Sie müssen Ihre RACF-RACMAP-Einstellungen vor Ihren Web-Service-Verbindungen konfigurieren, andernfalls empfangen Sie die RACF-Nachricht ICH408I für jede nicht zugeordnete Anforderung, die an RACF gesendet wird. Weitere Informationen zur Konfiguration des RACF-Befehls **RACMAP** finden Sie unter RACF für die Identitätsweitergabe konfigurieren.

About this task

Abbildung 29. CICS für den Empfang eines ICRX-Identitätstokens aus WebSphere DataPower konfigurieren.



CICS empfängt die SOAP-Nachricht von WebSphere DataPower. Die PIPELINE-Konfigurationsdatei gibt eine *blinde* Vertrauensbeziehung an, weil der einzige mögliche Client die WebSphere DataPower-Appliance ist und weil WebSphere DataPower mit CICS über eine sichere SSL-Verbindung kommuniziert. Deshalb müssen Sie keine zusätzliche Authentifizierung in der PIPELINE-Konfigurationsdatei angeben.

Das WS-Security-Handlerprogramm sucht nach dem ersten ICRX-Element im WS-Security-Header und identifiziert damit den Benutzer.

Procedure

1. Erstellen Sie eine PIPELINE-Ressource oder bearbeiten Sie eine vorhandene PIPELINE-Ressource, um den basic-ICRX-Modus anzugeben, der zulässt, dass die PIPELINE ein ICRX-Element empfängt. Die gängigste Kombination ist die blinde Vertrauensbeziehung mit dem basic-ICRX-Modus. Weitere Informationen zum PIPELINE-Ressourcenelement finden Sie unter The <authentication> element.

Dies ist ein Beispiel einer PIPELINE-Konfigurationsdatei mit blinder Vertrauensbeziehung und dem basic-ICRX-Modus:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <service_handler_list>
      <wsse_handler>
        <dfhwsse_configuration version="1">
          <authentication trust="blind" mode="basic-ICRX"/>
        </dfhwsse_configuration>
      </wsse_handler>
    </service_handler_list>
    <terminal_handler>
      <cics_soap_1.2_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

Dies ist ein Beispiel einer SOAP-Nachricht mit einer ICRX-Identität und blinder Vertrauensbeziehung:

```
<?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. Stellen Sie sicher, dass WebSphere DataPower für das Senden von ICRX-Informationen konfiguriert ist. Siehe Beispielnetworktopologien für die Verwendung der Identitätsweitergabe.

Results

Web-Service-Anforderungen von WebSphere DataPower mit einem ICRX-Identitätstoken im WS-Security-Header, die über eine clientzertifizierte SSL-Verbindung verbunden sind, können nun weitergegeben werden.

Pipeline für Web Services Security konfigurieren

Um eine Pipeline für die Unterstützung von Web Services Security (WSS) zu konfigurieren, müssen Sie einen Sicherheitshandler zu Ihren Pipelinekonfigurationsdateien hinzufügen. Sie können den mit CICS bereitgestellten Sicherheitshandler wie beschrieben verwenden oder Ihren eigenen erstellen.

Before you begin

Bevor Sie den von CICS bereitgestellten Sicherheitshandler definieren, müssen Sie die Pipelinekonfigurationsdateien identifizieren oder erstellen, in denen Sie Konfigurationsinformationen für WSS hinzufügen.

Procedure

1. Fügen Sie ein `<wsse_handler>`-Element zu Ihrer Pipeline hinzu. Der Handler muss im Element `<service_handler_list>` in einer Service-Provider- oder Service-Requester-Pipeline enthalten sein. Codieren Sie die folgenden Elemente:

```
<wsse_handler>
  <dfhwsse_configuration version="1">

    </dfhwsse_configuration>
  </wsse_handler>
```

Das `<dfhwsse_configuration>`-Element ist ein Container für die anderen Elemente in der Konfiguration.

2. Optional: Codieren Sie ein `<authentication>`-Element.
 - In einer Service-Requester-Pipeline gibt das Element `<authentication>` den Typ von Authentifizierung an, der im Sicherheitsheader von ausgehenden SOAP-Nachrichten verwendet werden muss.
 - In einer Service-Provider-Pipeline gibt das Element an, ob CICS die Sicherheitstoken in einer eingehenden SOAP-Nachricht verwendet, um die Benutzer-ID zu ermitteln, unter der die Verarbeitung stattfindet.
- a. Codieren Sie das Attribut **trust** so, dass es angibt, ob die zugesicherte Identität verwendet wird, ebenso wie die Art der Vertrauensbeziehung zwischen dem Service-Provider und dem Service-Requester. Details zum Attribut **trust** finden Sie unter The `<authentication>` element.
- b. Optional: Wenn Sie **trust=none** angegeben haben, codieren Sie das Attribut **mode** so, dass es angibt, wie in der Nachricht gefundene Berechtigungsnachweise verarbeitet werden. Details zum Attribut **mode** finden Sie unter The `<authentication>` element.
- c. Codieren Sie im Element `<authentication>` die folgenden Elemente:
 - 1) Ein optionales leeres `<suppress/>`-Element.

Wenn dieses Element in einer Service-Provider-Pipeline angegeben ist, versucht der Handler nicht, anhand von Sicherheitstoken in der Nachricht zu bestimmen, welche Benutzer-ID aktiv ist.

Wenn dieses Element in einer Service-Requester-Pipeline angegeben ist, versucht der Handler nicht, Sicherheitstoken, die für die Authentifizierung erforderlich sind, zu der ausgehenden SOAP-Nachricht hinzuzufügen.

- 2) In einer Requester-Pipeline ein optionales `<algorithm>`-Element, das den URI des Algorithmus angibt, der zum Signieren des Hauptteils der SOAP-Nachricht verwendet wird. Sie müssen dieses Element angeben, wenn die Kombination von 'trust' und 'mode'-Attributen angibt, dass die Nachrichten signiert sind. Sie können nur den 'RSA mit SHA1'-Algorithmus in diesem Element angeben. Der URI ist <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.
- 3) Ein optionales `<certificate_label>`-Element, das die Bezeichnung angibt, die einem digitalen X.509-Zertifikat zugeordnet ist, das in RACF installiert ist. Wenn Sie dieses Element in einer Service-Requester-Pipeline angeben und das `<suppress>`-Element ist nicht angegeben, wird das Zertifikat zum Sicherheitsheader in der SOAP-Nachricht hinzugefügt. Wenn Sie kein `<certificate_label>`-Element angeben, verwendet CICS das Standardzertifikat im RACF-Schlüsselring.

Dieses Element wird in einer Service-Provider-Pipeline ignoriert.

3. Optional: Codieren Sie ein Element `<sts_authentication>` als Alternative zum Element `<authentication>`. Sie dürfen in Ihrer Pipelinekonfigurationsdatei nicht beide codieren. Dieses Element gibt an, dass ein Sicherheitstokenservice (STS) für die Authentifizierung verwendet wird, und bestimmt die Art von Anforderung, die gesendet wird.

- a. Optional: Codieren Sie nur im Service-Providermodus das Attribut **action**, um anzugeben, ob der STS ein Sicherheitstoken verifiziert oder austauscht. Details zum Attribut **action** finden Sie unter The `<sts_authentication>` element.

- b. Codieren Sie im Element `<sts_authentication>` diese Elemente:

- 1) Ein Element `<auth_token_type>`. Dieses Element ist erforderlich, wenn Sie das Element `<sts_authentication>` in einer Service-Requester-Pipeline angeben, und optional in einer Service-Provider-Pipeline. Weitere Informationen finden Sie unter The `<auth_token_type>` element.

- In einer Service-Requester-Pipeline gibt das `<auth_token_type>`-Element den Typ des Tokens an, das der STS ausgibt, wenn CICS die Benutzer-ID sendet, die im Container DFHWS-USERID enthalten ist. Das Token, das CICS von dem STS empfängt, wird in dem Header der ausgehenden Nachricht platziert.
- In einer Service-Provider-Pipeline wird das `<auth_token_type>`-Element verwendet, um das Identitätstoken zu bestimmen, das CICS dem Nachrichtenheader entnimmt und zwecks Austausch oder Validierung an den STS sendet. CICS verwendet das erste Identitätstoken des angegebenen Typs im Nachrichtenheader. Wenn Sie dieses Element nicht angeben, verwendet CICS das erste Identitätstoken, das es im Nachrichtenheader findet. Folgendes zieht CICS nicht als Identitätstoken in Betracht:

- `wsu:Timestamp`
- `xenc:ReferenceList`
- `xenc:EncryptedKey`
- `ds:Signature`

- 2) Ein optionales leeres Element `<suppress/>` (nur in einer Service-Provider-Pipeline). Wenn dieses Element angegeben ist, versucht der Hand-

ler nicht, anhand von Sicherheitstoken in der Nachricht zu bestimmen, welche Benutzer-ID aktiv ist. Das Element `<suppress/>` schließt das Identitätstoken ein, das von dem STS zurückgegeben wird.

4. Optional: Codieren Sie ein Element `<sts_endpoint>`. Verwenden Sie dieses Element nur, wenn Sie auch ein Element `<sts_authentication>` angegeben haben. Codieren Sie im Element `<sts_endpoint>` die folgenden Elemente:

- Ein Element `<endpoint>`. Dieses Element enthält einen URI, der auf die Position des Sicherheitstokenservice (STS) im Netz zeigt. Es wird empfohlen, statt HTTP SSL oder TLS zu verwenden, um die Verbindung zum STS zu sichern.

Um die SAML-Unterstützung zu aktivieren, legen Sie den Endpunkt auf `cics://PROGRAM/DFHSAML` fest.

Sie können mithilfe des JMS-Formats des URI auch einen WebSphere MQ-Endpunkt angeben.

- Ein optionales Element `<jvmserver>`. Dieses Element gibt den JVM-Server an, der für die Ausführung des SAML-Token-Service konfiguriert ist. Ist dieses Element nicht angegeben, wird standardmäßig der JVM-Server der Beispielressource, DFHXSTS, angenommen. Dieses Element ist nur gültig, wenn SAML verwendet wird. In allen anderen Fällen tritt ein Fehler auf.

5. Optional: Wenn eingehende SOAP-Nachrichten digital signiert werden sollen, codieren Sie ein leeres Element `<expect_signed_body/>`.

Das Element `<expect_signed_body/>` gibt an, dass der Hauptteil (`<body>`) der eingehenden Nachricht signiert sein muss. Wenn der Hauptteil einer eingehenden Nachricht nicht korrekt signiert ist, weist CICS die Nachricht mit einem Sicherheitsfehler zurück.

6. Optional: Wenn eingehende SOAP-Nachrichten, die digital signiert sind, zurückgewiesen werden sollen, codieren Sie ein leeres Element `<reject_signature/>`.

7. Optional: Wenn eingehende SOAP-Nachrichten verschlüsselt werden sollen, codieren Sie ein leeres Element `<expect_encrypted_body/>`.

Das Element `<expect_encrypted_body/>` gibt an, dass der Hauptteil (`<body>`) der eingehenden Nachricht verschlüsselt sein muss. Wenn der Hauptteil einer eingehenden Nachricht nicht korrekt verschlüsselt ist, weist CICS die Nachricht mit einem Sicherheitsfehler zurück.

8. Wenn eingehende SOAP-Nachrichten, die partiell oder vollständig verschlüsselt sind, zurückgewiesen werden sollen, codieren Sie ein leeres Element `<reject_encryption/>`.

9. Optional: Wenn ausgehende SOAP-Nachrichten signiert werden sollen, codieren Sie ein Element `<sign_body>`.

- a. Codieren Sie im Element `<sign_body>` ein Element `<algorithm>`.
- b. Codieren Sie im Anschluss an das Element `<algorithm>` ein Element `<certificate_label>`.

Im Folgenden sehen Sie ein Beispiel eines vollständigen Elements `<sign_body>`:

```
<sign_body>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>SIGCERT01</certificate_label>
</sign_body>
```

10. Optional: Wenn ausgehende SOAP-Nachrichten verschlüsselt werden sollen, codieren Sie ein Element `<encrypt_body>`.

- a. Codieren Sie im Element `<encrypt_body>` ein Element `<algorithm>`.

- b. Codieren Sie im Anschluss an das Element <algorithm> ein Element <certificate_label>.

Im Folgenden sehen Sie ein Beispiel eines vollständigen Elements <encrypt_body>:

```
<encrypt_body>
  <algorithm>http://www.w3.org/2001/04/xmlenc#tripledes-cbc</algorithm>
  <certificate_label>ENCCERT02</certificate_label>
</encrypt_body>
```

Beispiel

Das folgende Beispiel zeigt einen vollständigen Sicherheitshandler, in dem ein Großteil der optionalen Elemente vorhanden ist.

```
<wsse_handler>
  <dfwsse_configuration version="1">
    <authentication trust="signature" mode="basic">
      <suppress/>
      <certificate_label>AUTHCERT03</certificate_label>
    </authentication>
    <expect_signed_body/>
    <expect_encrypted_body/>
    <sign_body>
      <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
      <certificate_label>SIGCERT01</certificate_label>
    </sign_body>
    <encrypt_body>
      <algorithm>http://www.w3.org/2001/04/xmlenc#tripledes-cbc</algorithm>
      <certificate_label>ENCCERT02</certificate_label>
    </encrypt_body>
  </dfwsse_configuration>
</wsse_handler>
```

Angepassten Sicherheitshandler schreiben

Wenn Sie Ihre eigenen Sicherheitsprozeduren und -prozesse verwenden wollen, können Sie einen angepassten Nachrichtenhandler schreiben, um sichere SOAP-Nachrichten in der Pipeline zu verarbeiten.

Before you begin

Sie müssen die Sicherheitsstufe festlegen, die Ihr Sicherheitshandler unterstützen muss. Und Sie müssen sicherstellen, dass ein entsprechender SOAP-Fehler zurückgegeben wird, wenn eine Nachricht eine nicht unterstützte Sicherheitsstufe aufweist.

About this task

Der Nachrichtenhandler muss außerdem in der Lage sein, die Sicherheit für eingehende und ausgehende Nachrichten zu organisieren.

Procedure

1. Rufen Sie den Container DFHREQUEST oder DFHRESPONSE mithilfe eines Befehls **EXEC CICS GET CONTAINER** ab.
2. Parsen Sie die XML, um das Sicherheitstoken im WS-Security-Nachrichtenheader zu finden. Der Header beginnt mit dem Element <wsse:Security>. Bei dem Sicherheitstoken kann es sich um einen Benutzernamen und ein Kennwort, ein

digitales Zertifikat oder einen Verschlüsselungsschlüssel handeln. Eine Nachricht kann viele Token im Sicherheitsheader enthalten, deshalb muss Ihr Handler das richtige identifizieren können.

3. Führen Sie die entsprechende Verarbeitung aus, abhängig von der in der Nachricht implementierten Sicherheit.
 - a. Wenn Sie eine Basisauthentifizierung eines Kerberos-Token ausführen möchten, geben Sie einen Befehl **EXEC CICS VERIFY TOKEN** aus. Dieser Befehl prüft, ob das angegebene Kerberos-Token gültig ist. Wenn der Befehl erfolgreich ist, aktualisieren Sie den Container DFHWS-USERID mithilfe des Befehls **EXEC CICS PUT CONTAINER**. Geben Sie andernfalls einen Befehl **EXEC CICS SOAPFAULT CREATE** aus.
 - b. Wenn Sie eine Basisauthentifizierung eines Kennworts oder einer Kennwortphrase ausführen möchten, geben Sie einen Befehl **EXEC CICS VERIFY PHRASE** aus. Dieser Befehl prüft den Benutzernamen und das Kennwort im Sicherheitsheader der Nachricht. Wenn der Befehl erfolgreich ist, aktualisieren Sie den Container DFHWS-USERID mithilfe des Befehls **EXEC CICS PUT CONTAINER**. Geben Sie andernfalls einen Befehl **EXEC CICS SOAPFAULT CREATE** aus.
 - c. Wenn Sie eine erweiterte Authentifizierung ausführen möchten, indem Sie eine Reihe von Token mithilfe eines Sicherheitstokenservice austauschen oder auswerten, verwenden Sie die vertrauenswürdige Clientschnittstelle. Ausführliche Informationen finden Sie unter „Vertrauenswürdigen Client aus einem Nachrichtenhandler aufrufen“.
 - d. Validieren Sie die Berechtigungsnachweise des digitalen Zertifikats, wenn die Nachricht signiert ist.
 - e. Wenn Teile der Nachricht verschlüsselt sind, entschlüsseln Sie die Nachricht unter Zuhilfenahme der Informationen im Sicherheitsheader. Die How CICS complies with Web Services Security specifications-Spezifikation enthält Informationen zur Vorgehensweise.

Results

Definieren Sie Ihr Sicherheitshandlerprogramm in CICS und aktualisieren Sie die Pipelinekonfigurationsdatei, wobei Sie sicherstellen, dass sie in der XML korrekt platziert wird. In einer Service-Requester-Pipelinekonfigurationsdatei muss der Sicherheitshandler so konfiguriert sein, dass er am Ende der Pipeline ausgeführt wird. In einer Service-Provider-Pipelinekonfigurationsdatei muss der Sicherheitshandler so konfiguriert sein, dass er am Anfang der Pipeline ausgeführt wird.

What to do next

Allgemeine Informationen zum Schreiben eines angepassten Nachrichtenhandlers finden Sie unter IBM Redbooks: Application Development for IBM CICS Web Services.

Vertrauenswürdigen Client aus einem Nachrichtenhandler aufrufen

CICS stellt eine Schnittstelle bereit, damit Sie Ihren eigenen Nachrichtenhandler schreiben können, um einen Sicherheitstokenservice (STS) aufzurufen. Über diese Schnittstelle können Sie eine komplexere Verarbeitung ausführen als die von CICS bereitgestellten Sicherheitshandler.

About this task

Sie können den vertrauenswürdigen Client anstelle des Sicherheitshandlers oder zusätzlich zu diesem verwenden. Gehen Sie wie folgt vor, um die vertrauenswürdige Clientschnittstelle zu verwenden:

Procedure

1. Extrahieren Sie das korrekte Token aus dem Sicherheitsnachrichtenheader der eingehenden oder ausgehenden Nachricht.
2. Stellen Sie eine Verknüpfung zum Programm DFHPIRT her und übergeben Sie den Kanal DFHWSTC-V1 und die folgenden erforderlichen Container:
 - DFHWS-STSTOKEN - enthält die Position des STS im Netz.
 - DFHWS-STSACTION - enthält den URI des Typs von Anforderung, die der STS ausführen muss. Die beiden unterstützten Aktionen sind Absetzen und Überprüfen.
 - DFHWS-IDTOKEN - enthält das Token, das vom STS entweder überprüft noch ausgetauscht werden muss.
 - DFHWS-TOKENTYPE - enthält den Typ von Token, das der STS in der Antwort zurücksenden muss.
 - DFHWS-SERVICEURI - enthält den URI der Web-Service-Operation, die aufgerufen wird.

Sie können optional den Container DFHWS-XMLNS einschließen, um die Namensbereiche der SOAP-Nachricht bereitzustellen, die das Sicherheitstoken enthält. Dieser Container wird genauer unter The header processing program interface beschrieben.

3. DFHPIRT kehrt mit der Antwort vom STS zurück. Eine erfolgreiche Antwort wird im Container DFHWS-RESTOKEN gespeichert.

Wenn der STS ein Problem mit der Anforderung feststellt, gibt er einen SOAP-Fehler zurück. DFHPIRT stellt den SOAP-Fehler in den Container DFHWS-STSTOKEN. Wenn der STS einen Grund für die Ausgabe des SOAP-Fehlers angibt, wird dieser Grund in den Container DFHWS-STSTOKEN gestellt.

Wenn eine abnormale Beendigung auftritt, wird ein DFHERROR-Container zurückgegeben, der Details des Verarbeitungsfehlers enthält.

Ihr Nachrichtenhandler muss diese Antworten verarbeiten und führt eine geeignete Verarbeitung im Fall eines Fehlers aus. Beispielsweise kann der Nachrichtenhandler einen geeigneten SOAP-Fehler an den Web-Service-Requester zurückgeben.

4. Verarbeiten Sie die Antwort nach Bedarf. Im Providermodus muss Ihre Pipelineverarbeitung sicherstellen, dass ein Benutzername, den CICS verstehen kann, in den Container DFHWS-USERID gestellt wird, wenn die Nachricht den Anwendungshandler erreicht. Im Requestermodus muss Ihr Nachrichtenhandler das richtige Token zum Sicherheitsheader für ausgehende Nachrichten hinzufügen.

What to do next

Wenn Sie Ihren Nachrichtenhandler geschrieben haben, implementieren Sie das Programm in CICS und aktualisieren Sie entsprechenden Pipelinekonfigurationsdateien. Definieren Sie in Service-Requester-Pipelines Ihren Nachrichtenhandler am Ende der Pipeline-Verarbeitung, jedoch vor dem von CICS bereitgestellten Sicherheitshandler. Definieren Sie in Service-Provider-Pipelines Ihren Nachrichtenhandler

am Anfang der Pipeline, aber nach dem von CICS bereitgestellten Sicherheitshandler.

Sicherheit für z/OS Connect

z/OS Connect ist eine WebSphere Liberty-Anwendung, für die dieselbe Konfiguration und dieselben Überlegungen gelten wie für andere WebSphere Liberty-Anwendungen. Darüber hinaus haben z/OS Connect for CICS 1.0 und z/OS Connect Enterprise Edition zusätzliche Sicherheitsanforderungen.

z/OS Connect for CICS 1.0 und z/OS Connect Enterprise Edition verfügen über eine REST-konforme Managementschnittstelle für eine dynamische Serviceerkennung. Diese Schnittstelle wird mit demselben Hostnamen und derselben Portnummer wie die einzelnen JSON-Services gehostet. Die Verwendung von Transport Layer Security (TLS) zum Schutz dieser Schnittstelle und der einzelnen JSON-Services wird unterstützt.

Standardmäßig müssen alle Clientverbindungen mit z/OS Connect for CICS 1.0 und z/OS Connect Enterprise Edition das HTTPS-Protokoll verwenden. Standardmäßig wird eine clientertifizierte TLS-Verbindung mit CICS vorausgesetzt. Wird dieses Standardverhalten beibehalten, müssen Clientzertifikate einer SAF-Benutzer-ID zugeordnet werden. Die Anwendung wird unter Verwendung dieser aus dem Zertifikat abgeleiteten Identität ausgeführt.

Sowohl z/OS Connect for CICS 1.0 als auch z/OS Connect Enterprise Edition können so konfiguriert werden, dass sie das HTTP-Basisauthentifizierungsprotokoll unterstützen. Dieses Protokoll ermöglicht einem Client, eine Verbindung herzustellen, indem TLS in Kombination mit einer SAF-Benutzer-ID und einem Kennwort verwendet wird. Um die Unterstützung für die HTTP-Basisauthentifizierung zu aktivieren, fügen Sie die folgende Zeile zur Liberty-Serverkonfigurationsdatei (server.xml) hinzu: `<webAppSecurity allowFailOverToBasicAuth="true"/>`

Benutzer von z/OS Connect for CICS 1.0 und z/OS Connect Enterprise Edition müssen ein Member der EJB-Rolle `zos.connect.access.roles.zosConnectAccess` sein. Weitere Informationen finden Sie unter Authorization using SAF role mapping.

Informationen zu Liberty, Configuring security for z/OS Connect for z/OS Connect und Configuring security for z/OS Connect EE in z/OS Connect Enterprise Edition V3.0 product documentation for z/OS Connect EE finden Sie unter Configuring authorization for applications in Liberty.

Weitere Informationen finden Sie unter Authorization using SAF role mapping und Configuring security for a Liberty JVM server.

Berechtigungen für z/OS Connect-Services und -APIs konfigurieren

Das CICS-Sicherheitsmodell erfordert bestimmte zusätzliche Aktionen zum Konfigurieren von Berechtigungen für Services und APIs mit z/OS Connect for CICS 1.0 und z/OS Connect Enterprise Edition.

About this task

Wenn z/OS Connect verwendet wird, um Arbeit in CICS einzufügen, werden die folgenden beiden Identitäten der Arbeit in verschiedenen Phasen der Verarbeitung zugeordnet.

- Eine ursprüngliche, temporäre Identität wird während des Anhängens der Arbeit zugeordnet.
- Anschließend wird eine authentifizierte Identität verwendet, um den Rest der Arbeit auszuführen.

Sie haben verschiedene Möglichkeiten, diese Identitäten zu konfigurieren, abhängig von Ihren Vorgaben und Ihrer Systemumgebung.

Procedure

1. Optional: Erstellen Sie eine alternative ursprüngliche Benutzer-ID für z/OS Connect.

Standardmäßig ist die ursprüngliche Identität die standardmäßige CICS-Benutzer-ID. Sie können jedoch eine andere Benutzer-ID zuweisen, wenn Sie der standardmäßigen CICS-Benutzer-ID keine Berechtigung zur Ausführung der Transaktion CPIH oder einer äquivalenten Transaktion erteilen möchten.

- a. Erteilen Sie der alternativen ursprünglichen Benutzer-ID die Berechtigung zur Ausführung der Transaktion CPIH und aller weiteren Transaktionen, die über z/OS Connect initiiert werden.

Die ursprüngliche Benutzer-ID muss über die Berechtigung zur Ausführung der Zieltransaktion für den Service oder die API verfügen.

2. Weisen Sie eine standardmäßige ursprüngliche Benutzer-ID zu. Sie können eine der beiden folgenden Methoden auswählen:

- Legen Sie einen Benutzer-ID-Überschreibungswert im JVM-Profil für die JVMSERVER-Ressource fest, die z/OS Connect hostet.

Im Folgenden finden Sie ein Beispiel für eine Überschreibung, wobei ZOSCUSER die standardmäßige ursprüngliche Benutzer-ID ist:

```
-Dcom.ibm.cics.jvmserver.http.userid=ZOSCUSER
```

Anmerkung: Wenn Sie eine standardmäßige ursprüngliche Benutzer-ID im JVM-Profil festlegen, müssen Sie keinen USERID-Wert für die einzelnen URIMAP-Instanzen bereitstellen. Wenn Sie jedoch sowohl eine USERID für eine URIMAP als auch einen Überschreibungswert im JVM-Profil bereitstellen, hat die USERID, die für eine bestimmte URIMAP angegeben wurde, Vorrang.

- Legen Sie das Feld USERID für eine gegebene URIMAP-Ressource fest, deren Ziel z/OS Connect ist.

Wenn eine HTTP-Anforderung von z/OS Connect empfangen wird, gleicht CICS sie mit den installierten URIMAP-Ressourcen ab. Wenn die URIMAP, die bei diesem Abgleich gefunden wird, das USERID-Attribut angibt, wird diese Benutzer-ID statt der standardmäßigen ursprünglichen Benutzer-ID für den JVM-Server als ursprüngliche Benutzer-ID verwendet.

Dies ist eine Beispielkonfiguration für eine URIMAP-Ressource namens ZOSCDEFT, wobei der USAGE-Wert JVMSERVER lautet, ein generischer Wert für das PATH-Attribut festgelegt ist, CPIH die Zieltransaktion ist und ZOSCUSER die standardmäßige ursprüngliche Benutzer-ID ist:

```
NAME: ZOSCDEFT
USAGE: JVMSERVER
SCHEME: HTTP
PORT: NO
```

```
HOST: *  
PATH: /zosConnect/*  
TRANSACTION: CPIH  
USERID: ZOSCUSER
```

Anmerkung: URIMAP-Ressourcen, die mithilfe des PIPELINE SCAN-Mechanismus installiert werden, sind wahrscheinlich nicht mit einer standardmäßigen Benutzer-ID konfiguriert. In diesem Szenario können Sie in Betracht ziehen, einen Benutzer-ID-Überschreibungswert in JVMSERVER anzugeben.

Anmerkung: Es ist möglich, eine ursprüngliche Benutzer-ID in einer WSBind-Datei zu speichern: Der Benutzer von DFHLS2JS oder DFHJS2LS kann einen Wert für den **USERID**-Eingabeparameter bereitstellen. Wenn der **USERID**-Parameter verwendet wird, enthalten alle URIMAPs, die während eines Pipeline-Scans erzeugt werden, die angeforderte ursprüngliche Benutzer-ID.

Results

Sie haben Ihre Umgebung jetzt so konfiguriert, dass CICS die URIs für Ihre Services und APIs erkennt und einer ursprünglichen Benutzer-ID zuordnet, damit sie verwendet werden, wenn die Zieltransaktion angehängt wird.

Kapitel 5. Fehlerbehebung bei Web-Services

Die möglichen Probleme bei der Implementierung von Web-Services in CICS können während der Implementierung selbst oder während der Ausführung auftreten, wenn CICS die Nachrichten umsetzt.

Fehlerbehebung bei SOAP-Web-Services

Die möglichen Probleme bei der Implementierung von SOAP-Web-Services in CICS können während der Implementierung selbst oder während der Ausführung auftreten, wenn CICS SOAP-Nachrichten umsetzt.

Implementierungsfehler diagnostizieren

Implementierungsfehler können auftreten, wenn Sie versuchen, die Stapeljobs des CICS-Web-Service-Assistenten oder des CICS-XML-Assistenten auszuführen, eine PIPELINE-Ressource in CICS zu installieren oder eine WEBSERVICE-Ressource in CICS zu installieren. Die gängigsten Implementierungsfehler werden nachfolgend beschrieben, einschließlich Symptom, Ursache und Lösung des Problems.

About this task

Im Falle eines Implementierungsfehlers werden PIPELINE-Ressourcen in der Regel mit einem DISABLED-Status und WEBSERVICE-Ressourcen mit einem UNUSABLE-Status installiert.

Informationen und Fehlernachrichten, die den Stapeljobs des CICS-Web-Service-Assistenten und des CICS-XML-Assistenten zugeordnet sind, werden im Jobprotokoll festgehalten. Fehlernachrichten, die der Installation von Ressourcen zugeordnet sind, werden im Systemprotokoll aufgezeichnet.

Die Codes 0, 4, 8 oder 12 werden von den Assistenten ausgegeben, andere Codes stammen üblicherweise von BPXBATCH, der JVM oder IEBGENER.

Von BPXBATCH ausgegebene Codes fallen in zwei Hauptkategorien: ein Code kleiner 128 weist auf das Fehlschlagen eines Befehls hin, ein Code größer als 128 weist darauf hin, dass der Prozess durch ein Signal beendet wurde. Weitere Informationen zu BPXBATCH und den zugehörigen Rückgabecodes finden Sie in der z/OS UNIX System Services Command Reference.

Procedure

- Sie empfangen den Rückgabecode 0, 4, 8 oder 12, wenn Sie die Stapeljobs des CICS-Web-Service-Assistenten oder des CICS-XML-Assistenten ausführen. Diese Rückgabecodes haben die folgende Bedeutung:
 - 0 - Der Job wurde erfolgreich abgeschlossen.
 - 4 - Warnung. Der Job wurde erfolgreich abgeschlossen, es wurde jedoch mindestens eine Warnung ausgegeben.
 - 8 - Eingabefehler. Der Job wurde nicht erfolgreich abgeschlossen. Während der Validierung der Eingabeparameter wurde mindestens eine Warnung ausgegeben.
 - 12 - Fehler. Der Job wurde nicht erfolgreich abgeschlossen. Während der Ausführung wurde mindestens eine Warnung ausgegeben.

1. Prüfen Sie das Jobprotokoll auf Warnungen oder Fehlermeldungen. Schlagen Sie die detaillierten Erläuterungen für die Nachrichten nach. Die Erläuterungen beschreiben normalerweise Aktionen, die Sie ausführen können, um das Problem zu beheben.
 2. Stellen Sie sicher, dass Sie die richtigen Werte für die einzelnen Parameter im Job eingegeben haben. Für Parameterwerte wie Dateinamen und Elemente in der Web-Service-Beschreibung muss die Groß-/Kleinschreibung beachtet werden.
 3. Stellen Sie sicher, dass Sie die richtige Kombination von Parametern angegeben haben. Wenn Sie beispielsweise beim Generieren einer Web-Service-Bindungsdatei für einen Service-Requester den Parameter **PGMNAME** in DFHWS2LS angeben, wird ein Fehler ausgegeben und der Job wird nicht erfolgreich abgeschlossen.
- Sie empfangen den Rückgabecode 1, 136 oder 139, wenn Sie die Stapeljobs des CICS-Web-Service-Assistenten oder des CICS-XML-Assistenten ausführen. Diese Rückgabecodes geben an, dass die JVM fehlgeschlagen ist, in der Regel, weil nicht genügend Speicher verfügbar ist. Für CICS-Assistenten ist eine JCL-Regionsgröße von mindestens 300 MB erforderlich, wobei einige Dokumente möglicherweise 400 MB benötigen.
 1. Korrigieren Sie die Regionsgröße nach oben oder ziehen Sie in Betracht, die Regionsgröße auf 0M festzulegen.
 2. Prüfen Sie, ob aktive IEFUSI-Exits vorhanden sind, die die Regionsgröße begrenzen können.

Anmerkung: Wenn Sie eine 64-Bit-JVM verwenden, stellen Sie sicher, dass Sie einen geeigneten MEMLIMIT-Wert angeben.

- Sie empfangen den Rückgabecode 137, wenn Sie den Stapeljob DFHLS2WS des CICS-Web-Service-Assistenten oder den Stapeljob DFHLS2SC des CICS-XML-Assistenten ausführen. Dieser Rückgabecode bedeutet, dass der Job das zulässige Zeitlimit überschritten hat.
 1. Korrigieren Sie die Zeitangabe nach oben, indem Sie den Parameter **TIME** in der EXEC-Anweisung Ihres Jobs als **TIME=1440** codieren, oder erhöhen Sie den MAXCPU-TIME-Wert im Member SYS1.PARMLIB(BPXPRMxx).
- Sie empfangen eine DFHPI0914-Fehlermeldung, wenn Sie versuchen, eine WEB-SERVICE-Ressource zu installieren. Die Nachricht enthält Informationen zur Ursache des Installationsfehlers.
 1. Prüfen Sie, dass Sie CICS die Berechtigung erteilt haben, die Web-Service-Bindungsdatei in z/OS UNIX zu lesen.
 2. Prüfen Sie, dass die Web-Service-Bindungsdatei nicht beschädigt ist. Dies kann beispielsweise passieren, wenn Sie FTP für die Übertragung der Datei nach z/OS UNIX im Textmodus statt im Binärmodus verwenden.
 3. Prüfen Sie, dass sich die beiden Web-Service-Bindungsdateien mit demselben Namen nicht in verschiedenen Abholverzeichnis befinden.
 4. Wenn Sie versuchen, eine Ressource für einen Web-Service-Requester-Anwendung zu installieren, prüfen Sie, dass die Version der SOAP-Bindung mit der unterstützten Version in der Pipeline übereinstimmt. Sie können keine SOAP 1.1-WEBSERVICE in einer Service-Requester-Pipeline installieren, die SOAP 1.2 unterstützt.
 5. Prüfen Sie, dass Sie keine WEBSERVICE-Ressource im Providermodus in einer Pipeline im Requestermodus installieren. Im Gegensatz zu Bindungsdateien im Requestermodus geben Web-Service-Bindungsdateien im Providermodus einen Wert **PROGRAM** an.

6. Wenn Sie DFHWS2LS oder DFHLS2WS verwenden, prüfen Sie, dass Sie beim Generieren der Web-Service-Bindungsdatei die richtigen Parameter angegeben haben. Manche Parameter, darunter **PGMNAME**, sind nur für Web-Service-Provider zulässig und müssen ausgeschlossen werden, wenn Sie einen Web-Service-Requester erstellen.
 7. Wenn Sie DFHWS2LS oder DFHLS2WS verwenden, prüfen Sie die Nachrichten, die von dem Job ausgegeben wurden, auf Probleme, die behoben werden müssen, bevor Sie die WEBSERVICE-Ressource erstellen.
- Die PIPELINE-Ressource kann nicht installiert werden und Sie empfangen eine Fehlnachricht wie DFHPI0700, DFHPI0712, DFHPI0714.
 1. Wenn Sie eine DFHPI0700-Fehlnachricht empfangen haben, müssen Sie die PL/I-Sprachunterstützung in Ihrer CICS-Region aktivieren. Dies ist eine Voraussetzung für die Installation von PIPELINE-Ressourcen. Weitere Informationen finden Sie unter Language Environment support for PL/I.
 2. Prüfen Sie, dass Sie CICS die Berechtigung erteilt haben, auf die z/OS UNIX-Verzeichnisse zuzugreifen, um die Pipelinekonfigurationsdateien zu lesen.
 3. Prüfen Sie, dass das Verzeichnis, das Sie im Parameter **WSDIR** angeben, gültig ist. Achten Sie insbesondere auf die Groß-/Kleinschreibung der Verzeichnis- und Dateinamen in z/OS UNIX.
 4. Stellen Sie sicher, dass keine PIPELINE-Ressource desselben Namens mit dem Status ENABLED in der CICS-Region vorhanden ist.
 - Die PIPELINE-Ressource wird mit einem DISABLED-Status installiert. Es wird eine Fehlnachricht im Bereich zwischen DFHPI0702 bis DFHPI0711 ausgegeben.
 1. Prüfen Sie, dass die Pipelinekonfigurationsdatei fehlerfrei ist. Die Elemente in der Pipelinekonfigurationsdatei können nur an bestimmten Stellen vorkommen. Wenn Sie diese falsch angeben, empfangen Sie die Fehlnachricht DFHPI0702. Diese Nachricht enthält den Namen des Elements, das das Problem verursacht. Prüfen Sie die Elementbeschreibung, um sicherzustellen, dass Sie es an der richtigen Stelle codiert haben.
 2. Prüfen Sie, dass keine nicht druckbaren Zeichen wie Tabulatoren in der Pipelinekonfigurationsdatei vorhanden sind.
 3. Prüfen Sie, dass die XML gültig ist. Wenn die XML nicht gültig ist, kann dies beim Versuch, die PIPELINE-Ressource zu installieren, zu Parsing-Fehlern führen.
 4. Stellen Sie sicher, dass die Pipelinekonfigurationsdatei US EBCDIC codiert ist. Wenn Sie versuchen, eine andere EBCDIC-Codierung zu verwenden, kann CICS die Datei nicht verarbeiten.
 - Die WEBSERVICE-Ressource hat einen DISABLED-Status. Die Status DISABLED und DISABLING sind nur für WEBSERVICE-Ressourcen verfügbar, die in CICS-Bundles definiert und installiert sind.
 1. Wenn die der WEBSERVICE-Ressource zugeordnete PIPELINE-Ressource inaktiviert wurde, nimmt die WEBSERVICE-Ressource den Status DISABLED an. Finden Sie heraus, warum die PIPELINE-Ressource fehlt, und ersetzen Sie sie bei Bedarf.
 2. Wenn für das CICS-Bundle, in dem WEBSERVICE-Ressource definiert ist, eine Inaktivierungsaktion ausgeführt wurde, nimmt die WEBSERVICE-Ressource den Status DISABLED an, wenn der Web-Service nicht mehr verwendet wird. Überprüfen Sie den Status des CICS-Bundles und aktivieren Sie es bei Bedarf.

Laufzeitfehler des Service-Providers diagnostizieren

Wenn Sie Probleme beim Empfangen oder Verarbeiten eingehender Nachrichten in einer Pipeline im Providermodus haben, könnte der Transport oder eine bestimmte SOAP-Nachricht die Ursache sein.

Procedure

- Sie empfangen eine Nachricht wie DFHPI0401 oder DFHPI0502, die angibt, dass ein HTTP- oder WebSphere MQ-Transportfehler aufgetreten ist. Bei einem HTTP-Transport empfängt der Client eine '500 Server Internal Error'-Nachricht. Bei einem WebSphere MQ-Transport wird die Nachricht in die Warteschlange für nicht zustellbare Nachrichten geschrieben. Es wird kein SOAP-Fehler an den Web-Service-Requester zurückgegeben, weil CICS nicht bestimmen kann, welcher Typ von Nachricht empfangen wurde.
- Sie empfangen eine 'DFHxx'-Nachricht und eine '404 Not Found'-Fehlernachricht.
 1. Wenn Sie den Web-Service-Assistenten nicht verwenden, müssen Sie eine URIMAP-Ressource erstellen. Falls Sie den Web-Service-Assistenten verwenden, wird die URIMAP automatisch für Sie erstellt, wenn Sie den Befehl **PIPELINE SCAN** ausführen. Im Systemprotokoll finden Sie Informationen zu allen Fehlern, die infolge dieses Befehls aufgetreten sind.
 2. Prüfen Sie, dass die WEBSERVICE-Ressource aktiviert ist und dass die URIMAP, der sie zugeordnet ist, Ihren Erwartungen entspricht. Wenn Ihre WEBSERVICE-Ressource den Status UNUSABLE hat, finden Sie weitere Informationen unter „Implementierungsfehler diagnostizieren“ auf Seite 673.
 3. Prüfen Sie, dass Sie den URI und die Portnummer korrekt angegeben haben. Achten Sie insbesondere auf die Groß-/Kleinschreibung des Attributs PATH in der URIMAP-Ressource.
- Wenn keine unerwarteten Fehler berichtet werden, könnten Sie CEDX zum Debuggen der Web-Service-Anwendung verwenden.
 1. Prüfen Sie im Systemprotokoll, welche Fehlernachrichten von CICS berichtet werden. Dies kann ein Indikator für die Art der aufgetretenen Fehler sein. Wenn CICS keine Fehler berichtet, stellen Sie sicher, dass die Anforderung CICS über das Netz erreicht.
 2. Führen Sie CEDX gegen CPIH für den HTTP-Transport, CPIQ für den WebSphere MQ-Transport oder die in der URIMAP angegebene Transaktion aus, falls sich diese unterscheidet.

Wenn während der Pipelineverarbeitung vor dem Anwendungshandler ein Taskwechsel auftritt, wird die neue Task unter derselben Transaktions-ID wie die erste Task ausgeführt, es sei denn, der Container DFHWS-TRANID ist gefüllt. Dies kann die Ausführung von CEDX beeinträchtigen, da die erste Task in der CEDX-Sitzung gesperrt ist. Sie können dieses Problem vermeiden, indem Sie DFHWS-TRANID verwenden, um die Transaktions-ID zu ändern, wenn die Task gewechselt wird, wodurch Sie CEDX separat in den Pipeline- und den Anwendungstasks verwenden können. Weitere Informationen zu CEDX finden Sie unter Using the CEDX transaction.
 3. Wenn CEDX nicht aktiviert wird und Sie das Problem nicht beheben können, können Sie einen Hilfstrace ausführen, wobei die PI-, SO-, AP-, EI- und XS-Domänen aktiv sind. Dies könnte anzeigen, ob es ein Sicherheitsproblem, ein TCP/IP-Problem, ein Anwendungsprogrammproblem oder ein Pipelineproblem in Ihrer CICS-Region gibt. Suchen Sie nach Ausnahmebedingungstracepunkten oder Abbrüchen.
- Informationen zu Konvertierungsfehlern finden Sie unter „Datenkonvertierungsfehler diagnostizieren“ auf Seite 681.

- Wenn Sie der Auffassung sind, dass das Problem in Zusammenhang mit MTOM-Nachrichten steht, finden Sie weitere Informationen unter „MTOM/XOP-Fehler diagnostizieren“ auf Seite 679.
- Wenn eine persistente HTTP-Verbindung in regelmäßigen Abständen geschlossen wird:
 1. Prüfen Sie, ob eine Leistungsoptimierung für HTTP-Verbindungen aktiviert ist. Weitere Informationen finden Sie unter SOTUNING.
 2. Wenn eine Leistungsoptimierung aktiviert ist, schließt CICS persistente HTTP-Verbindungen in regelmäßigen Abständen, damit Verbindungen zwischen den Regionen, die an gemeinsam genutzten IP-Endpunkten empfangsbereit sind, neu verteilt werden können.
- Falls Verbindungsversuche zurückgewiesen werden, wenn Ihre CICS-Region maximal ausgelastet ist:
 1. Prüfen Sie, ob eine Leistungsoptimierung für HTTP-Verbindungen aktiviert ist. Weitere Informationen finden Sie unter SOTUNING.
 2. Ist eine Leistungsoptimierung aktiviert, werden alle eingehenden HTTP-Öffnungsanforderungen außerhalb von CICS in der Rückstandswarteschlange der Empfangsverbindung des TCPIP SERVICE in TCP/IP gestellt, wenn CICS maximal ausgelastet ist. Das TCP/IP Global statistics-Feld SOG_PAUSING_HTTP_LISTENING spiegelt wider, ob dies aktuell der Fall ist, und die Felder SOG_TIMES_AT_ACCEPT_LIMIT/ SOG_TIME_LAST_PAUSED_HTTP_LISTENING enthalten weitere Informationen zu früheren Vorkommen.
 3. Verwenden Sie **NETSTAT ALL**, um die Statistik zu gelöschten Verbindungen für die Empfangsverbindung des TCPIP SERVICE abzurufen. Dies ist die Anzahl von Verbindungsanforderungen, die empfangen und gelöscht wurden, da die Anzahl von Verbindungsanforderungen in der Rückstandswarteschlange den Maximalwert erreicht hat. Weitere Informationen finden Sie unter Netstat in z/OS Communications Server: IP System Administrator's Commands.
 Statistiken zu gelöschten Verbindungen finden Sie außerdem in den folgenden Feldern unter TCP/IP-Services: Ressourcenstatistik:
 - Gelöschte Verbindungen (SOR_CONNS_DROPPED)
 - Zeitpunkt der zuletzt gelöschten Verbindung (SOR_CONN_LAST_DROPPED)
 4. Wenn die Anzahl der gelöschten Verbindungen zu hoch ist, ziehen Sie in Betracht, den Maximalwert für die Rückstandswarteschlange des TCPIP SERVICE zu erhöhen.

Laufzeitfehler des Service-Requesters diagnostizieren

In diesem Thema finden Sie hilfreiche Informationen, wenn beim Senden von Web-Service-Anforderungen aus Ihrer Service-Requester-Anwendung Probleme auftreten oder wenn Sie SOAP-Fehlernachrichten vom Web-Service-Provider empfangen.

About this task

Probleme können aufgrund von Fehlern ein einzelnen Web-Services oder auf Transportebene auftreten.

Procedure

- Wenn Sie den Befehl **INVOKE SERVICE** in Ihrem Anwendungsprogramm verwenden, wird ein RESP- oder RESP2-Code zurückgegeben, wenn ein Problem auftritt.
 1. Schlagen Sie die Bedeutung der RESP- und RESP2-Codes für den Befehl **INVOKE SERVICE** nach, um zu verstehen, worin das Problem bestehen könnte.
 2. Prüfen Sie das CICS-Systemprotokoll auf Nachrichten, die zur Bestimmung der Ursache des Problems beitragen können.
- Wenn Sie keine SOAP-Anforderungsnachricht senden können und die Pipeline einen DFHERROR-Container zurückgibt, gab es ein Problem, als die Pipeline versucht hat, die SOAP-Nachricht zu verarbeiten.
 1. Sehen Sie sich die Inhalte des DFHERROR-Containers an. Diese sollten eine Fehlernachricht sowie Daten enthalten, die das aufgetretene Problem beschreiben.
 2. Haben Sie neue Nachrichtenhandler oder Programme zur Headerverarbeitung in der Pipeline eingeführt? Falls ja, versuchen Sie, das neue Programm zu entfernen und den Web-Service erneut auszuführen, um zu sehen, ob dies das Problem löst. Wenn Ihr Nachrichtenhandler versucht, eine Verarbeitung mithilfe eines Containers auszuführen, der nicht in der Pipeline vorhanden ist, oder wenn er versucht, einen schreibgeschützten Container zu aktualisieren, stoppt die Pipeline die Verarbeitung und gibt einen Fehler im Container DFHERROR zurück. Programme zur Headerverarbeitung können nur eine begrenzte Anzahl von Containern in der Pipeline aktualisieren. Ausführliche Informationen finden Sie unter The header processing program interface.
 3. Wenn die Web-Service-Requester-Anwendung nicht den Befehl **INVOKE SERVICE** zum Senden einer Web-Service-Anwendung verwendet, prüfen Sie, dass alle erforderlichen Steuercontainer erstellt wurden und dass sie den richtigen Datentyp haben. Prüfen Sie insbesondere, dass der Container DFHREQUEST den Datentyp CHAR statt BIT hat.
 4. Wenn die Web-Service-Requester-Anwendung den Befehl **INVOKE SERVICE** verwendet und der RESP-Wert INVREQ sowie ein RESP2-Code von 14 zurückgegeben werden, weist dies auf einen Datenkonvertierungsfehler hin. Siehe „Datenkonvertierungsfehler diagnostizieren“ auf Seite 681.
 5. Prüfen Sie, dass die XML in Ihrer SOAP-Nachricht nicht durch einen angepassten Nachrichtenhandler während der Pipelineverarbeitung ungültig gemacht wurde. CICS führt keine Validierung für ausgehende Nachrichten in der Pipeline durch. Wenn Ihre Anwendung den Befehl **INVOKE SERVICE** verwendet, wird die XML von CICS generiert und korrekt formatiert, wenn der Hauptteil der SOAP-Nachricht in den Container DFHREQUEST gestellt wird. Wenn Sie jedoch über zusätzliche Nachrichtenhandler verfügen, die den Inhalt der SOAP-Nachricht ändern, wird dies in der Pipeline nicht validiert.
- Wenn Sie eine SOAP-Nachricht senden können, aber einen Fehler aufgrund einer Zeitlimitüberschreitung oder einen Transportfehler empfangen, wird dies normalerweise als SOAP-Fehler zurückgegeben. Wenn Ihr Programm den Befehl **INVOKE SERVICE** verwendet, gibt CICS den RESP-Wert TIMEDOUT und den RESP2-Code '2' für einen Fehler aufgrund einer Zeitlimitüberschreitung sowie den RESP-Wert INVREQ und den RESP2-Code '17' für einen Transportfehler zurück.
 1. Prüfen Sie, dass der Netzendpunkt vorhanden ist.
 2. Stellen Sie sicher, dass das Attribut RESPWAIT in der PIPELINE-Ressource korrekt konfiguriert ist, um die Anforderungen Ihrer Anwendung zu erfüllen. Das Attribut RESPWAIT definiert, wie lange CICS auf eine Antwort vom Web-Service-Provider wartet, bevor es zur Anwendung zurückkehrt. Wird kein Wert angegeben, verwendet CICS die Standardwerte von 10 Sekunden

für HTTP und von 60 Sekunden für WebSphere MQ. Allerdings verfügt CICS auch über ein Zeitlimit in der Zustelleroutine für jede Transaktion und wenn dieses kleiner ist als der Standardwert des verwendeten Protokolls, verwendet CICS stattdessen das Zeitlimit der Zustelleroutine.

- Wenn Sie eine SOAP-Nachricht senden können, aber vom Web-Service-Provider eine SOAP-Fehlernachricht empfangen, mit der Sie nicht gerechnet haben, prüfen Sie die Inhalte des Containers DFHWS-BODY auf Details zu dem SOAP-Fehler.
 1. Wenn Sie einen vollständigen SOAP-Envelope in DFHREQUEST über die Schnittstelle DFHPIRT gesendet haben, stellen Sie sicher, dass die ausgehende Nachricht keine doppelten SOAP-Header enthält. Dies kann passieren, wenn die Requester-Pipeline einen SOAP 1.1- oder SOAP 1.2-Nachrichtenhandler verwendet. Die SOAP-Nachrichtenhandler fügen SOAP-Header hinzu, selbst wenn sie bereits von der Service-Requester-Anwendung im SOAP-Envelope angegeben sind. In diesem Szenario haben Sie folgende Möglichkeiten:
 - Entfernen Sie den SOAP 1.1- oder SOAP 1.2-Nachrichtenhandler aus der Pipeline. Dies wirkt sich auf alle anderen Service-Requester-Anwendungen aus, die diese Pipeline verwenden.
 - Entfernen Sie die SOAP-Header aus dem SOAP-Envelope, den die Anwendung in DFHREQUEST platziert. CICS fügt die erforderlichen SOAP-Header für Sie hinzu. Wenn Sie zusätzliche Verarbeitungsschritte für die Header ausführen möchten, können Sie dies über die Schnittstelle des Programms zur Headerverarbeitung tun.
 - Verwenden Sie in Ihrer Anwendung stattdessen einen Befehl **WEB SEND** und inaktivieren Sie die Web-Service-Unterstützung.
- Wenn Sie der Auffassung sind, dass das Problem in Zusammenhang mit dem Senden oder Empfangen von MTOM-Nachrichten steht, finden Sie weitere Informationen unter „MTOM/XOP-Fehler diagnostizieren“.

MTOM/XOP-Fehler diagnostizieren

MTOM/XOP-Fehler können zur Laufzeit auftreten, sowohl in Requestermodus- als auch in Providermodus-Pipelines.

Before you begin

Wenn bei der Konfiguration einer Pipeline für die Unterstützung von MTOM/XOP Probleme auftreten, finden Sie weitere Informationen unter „Implementierungsfehler diagnostizieren“ auf Seite 673.

About this task

Procedure

- Wenn Sie eine Web-Service-Anforderungsnachricht im MTOM-Format senden können, aber vom Web-Service-Provider eine SOAP-Fehlernachricht empfangen, prüfen Sie die Inhalte des Containers DFHWS-BODY auf Details zu dem SOAP-Fehler.
 1. Kann der Web-Service-Provider MIME Multipart/Related-Nachrichten empfangen? Wenn der Web-Service-Provider das MTOM-Format nicht unterstützt, kann der Fehler, den Sie empfangen, abhängig von der Implementierung variieren. Wenn der Web-Service-Provider eine andere CICS-Anwendung ist, würde der SOAP-Fehler darauf hinweisen, dass die MIME-Nachricht kein gültiger Inhaltstyp ist.

2. Wenn der Web-Service-Provider MIME-Nachrichten empfangen kann, überprüfen Sie, ob die Pipeline die Nachricht im Direktmodus oder im Kompatibilitätsmodus sendet. Wenn Sie eine MTOM-Nachricht im Direktmodus senden, könnte ein Problem mit der XML vorliegen.
 3. Um herauszufinden, ob die XML das Problem ist, aktivieren Sie die Validierung für den Web-Service. Dies führt dazu, dass die MTOM-Nachricht im Kompatibilitätsmodus über die Pipeline verarbeitet wird. Als Teil dieser Verarbeitung parst der MTOM-Handler die Nachrichteninhalte, um die base64binary-Daten zu optimieren. Wenn in der XML ein Fehler vorliegt, stellt CICS den Fehler in den Container DFHERROR und gibt einen MTOM-Transportfehler in der Pipeline aus.
 4. Überprüfen Sie die Inhalte des Containers DFHERROR auf Informationen zur Ursache des Problems. Wenn dies die Diagnose der Ursache des Problems nicht voranbringt, führen Sie einen Trace der Ebene 2 der PI-Domäne aus.
 5. Suchen Sie nach dem Tracepunkt PI 0C16. Dieser beschreibt das aufgetretene Problem detaillierter und sollte Ihnen dabei helfen, das Problem anhand der XML, die von der Requester-Anwendung bereitgestellt wird, zu beheben.
- Wenn erwartete binäre Anlagen in der ausgehenden MTOM-Nachricht fehlen, könnte dies ein Hinweis sein, dass die binären Daten zu klein für die Optimierung als binäre Anlage sind. CICS erstellt nur für solche Daten binäre Anlagen, die groß genug sind, um den Verarbeitungsaufwand einer Optimierung in der Pipeline zu rechtfertigen. Alle binären Daten kleiner als 1.500 Bytes werden nicht optimiert.
 - Wenn Sie keine ausgehende MTOM-Nachricht im Kompatibilitätsmodus senden können und die Pipeline einen DFHERROR-Container zurückgibt, gab es ein Problem, als die Pipeline versucht hat, die MTOM-Nachricht zu verarbeiten.
 1. Sehen Sie sich die Inhalte des DFHERROR-Containers an. Diese sollten eine Fehlernachricht sowie Daten enthalten, die das aufgetretene Problem beschreiben.
 2. Prüfen Sie, dass die XML in der ausgehenden MTOM-Nachricht gültig ist. CICS führt keine Validierung für ausgehende Nachrichten in der Pipeline durch.
 - Wenn Sie eine Nachricht DFHP1100E empfangen, gab es ein Problem mit den MIME-Headern einer MTOM-Nachricht, die von CICS empfangen wurde. Die CICS-Nachricht enthält die allgemeine Klasse des MIME-Fehlers, der aufgetreten ist. Gehen Sie wie folgt vor, um herauszufinden, welches Problem genau aufgetreten ist:
 1. Wenn ein Hilfstrace in Ihrer CICS-Region aktiv ist, prüfen Sie, ob es Einträge im Ausnahmebedingungstrace gibt.
 2. Suchen Sie nach dem Tracepunkt PI 1305. Dieser beschreibt die Art des MIME-Headerfehlers, die Position des Fehlers im Header und bis zu 80 Bytes Text vor und nach dem Fehler, damit Sie nachvollziehen können, in welchem Kontext der Fehler aufgetreten ist.

Der folgende Traceauszug gibt beispielsweise an, dass der MIME-Startparameter 'content-type' ungültig war, weil er nicht in Anführungsstriche gesetzt wurde, sondern Zeichen enthielt, die außerhalb einer Zeichenfolge in Anführungszeichen ungültig sind.

PI 1305 PIMM *EXC* - MIME_PARSE_ERROR -

```
TASK-01151 KE_NUM-0214 TCB-QR /009C7B68 RET-9C42790A TIME-10:33:41.3667303015 INTERVAL-00.0000053281 =000599=
1-0000 C5A79785 83A38584 40978199 819485A3 859940A5 8193A485 40A39692 85954096 *Expected parameter value token o*
0020 994098A4 96A38584 40A2A399 899587 *r quoted string *
2-0000 D4C9D4C5 40A2A895 A381A740 85999996 994081A3 404EF0F0 F0F0F1F1 F2408995 *MIME syntax error at +0000112 in*
0020 40C39695 A38595A3 60A3A897 85408885 81848599 * Content-type header *
3-0000 5F626F75 6E646172 793B2074 7970653D 22617070 6C696361 74696F6E 2F786F70 *_boundary; type="application/xop*
```



```

0020 2B786D6C 223B2073 74617274 2D696E66 6F3D2261 70706C69 63617469 6F6E2F73 **xml"; start-info="application/s*
0040 6F61702B 786D6C22 3B207374 6172743D *oap+xml"; start=
4-0000 3C736F61 70736C75 6E674074 6573742E 68757273 6C65792E 69626D2E 636F6D3E *soapslung@test.hursley.ibm.com>*
0020 3B206368 61727365 743D7574 662D38 *; charset=utf-8

```

- Die Pipelineverarbeitung kann eine eingehende MTOM-Nachricht nicht parsen und der Web-Service-Requester empfängt eine SOAP-Fehlernachricht. Dies weist darauf hin, dass es ein Problem mit dem XOP-Dokument in der MTOM-Nachricht gibt. Im Direktmodus wird der SOAP-Fehler vom Anwendungshandler generiert. Wenn die Pipeline im Kompatibilitätsmodus ausgeführt wird, wird die Nachricht von dem MTOM-Handler beim Erstellen der SOAP-Nachricht geparkt. In diesem Fall gibt CICS eine Fehlernachricht mit dem Präfix DFHPI und einen SOAP-Fehler aus.

1. Die Fehlernachricht mit dem Präfix DFHPI gibt an, was mit dem XOP-Dokument nicht gestimmt hat. Es könnte beispielsweise ein ungültiger MIME-Header oder eine fehlende binäre Anlage sein.
2. Prüfen Sie auf Ausnahmebedingungstracepunkte, um die genaue Ursache des Problems herauszufinden. Suchen Sie insbesondere nach Tracepunkten, die mit 'PI 13xx' beginnen. Dies beschreibt die aufgetretene Ausnahme detaillierter.

Sie können auch einen PI-Trace der Ebene 2 ausführen, um die Abfolge der Ereignisse festzustellen, die zu dem Fehler geführt haben, dies kann sich allerdings deutlich auf die Leistung auswirken und wird in Produktionsregionen nicht empfohlen.

Datenkonvertierungsfehler diagnostizieren

Datenkonvertierungsfehler können zur Laufzeit auftreten, während der Konvertierung einer SOAP-Nachricht in einen CICS-Kommunikationsbereich oder -Container und aus einem Kommunikationsbereich oder Container in eine SOAP-Nachricht.

Before you begin

Zu den Symptomen gehören die Generierung von SOAP-Fehlernachrichten und CICS-Nachrichten, die darauf hinweisen, dass ein Fehler aufgetreten ist.

About this task

Führen Sie die folgenden Schritte aus, wenn bei der Datenkonvertierung ein Problem auftritt:

Procedure

1. Stellen Sie sicher, dass die WEBSERVICE-Ressource aktuell ist. Generieren Sie die Web-Service-Bindungsdatei für den Web-Service neu und implementieren Sie sie wieder in CICS.
2. Stellen Sie sicher, dass der ferne Web-Service unter Verwendung derselben Version des Web-Service-Dokuments (WSDL) generiert wurde, die von CICS verwendet oder generiert wurde.
3. Wenn Sie sicher sind, dass die WEBSERVICE-Ressource eine aktuelle Web-Service-Bindungsdatei verwendet, gehen Sie wie folgt vor:
 - a. Aktivieren Sie die Laufzeitvalidierung für die WEBSERVICE-Ressource mit dem Befehl SET WEBSERVICE(*name*) VALIDATION, wobei *name* der WEBSERVICE-Ressourcenname ist.
 - b. Suchen Sie nach den CICS-Nachrichten DFHPI1001 oder DFHPI1002 in dem Nachrichtenprotokoll. DFHPI1001 beschreibt die genaue Art des Datenkon-

vertierungsproblems und unterstützt Sie dabei, die Quelle des Konvertierungsfehlers ausfindig zu machen. DFHPI1002 gibt an, dass keine Probleme gefunden wurden.

- c. Wenn Sie keine Validierung mehr für den Web-Service benötigen, setzen Sie den folgenden Befehl ab, um sie auszuschalten: SET WEBSERVICE(*name*) NOVALIDATION.
4. Wenn Sie die Ursache des Konvertierungsfehlers noch immer nicht feststellen könnten, erstellen Sie einen CICS-Trace des fehlerhaften Web-Service. Suchen Sie nach den folgenden Einträgen im Ausnahmebedingungs-Trace der PI-Domäne:

```
PI 0F39 - PICC    *EXC* - CONVERSION_ERROR
PI 0F08 - PIII    *EXC* - CONVERSION_ERROR
```

Ein PICC-Konvertierungsfehler weist darauf hin, dass bei der Umsetzung einer eingehenden SOAP-Nachricht in einen Kommunikationsbereich oder Container ein Problem aufgetreten ist. Ein PIII-Konvertierungsfehler weist darauf hin, dass bei der Generierung einer SOAP-Nachricht aus einem Kommunikationsbereich oder Container, der von dem Anwendungsprogramm bereitgestellt wird, ein Problem aufgetreten ist. In beiden Fällen gibt der Tracepunkt den Namen des Felds an, das dem Konvertierungsfehler zugeordnet ist, und unter Umständen auch den Wert, der das Problem verursacht. Wenn diese Tracepunkte vorhanden sind, folgt ihnen ein Konvertierungsfehler. Eine mögliche Interpretation dieser Konvertierungsfehler finden Sie in den Erläuterungen zu den Nachrichten DFHPI1007 bis DFHPI1010.

Gründe für Datenkonvertierungsfehler

CICS validiert SOAP-Nachrichten nur in dem Maße, in dem es erforderlich ist, um zu bestätigen, dass sie korrekt formatierte XML enthalten, und um sie umzusetzen. Das bedeutet, dass eine SOAP-Nachricht erfolgreich mithilfe von WSDL überprüft werden kann, aber dann in der Laufzeitumgebung fehlschlägt und umgekehrt.

Die WEBSERVICE-Ressource bindet die Zuordnungsanweisungen ein, um CICS die Ausführung der Datenkonvertierung zur Laufzeit zu ermöglichen. Ein Konvertierungsfehler tritt auf, wenn die Eingabe nicht mit den erwarteten Daten übereinstimmt (siehe Beschreibung in der WEBSERVICE-Ressource).

Diese fehlende Übereinstimmung kann aus den folgenden Gründen auftreten:

- Eine SOAP-Nachricht, die von CICS empfangen wird, ist nicht korrekt formatiert und gültig, wenn sie anhand der Web-Service-Beschreibung (WSDL), die der WEBSERVICE-Ressource zugeordnet ist, überprüft wird.
- Eine SOAP-Nachricht, die von CICS empfangen wird, ist korrekt formatiert und gültig, enthält aber Werte außerhalb des gültigen Bereichs der WEBSERVICE-Ressource.
- Die Inhalte eines Kommunikationsbereichs oder Containers sind nicht konsistent mit der WEBSERVICE-Ressource und der Sprachstruktur, aus der der Web-Service generiert wurde.

Das WSDL-Dokument kann beispielsweise Bereichseinschränkungen für ein Feld angeben, z. B. eine nicht signierte ganze Zahl, die nur Werte zwischen 10 und 20 haben kann. Wenn eine SOAP-Nachricht einen Wert von 25 enthält, würde die Validierung der SOAP-Nachricht dazu führen, dass sie als ungültig zurückgewiesen wird. Der Wert 25 wird als gültiger Wert für eine ganze Zahl akzeptiert und an die Anwendung übergeben.

In einem zweiten Beispiel gibt das WSDL-Dokument eine Zeichenfolge an, ohne eine maximale Länge anzugeben. DFHWS2LS nimmt standardmäßig eine maximale Länge von 255 Zeichen an, wenn die Web-Service-Bindungsdatei generiert wird. Wenn die SOAP-Nachricht 300 Zeichen enthält, würde beim Versuch, die Nachricht umzusetzen, ein Fehler berichtet, auch wenn die Prüfung gegen die WSDL feststellen würde, dass für die Nachricht keine maximale Länge festgelegt ist, da der Wert nicht in den 255-Zeichen-Puffer von CICS passt.

Codepageprobleme

CICS verwendet den Wert des Systeminitialisierungsparameters **LOCALCCSID**, um die Anwendungsprogrammdateien zu codieren. Die Web-Service-Bindungsdatei ist jedoch in US EBCDIC (Cp037) codiert. Dies kann zu Problemen bei der Datenkonvertierung führen, wenn die vom Anwendungsprogramm verwendete Codepage Zeichen anders als die US EBCDIC-Codepage codiert. Um dieses Problem zu vermeiden, können Sie den Parameter **CCSID** in den Stapeljobs des Web-Service-Assistenten verwenden, um eine andere Codepage zum Codieren von Daten zwischen dem Anwendungsprogramm und der Web-Service-Bindungsdatei zu codieren. Der Wert dieses Parameters überschreibt den Systeminitialisierungsparameter **LOCALCCSID** für diese bestimmte WEBSERVICE-Ressource. Der Wert von **CCSID** muss eine EBCDIC-CCSID sein.

SOAP-Fehlernachrichten für Konvertierungsfehler

Wenn zur Laufzeit ein Konvertierungsfehler auftritt und CICS als Web-Service-Provider fungiert, wird eine SOAP-Fehlernachricht an den Service-Requester ausgegeben. Diese SOAP-Fehlernachricht enthält die Nachricht, die von CICS ausgegeben wird.

Der Service-Requester kann eine der folgenden SOAP-Fehlernachrichten empfangen:

- Cannot convert SOAP message

Diese Fehlernachricht gibt an, dass die SOAP-Nachricht entweder nicht korrekt formatiert und gültig ist oder dass ihre Werte außerhalb des gültigen Bereichs liegen.

- Outbound data cannot be converted

Diese Fehlernachricht gibt an, dass der Inhalt eines Kommunikationsbereichs oder Containers nicht konsistent ist.

- Operation not part of web service

Diese Fehlernachricht ist eine spezielle Variante einer ungültigen SOAP-Nachricht, die von CICS empfangen wird.

Wenn CICS der Web-Service-Requester ist, gibt der Befehl **INVOKE SERVICE** einen RESP-Code von INVREQ und einen RESP2-Wert von 14 zurück.

Fehlerbehebung bei JSON-Web-Services

Die möglichen Probleme bei der Implementierung von JSON-Web-Services in CICS können während der Implementierung selbst oder während der Ausführung auftreten, wenn CICS die Nachrichten umsetzt.

About this task

Diese Informationen zur Problembestimmung sind für CICS spezifisch. Wenn Sie ein Problem dem IBM Support melden möchten, verwenden Sie die Informationen in den MustGather-Daten, um sicherzustellen, dass Sie alle erforderlichen Diagnoseinformationen zusammengestellt haben.

Der Support für CICS als Service-Provider für JSON-Anforderungen beruht in weiten Teilen auf dem CICS-Support für SOAP-Web-Services.

Fehlerbehebung bei JSON-Implementierungsproblemen

Implementierungsfehler können auftreten, wenn Sie versuchen, eine PIPELINE- oder WEBSERVICE-Ressource in CICS zu installieren. Die gängigsten Implementierungsfehler werden nachfolgend beschrieben, einschließlich Symptom, Ursache und Lösung des Problems.

Procedure

Die folgenden Fehler können auftreten, wenn Sie eine PIPELINE- oder WEBSERVICE-Ressource in CICS installieren:

- Sie empfangen eine DFHPI0914-Fehlernachricht, wenn Sie versuchen, eine WEBSERVICE-Ressource zu installieren. Die Nachricht enthält Informationen zur Ursache des Installationsfehlers.
 1. Prüfen Sie, dass Sie CICS die Berechtigung erteilt haben, die Web-Service-Bindungsdatei in z/OS UNIX zu lesen.
 2. Prüfen Sie, dass die Web-Service-Bindungsdatei nicht beschädigt ist. Dies kann beispielsweise passieren, wenn Sie FTP für die Übertragung der Datei nach z/OS UNIX im Textmodus statt im Binärmodus verwenden.
 3. Prüfen Sie, dass sich die beiden Web-Service-Bindungsdateien mit demselben Namen nicht in verschiedenen Abholverzeichnissen befinden.
 4. Prüfen Sie, dass Sie keine WEBSERVICE-Ressource im Providermodus in einer Pipeline im Requestermodus installieren. Im Gegensatz zu Bindungsdateien im Requestermodus geben Web-Service-Bindungsdateien im Providermodus einen Wert **PROGRAM** an.
 5. Wenn Sie DFHJS2LS oder DFHLS2JS verwenden, prüfen Sie die Nachrichten, die von dem Job ausgegeben werden, auf Probleme, die behoben werden müssen, bevor Sie die WEBSERVICE-Ressource erstellen.
- Die PIPELINE-Ressource kann nicht installiert werden und Sie empfangen eine DFHPI0700-, DFHPI0712-, DFHPI0714- oder andere Fehlernachricht, die angibt, dass die PIPELINE-Ressource fehlgeschlagen ist.
 1. Wenn Sie eine DFHPI0700-Fehlernachricht empfangen haben, müssen Sie die PL/I-Sprachunterstützung in Ihrer CICS-Region aktivieren. Diese Unterstützung ist eine Voraussetzung für die Installation von PIPELINE-Ressourcen. Weitere Informationen finden Sie unter Language Environment support for PL/I.
 2. Prüfen Sie, dass Sie CICS die Berechtigung erteilt haben, auf die z/OS UNIX-Verzeichnisse zuzugreifen, um die Pipelinekonfigurationsdateien zu lesen.
 3. Prüfen Sie, dass das Verzeichnis, das Sie im Parameter **WSDIR** angeben, gültig ist. Achten Sie insbesondere auf die Groß-/Kleinschreibung der Verzeichnis- und Dateinamen in z/OS UNIX.
 4. Stellen Sie sicher, dass keine PIPELINE-Ressource desselben Namens mit dem Status ENABLED in der CICS-Region vorhanden ist.

- Die PIPELINE-Ressource wird mit einem DISABLED-Status installiert. Es wird eine Fehlermeldung im Bereich zwischen DFHPI0702 bis DFHPI0711 ausgegeben.
 1. Prüfen Sie, dass die Pipelinekonfigurationsdatei fehlerfrei ist. Die Elemente in der Pipelinekonfigurationsdatei können nur an bestimmten Stellen vorkommen. Wenn Sie diese falsch angeben, empfangen Sie die Fehlermeldung DFHPI0702. Diese Nachricht enthält den Namen des Elements, das das Problem verursacht. Prüfen Sie die Elementbeschreibung, um sicherzustellen, dass Sie es an der richtigen Stelle codiert haben.
 2. Prüfen Sie, dass keine nicht druckbaren Zeichen wie Horizontaltabulatorzeichen in der Pipelinekonfigurationsdatei vorhanden sind.
 3. Prüfen Sie, dass die XML gültig ist. Wenn die XML nicht gültig ist, kann dies beim Versuch, die PIPELINE-Ressource zu installieren, zu Parsing-Fehlern führen.
 4. Stellen Sie sicher, dass die Pipelinekonfigurationsdatei US EBCDIC codiert ist. Wenn Sie versuchen, eine andere EBCDIC-Codierung zu verwenden, kann CICS die Datei nicht verarbeiten.

Fehlerbehebung für den JSON-Assistenten

Wenn Sie Probleme mit dem JSON-Assistenten haben, können Sie diese mithilfe von Fehlerbehebungsverfahren diagnostizieren.

Procedure

Die folgenden Fehler können auftreten, wenn Sie den JSON-Assistenten ausführen:

- Sie empfangen den Rückgabecode 0, 4, 8 oder 12, wenn Sie die Stapeljobs des CICS-JSON-Assistenten ausführen. Weitere Informationen zu den Rückgabecodes finden Sie unter „Rückgabecodes des JSON-Assistenten“ auf Seite 688.
 1. Prüfen Sie das Jobprotokoll auf Warnungen oder Fehlermeldungen. Schlagen Sie die detaillierten Erläuterungen für die Nachrichten nach. Die Erläuterungen beschreiben oft Aktionen, die Sie ausführen können, um das Problem zu beheben.
 2. Stellen Sie sicher, dass Sie die richtigen Werte für die einzelnen Parameter im Job eingegeben haben. Für Parameterwerte wie Dateinamen und Elemente in der Web-Service-Beschreibung muss die Groß-/Kleinschreibung beachtet werden.
 3. Stellen Sie sicher, dass Sie die richtige Kombination von Parametern angegeben haben.
- Sie empfangen den Rückgabecode 1, 136 oder 139, wenn Sie die Stapeljobs des CICS-JSON-Assistenten ausführen. Diese Rückgabecodes geben an, dass die JVM fehlgeschlagen ist, in der Regel, weil nicht genügend Speicher verfügbar ist. Für CICS-Assistenten ist eine JCL-Regionsgröße von mindestens 300 MB erforderlich.
 1. Korrigieren Sie die Regionsgröße nach oben oder ziehen Sie in Betracht, die Regionsgröße auf 0M festzulegen.
 2. Prüfen Sie, ob aktive IEFUSI-Exits vorhanden sind, die die Regionsgröße begrenzen können.
- Sie empfangen den Rückgabecode 137, wenn Sie den Stapeljob des CICS-JSON-Assistenten ausführen. Dieser Rückgabecode bedeutet, dass der Job das zulässige Zeitlimit überschritten hat.
 1. Korrigieren Sie die Zeitangabe nach oben, indem Sie den Parameter **TIME** in der EXEC-Anweisung Ihres Jobs als **TIME=1440** codieren, oder erhöhen Sie den **MAXCPU**TIME-Wert im Member SYS1.PARMLIB(BPXPRMxx).

- Sie empfangen eine Nachricht im Bereich DFHPI9700 - DFHPI9711, die bei der Ausführung von DFHJS2LS auf ein ungültiges oder nicht unterstütztes JSON-Schema hinweist.
 1. Prüfen Sie, dass Ihr JSON-Schema gültig ist. Beispielsweise indem Sie das Schema mit dem JSON Schema Validation vergleichen oder indem Sie ein Tool wie json-schema-validator verwenden.
 2. Prüfen Sie, dass Ihr JSON-Schema von DFHJS2LS unterstützt wird. Weitere Informationen finden Sie unter High-level language and JSON schema mapping.

What to do next

Nachdem Sie das Problem behoben haben, führen Sie den Stapeljob Ihres JSON-Assistenten erneut aus.

Fehlerbehebung bei JSON-Anforderungen

Wenn CICS JSON zurückweist, entweder als Eingabenachricht für einen Web-Service oder bei der Umsetzung von JSON mit der verknüpfbaren Schnittstelle, ist das JSON-Element möglicherweise ungültig oder entspricht nicht dem Schema.

About this task

Wenn CICS ein Problem mit einer von einem Web-Service-Requester oder einem Anwendungsprogramm bereitgestellten JSON feststellt, wird eine Fehlnachricht mit Details des Problems zurückgegeben. Wenn CICS als Web-Service-Provider fungiert, wird die Fehlnachricht an die Clientanwendung als JSON-Antwort zurückgegeben. Weitere Informationen finden Sie unter „An den Client zurückgegebene Fehlerantworten“ auf Seite 687. Wenn eine Anwendung die verknüpfbare Schnittstelle aufruft, um JSON umzusetzen, werden im Container DFHJSON-ERROR ein Fehlercode und im Container DFHJSON-ERRORMSG eine detaillierte Nachricht zurückgegeben. Weitere Informationen finden Sie unter JSON transformer linkable interface. In beiden Fällen sind die Schritte zur Fehlerbehebung identisch und hängen von der Art des Fehlers ab.

Procedure

- Prüfen Sie, dass Ihre JSON den Einschränkungen entspricht, die von CICS vorgegeben werden. Weitere Informationen finden Sie unter JSON web service restrictions.
- Wenn ein JSON-Parsing-Fehler auftritt, prüfen Sie, dass die JSON korrekt formatiert ist. Die Fehlnachricht enthält Details zu dem Problem, z. B.:
 "Expected a ',' or '}' at character 44 of {"inquireCatalogRequest":
 "myData } Sie können ein Tool verwenden, um die JSON-Syntax auszuwerten, z. B. JSONLint.
- Wenn CICS beim Versuch, JSON Anwendungsdaten zuzuordnen, einen strukturellen Fehler feststellt, wird eine Nachricht DFHPI1007 ausgegeben, die Details zu dem Fehler enthält. Beispiel:
 DFHPI1007 04/19/2013 15:14:42 IYK2ZKE1 00112 JSON to data transformation failed because of incorrect input (UNDEFINED_ELEMENT Operation) for WEBSERVICE SimpleMappings.
 Prüfen Sie, dass die JSON die JSON-Objekte und -Eigenschaften enthält, die in dem von Ihnen für DFHJS2LS bereitgestellten Schema oder in dem von DFHLS2JS generierten Schema beschrieben sind. Sie können ein Tool verwenden, um zu überprüfen, dass die JSON dem Schema entspricht.

- Wenn kein Fehler auftritt, aber Ihre Anwendung für manche Felder leere Daten empfängt, prüfen Sie, dass die entsprechende JSON bereitgestellt wurde. CICS wird das Fehlen von im Schema beschriebenen JSON-Eigenschaften nicht erkennen. Sie können ein JSON-Validierungstool verwenden, um dies wie bereits beschrieben zu prüfen.

- Ein interner Serverfehler tritt auf.

Internal Server Error

CICS TS: Unhandled Pipeline Error

Release: 670

Suchen Sie im Browser nach der Nachricht DFHSJ1006. Prüfen Sie, dass der JVM-Server vorhanden und nicht inaktiviert ist.

- Unter bestimmten Umständen fügt CICS während der internen Verarbeitung ein Wrapperelement zur JSON-Anforderung oder -Antwort hinzu. Dies ist für die Anwendung nie sichtbar, kann aber manchmal in Fehlnachrichten vorkommen. Beispiel:

```
"Error obtaining parser from data source:Expected a ':'  
after a key at character 25 of {"DFHWrapper": { 12jb01c\":"...
```

In diesen Fällen sollte das DFHWrapper-Element beim Bestimmen der Ursache des Fehlers ignoriert werden.

What to do next

Nachdem Sie Ihre JSON korrigiert haben, führen Sie die Anwendung erneut aus.

An den Client zurückgegebene Fehlerantworten

Diese Antworten werden zurückgegeben, wenn Fehler während der Verarbeitung im JSON-Handler von CICS auftreten.

About this task

Wenn während der Verarbeitung im JSON-Handler von CICS Fehler auftreten, gibt CICS eine Antwort mit Informationen zu dem Fehler an den Client zurück. Der HTTP-Statuscode 500 (Interner Serverfehler) wird zurückgegeben und der Nachrichtenhauptteil enthält Details zu dem Fehler im JSON-Format, abhängig von dem Typ von Fehler, der auftritt.

Example

Wenn ein Fehler auftritt, bevor oder nachdem CICS die Axis2-Pipeline aufgerufen hat, enthält die Nachricht Informationen ähnlich den folgenden:

```
{  
  "exception" : {  
    "message" : "An exception has occurred while validating HTTP headers".  
    "class" : "com.ibm.cicsts.axis2.Controller"  
  }  
}
```

Wenn ein Fehler an irgendeinem anderen Punkt während der Verarbeitung auftritt, enthält die Nachricht ähnliche Informationen wie ein SOAP-Fehler, mit einem Detailabschnitt, der abhängig von der Art des Fehlers variiert. Dies kann keine CICS-Nachricht wie die folgende umfassen:

```

{
  "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."
    }
  }
}

```

Rückgabecodes des JSON-Assistenten

Wenn bei der Ausführung des JSON-Assistenten ein Fehler auftritt, wird ein Rückgabecode bereitgestellt, der den Typ des Fehlers angibt. Diese Informationen sind im Jobprotokoll enthalten.

Rückgabecodes des JSON-Assistentenprogramms

Im Falle eines Implementierungsfehlers werden PIPELINE-Ressourcen in der Regel mit einem DISABLED-Status und WEBSERVICE-Ressourcen mit einem UNUSABLE-Status installiert. Informationen und Fehlernachrichten, die den Stapeljobs des CICS-JSON-Assistenten zugeordnet sind, befinden sich im Jobprotokoll. Fehlernachrichten, die der Installation von Ressourcen zugeordnet sind, werden im Systemprotokoll aufgezeichnet.

Die Codes 0, 4, 8 oder 12 werden von dem JSON-Assistenten ausgegeben, andere Codes stammen üblicherweise von BPXBATCH, der JVM oder IEBGENER.

Von BPXBATCH ausgegebene Codes fallen in zwei Hauptkategorien: ein Code kleiner 128 weist auf das Fehlschlagen eines Befehls hin, ein Code größer als 128 weist darauf hin, dass der Prozess durch ein Signal beendet wurde. Weitere Informationen zu BPXBATCH und den zugehörigen Rückgabecodes finden Sie in der z/OS UNIX System Services Command Reference.

Sie empfangen den Rückgabecode 0, 4, 8 oder 12, wenn Sie die Stapeljobs des JSON-Assistenten ausführen.

Rückgabecode	Beschreibung
0	Der Job wurde erfolgreich abgeschlossen.
4	Warnung. Der Job wurde erfolgreich abgeschlossen, es wurde jedoch mindestens eine Warnung ausgegeben.
8	Eingabefehler. Der Job wurde nicht erfolgreich abgeschlossen. Während der Validierung der Eingabeparameter wurde mindestens eine Warnung ausgegeben.
12	Fehler. Der Job wurde nicht erfolgreich abgeschlossen. Während der Ausführung wurde mindestens eine Warnung ausgegeben.

Rückgabecodes der Stapeljobs des JSON-Assistenten

Sie empfangen den Rückgabecode 1, 136, 137 oder 139, wenn Sie die Stapeljobs des CICS-JSON-Assistenten ausführen.

Rückgabecode	Beschreibung
1	Die JVM ist fehlgeschlagen, in der Regel, weil nicht genügend Speicher verfügbar ist. Für CICS-Assistenten ist eine JCL-Regionsgröße von mindestens 300 MB erforderlich.
136	Die JVM ist fehlgeschlagen, in der Regel, weil nicht genügend Speicher verfügbar ist. Für CICS-Assistenten ist eine JCL-Regionsgröße von mindestens 300 MB erforderlich.
137	Der Job hat das zulässige Zeitlimit überschritten.
139	Die JVM ist fehlgeschlagen, in der Regel, weil nicht genügend Speicher verfügbar ist. Für CICS-Assistenten ist eine JCL-Regionsgröße von mindestens 300 MB erforderlich.

Anhang A. Container für verknüpfbare Schnittstelle des JSON-Umsetzungsprogramms

Die verknüpfbare Schnittstelle des JSON-Umsetzungsprogramms ist ein von CICS bereitgestelltes Programm, das aufgerufen werden kann, um Datenumsetzungen in und aus JSON durchzuführen. Sie kann JSON-Daten aus strukturierten Anwendungsdaten bzw. strukturierte Anwendungsdaten aus JSON-Daten erstellen.

Sie können die verknüpfbare Schnittstelle verwenden, um JSON aus einer beliebigen Quelle umzusetzen - ob über ein Protokoll wie WebSphere MQ oder aus einer JSON-Datenbank. Anders dagegen die JSON-Web-Service-Unterstützung, die nur HTTP und Providermodus zulässt.

Weitere Informationen zu JSON finden Sie unter Konzepte von JSON-Web-Services.

Um die verknüpfbare Schnittstelle des JSON-Umsetzungsprogramms zu verwenden, muss das Anwendungsprogramm über einen Kanal eine Verknüpfung mit dem von CICS bereitgestellten Programm DFHJSON herstellen. Container können verwendet werden, um Eingabedaten an DFHJSON zu übergeben, und um Ausgabedaten von DFHJSON zu empfangen. Details hierzu finden Sie in den Unterthemen.

CICS bietet zwei Mechanismen für das Umsetzen von strukturierten Anwendungsdaten in und aus JSON-Daten: mit einem JVM-Server und ohne einen JVM-Server. Der Mechanismus mit einem JVM-Server bietet die Vorteile der Java-Plattform (wie Berechtigung für zEnterprise Application Assist Processors (zAAP), falls verfügbar). Für den Mechanismus ohne Java muss kein JVM-Server konfiguriert werden. Die Anwendung entscheidet sich mithilfe des Containers DFHJSON_JVMSERVER für einen der beiden Mechanismen. Der Inhalt identifiziert den JVM-Server für die Umsetzung der Daten, das Fehlen dieses Containers weist darauf hin, dass der Nicht-Java-Umsetzungsservice verwendet wird.

Um diesen Service verwenden zu können, müssen Sie über einen passenden JSON-TRANSFRM-Bundleteil verfügen, der in CICS installiert ist. Dieser wird mithilfe der JSON-Assistenten generiert. Weitere Informationen finden Sie unter Mapping and transforming application data and JSON.

Container DFHJSON-JSON

DFHJSON-JSON ist ein Container vom Typ DATATYPE(CHAR).

- Wenn Sie Anwendungsdaten in JSON umsetzen: Dieser Container enthält die JSON, die von dem Umsetzungsprogramm entsprechend der Anwendungsdateneingabe erstellt wurde.
- Wenn Sie JSON in Anwendungsdaten umwandeln: Dieser Container enthält die JSON, die von der Benutzeranwendung bereitgestellt wird, die umgesetzt werden soll.

Container DFHJSON-DATA

DFHJSON-DATA ist ein Container vom Typ DATATYPE(BIT).

- Wenn Sie Anwendungsdaten in JSON umsetzen: Dieser Container enthält die Anwendungsdaten, die vom Benutzer bereitgestellt werden, um in JSON umgesetzt zu werden.
- Wenn Sie JSON in Anwendungsdaten umsetzen: Dieser Container enthält die Anwendungsdaten, die von den umgesetzten Daten erstellt werden, die der JSON-Eingabe entsprechen.

Container DFHJSON-TRANSFRM

DFHJSON-TRANSFRM ist ein Container vom Typ DATATYPE(CHAR). Er enthält den 16 Zeichen langen Namen des JSONTRANSFRM-Bundleteils, dessen Zuordnungen angewendet werden, um die Daten umzusetzen.

JSONTRANSFRM ist ein Bundleteil. Er ist in einem Bundle enthalten, das mithilfe des JSON-Assistenten generiert wird.

Weitere Informationen zur Anzeige von Bundleteilen finden Sie unter *Bundle Parts view in the CICS Explorer product documentation*.

Alternativ können Sie den Befehl **INQUIRE BUNDLEPART** verwenden. Siehe **INQUIRE BUNDLEPART**.

Container DFHJSON-JVMSEVR

DFHJSON-JVMSEVR ist ein Container vom Typ DATATYPE(CHAR). Er enthält den acht Zeichen langen Namen des JVM-Servers (JVMSEVR), auf dem das Zielprogramm ausgeführt werden soll.

Wenn der Container nicht bereitgestellt wird oder wenn der Container vorhanden ist, aber eine Länge von null hat, wird die Datenumsetzung in CICS statt auf einem JVM-Server ausgeführt. Wenn Sie eine Umsetzung auf dem von CICS-bereitgestellten Axis2-JVM-Server ausführen möchten, geben Sie den Wert DFHA-XIS in diesem Container an.

Container DFHJSON-ERROR

DFHJSON-ERROR ist ein Container vom Typ DATATYPE(BIT). Fehler, die während der Validierung oder Ausführung von DFHJSON generiert werden, werden als Vollwort-Binärwert im Container DFHJSON-ERROR zurückgegeben.

Die folgenden Fehler können von DFHJSON zurückgegeben werden:

- 1 Der Container DFHJSON-TRANSFRM war nicht vorhanden.
- 2 Der Name von JSONTRANSFRM, bereitgestellt im Container DFHJSON-TRANSFRM, war länger als 16 Zeichen.
- 3 Ein Container hatte den falschen Datentyp. DFHJSON-ERRORMSG enthält den Namen des falschen Containers.
- 4 Der Name von JVMSEVR, bereitgestellt im Container DFHJSON-JVMSEVR, war länger als acht Zeichen.
- 11 JSONTRANSFRM, genannt im Container DFHJSON-TRANSFRM, wurde nicht gefunden.
- 12 JSONTRANSFRM, genannt im Container DFHJSON-TRANSFRM, hatte nicht den Status ENABLED.

- 13 Weder der Container DFHJSON-DATA noch der Container DFHJSON-JSON ist vorhanden; einer der beiden Container muss angegeben sein.
- 14 Sowohl der Container DFHJSON-DATA als auch der Container DFHJSON-JSON war vorhanden; es darf nur einer der beiden Container angegeben sein.
- 15 Ein Datenumsetzungsfehler ist aufgetreten. DFHJSON-ERRORMSG enthält Details des Fehlers.
- 16 Während des Parsens von JSON, bereitgestellt im Container DFHJSON-JSON, ist ein Fehler aufgetreten. DFHJSON-ERRORMSG enthält Details des Fehlers.
- 17 Im Java-Teil des Umsetzungsprogramms ist ein Fehler aufgetreten. Bitten Sie den IBM Support um Unterstützung.
- 20 Ein unerwarteter Fehler ist aufgetreten. Bitten Sie den IBM Support um Unterstützung.
- 51 Der JVM-Server, der im Container DFHJSON-JVMSEVR genannt wird, wurde nicht gefunden.
- 52 Der JVM-Server, der im Container DFHJSON-JVMSEVR genannt wird, hatte nicht den Status ENABLED.

Container DFHJSON-ERRORMSG

DFHJSON-ERRORMSG ist ein Container vom Typ DATATYPE(CHAR). Dieser Container wird erstellt, wenn während der Verarbeitung ein Fehler auftritt. Er enthält eine lesbare zusätzliche Fehlernachricht, z. B. die Fehlernachricht vom JSON-Parser. Diese Nachricht wird nur gefüllt, wenn zusätzliche Details erforderlich sind.

Weitere Informationen zu Fehlercodes finden Sie unter „Container DFHJSON-ERROR“ auf Seite 692.

Anhang B. Web-Service-Beispiele

CICS stellt eine Reihe von Beispielen bereit, die Sie als Ausgangspunkt für die Entwicklung von Anwendungen und für die Konfiguration von CICS verwenden können.

Die Beispiele sind wie folgt kategorisiert:

Beispielanwendung des CICS-Katalogmanagers

Die Beispielanwendung des CICS-Katalogmanagers ist eine funktionierende COBOL-Anwendung, die bewährte Verfahren für die Herstellung von Verbindungen von CICS-Anwendungen zu externen Clients und Servern veranschaulichen soll.

Die Basisanwendung hat eine 3270-Benutzerschnittstelle, aber die modulare Struktur mit klar definierten Schnittstellen zwischen den Komponenten ermöglicht es, weitere Komponenten hinzuzufügen. Insbesondere bietet die Anwendung Web-Service-Unterstützung, die veranschaulichen soll, wie Sie eine vorhandene Anwendung in eine Web-Service-Umgebung erweitern können.

- „Beispielanwendung des CICS-Katalogmanagers“

JSON-Beispiele

Diese Beispiele erleichtern Ihnen das Verständnis von JSON-Anforderungen.

- „JSON-Beispiele“ auf Seite 735

Beispielanwendung des CICS-Katalogmanagers

Die Beispielanwendung des CICS-Katalogmanagers ist eine funktionierende COBOL-Anwendung, die bewährte Verfahren für die Herstellung von Verbindungen von CICS-Anwendungen zu externen Clients und Servern veranschaulichen soll.

Das Beispiel ist um einen einfachen Vertriebskatalog und eine einfache Bestellabwicklungsanwendung herum konzipiert, in der ein Benutzer die folgenden Tasks ausführen kann:

- Listen Sie die Artikel in einem Katalog auf.
- Fragen Sie einzelne Artikel im Katalog ab.
- Bestellen Sie Artikel aus dem Katalog.

Der Katalog wird als VSAM-Datei implementiert.

Die Basisanwendung hat eine 3270-Benutzerschnittstelle, aber die modulare Struktur mit klar definierten Schnittstellen zwischen den Komponenten ermöglicht es, weitere Komponenten hinzuzufügen. Insbesondere bietet die Anwendung Web-Service-Unterstützung, die veranschaulichen soll, wie Sie eine vorhandene Anwendung in eine Web-Service-Umgebung erweitern können.

Bei diesem Beispiel könnte CICS Explorer verwendet werden, um die Anwendung zu installieren und zu implementieren. CICS Explorer ist eine Eclipse-basierte grafische Toolschnittstelle für CICS.

Optional: Benutzer, die nicht mit CICS Explorer arbeiten, können die CEDA-Transaktion verwenden, um die in den Gruppen DFH\$EXBS und DFH\$EXWS enthaltenen Ressourcen zu ändern und zu installieren.

Basisanwendung

Die Basisanwendung stellt zusammen mit ihrer 3270-Benutzerschnittstelle Funktionen zum Auflisten des Inhalts eines gespeicherten Katalogs, zur Auswahl eines Artikels aus der Liste und zur Eingabe einer Bestellmenge bereit. Die Anwendung hat ein modulares Design, das ein Erweitern der Anwendung um neuere Technologien, z. B. Web-Services, einfach macht.

Abb. 30 zeigt die Struktur der Basisanwendung. Die Komponenten der Basisanwendung sind:

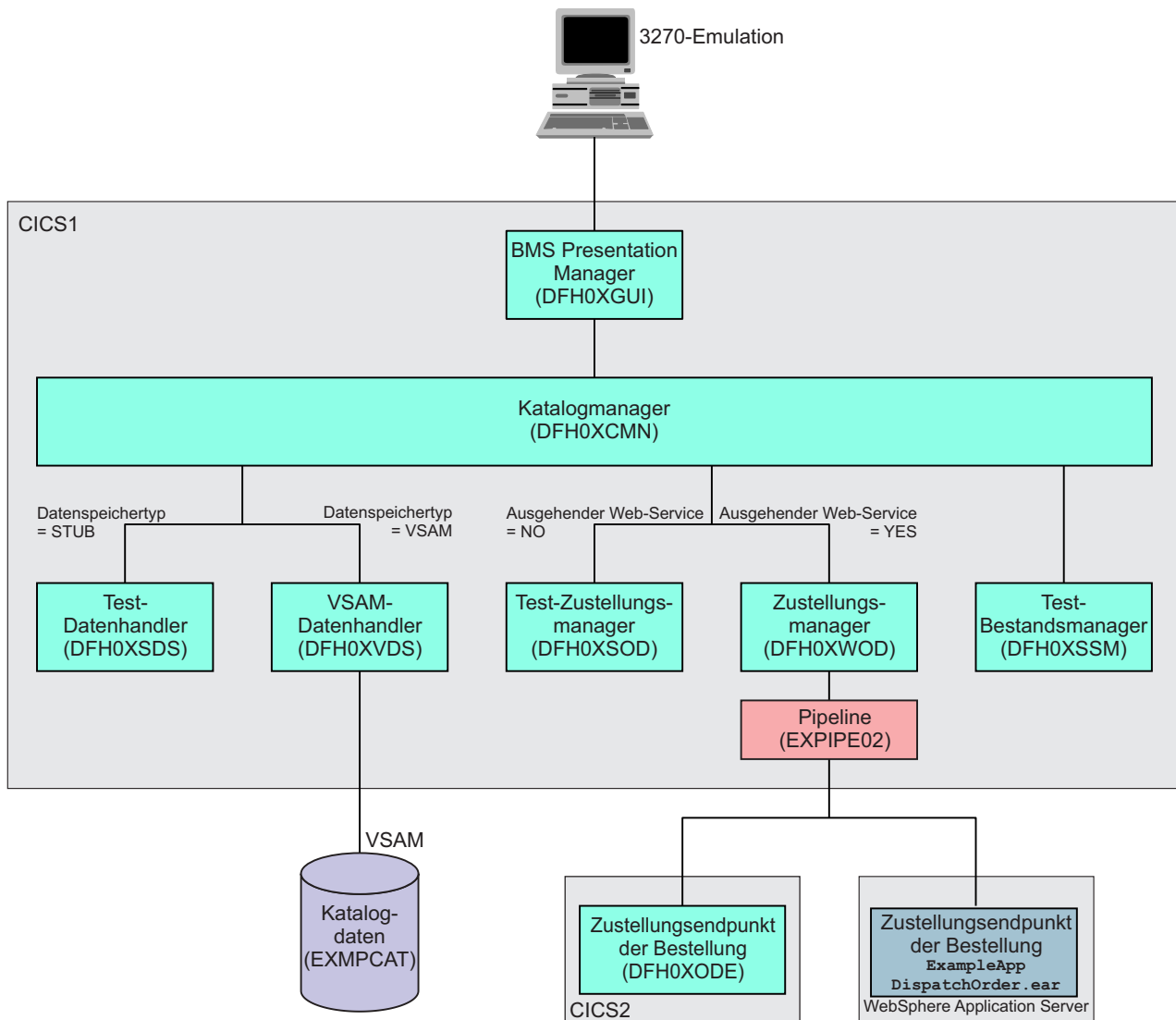


Abbildung 30. Struktur der Basisanwendung

- Ein BMS-Presentation Manager (DFH0XGUI), der ein 3270-Terminal oder einen 3270-Emulator unterstützt und mit dem Katalogmanager-Hauptprogramm interagiert.
- Ein Katalogmanagerprogramm (DFH0XCMN), das den Kern der Beispielanwendung darstellt und mit mehreren Back-End-Komponenten interagiert.

- Ein Datenhandlerprogramm, das die Schnittstelle zwischen dem Katalogmanagerprogramm und dem Datenspeicher bereitstellt. Die Basisanwendung stellt zwei Versionen dieses Programms zur Verfügung. Dies sind das VSAM-Datenhandlerprogramm (DFH0XVDS), das Daten in einer VSAM-Datei speichert, und ein Testdatenhandler (DFH0XSDS), der keine Daten speichert, aber gültige Antworten an den Aufrufenden zurückgibt. In der Konfiguration können Sie zwischen den beiden Programmen wählen.
- Ein Zustellungsmanagerprogramm, das eine Schnittstelle für die Zustellung einer Bestellung an einen Kunden bereitstellt. Auch hier können Sie in der Konfiguration zwischen den beiden Versionen dieses Programms wählen: DFHX0WOD ist ein Web-Service-Requester, der einen fernen Bestellzustellungsendpoint aufruft, und DFHX0SOD ist ein Testprogramm, das gültige Antworten an den Aufrufenden zurückgibt.
Es gibt zwei äquivalente Bestellzustellungsendpunkte: DFH0XODE ist ein CICS-Service-Provider-Programm, ExampleAppDispatchOrderV855.war ist eine Java-Webarchivdatei, die auf dem CICS-Liberty-JVM-Server oder in ähnlichen Umgebungen implementiert werden kann.
- Ein Bestandsmanager-Testprogramm (DFH0XSSM), das gültige Antworten an den Aufrufenden zurückgibt, aber keine Aktionen ausführt.

BMS Presentation Manager

Der Presentation Manager ist verantwortlich für alle Interaktionen mit dem Benutzer über 3270-BMS-Anzeigen. In diesem Programm werden keine Geschäftsentcheidung getroffen.

Der BMS Presentation Manager kann auf zwei Arten verwendet werden:

- Als Teil der Basisanwendung.
- Als CICS-Web-Service-Client, der mit der Basisanwendung unter Verwendung von SOAP-Nachrichten kommuniziert.

Datenhandler

Der Datenhandler stellt die Schnittstelle zwischen dem Katalogmanager und dem Datenspeicher bereit.

Die Beispielanwendung stellt zwei Versionen des Datenhandlers bereit:

- Die erste Version verwendet eine VSAM-Datei als Datenspeicher.
- Die zweite Version ist ein Testprogramm, das immer dieselben Daten auf eine Anfrage zurückgibt und die Ergebnisse von Aktualisierungsanforderungen nicht speichert.

Zustellungsmanager

Der Zustellungsmanager ist verantwortlich für das Zustellen der Bestellung an den Kunden, nachdem die Bestellung bestätigt wurde.

Die Beispielanwendung stellt zwei Versionen des Zustellungsmanagerprogramms bereit:

- Die erste Version ist ein Testprogramm, das eine korrekte Antwort an den Aufrufenden zurückgibt, aber keine weitere Aktion ausführt.
- Die zweite Version ist ein Web-Service-Requester-Programm, das eine Anforderung an die Endpunktadresse stellt, die in der Konfigurationsdatei definiert ist.

Bestellszustellungsprogramm

Das Bestellzustellungsprogramm ist ein Web-Service-Provider-Programm, das für die Zustellung des Artikels an den Kunden verantwortlich ist.

In der Beispielanwendung ist der Bestellsdispatcher ein Testprogramm, das eine korrekte Antwort an den Aufrufenden zurückgibt, aber keine weitere Aktion ausführt. Er macht es möglich, dass alle Konfigurationen der Beispiel-Web-Services funktionsbereit sein.

Bestandsmanager

Der Bestandsmanager ist verantwortlich für das rechtzeitige Auffüllen des Bestands.

In dem Beispielprogramm ist der Bestandsmanager ein Testprogramm, das eine korrekte Antwort an den Aufrufenden zurückgibt, aber keine weitere Aktion ausführt.

Anwendungskonfiguration

Die Beispielanwendung umfasst ein Programm, mit dem Sie die Basisanwendung konfigurieren können.

Basisanwendung installieren und konfigurieren

Bevor Sie die Basisanwendung ausführen können, müssen Sie zwei VSAM-Datensätze definieren und füllen, und Sie müssen zwei TRANSACTION-Ressourcen erstellen.

VSAM-Dateien erstellen und definieren

Zwei KSDS-VSAM-Dateien werden zum Definieren und Füllen der Beispielanwendung verwendet. Eine Datei enthält Konfigurationsinformationen für die Beispielanwendung. Die andere enthält den Verkaufskatalog.

Procedure

1. Machen Sie die JCL ausfindig, um die VSAM-Datei zu erstellen. Während der CICS-Installation wird die JCL in der Bibliothek *hlq.SDFHINST* platziert:
 - Member *DFH\$ECNF* enthält die JCL zum Generieren der Konfigurationsdatei.
 - Member *DFH\$ECAT* enthält die JCL zum Generieren der Katalogdatei.
2. Ändern Sie die JCL und Zugriffsmethodenservices-Befehle.
 - a. Geben Sie eine gültige JOB-Karte an.
 - b. Geben Sie ein geeignetes übergeordnetes Qualifikationsmerkmal für die Dateinamen in den Zugriffsmethodenservices-Befehlen ein. Die JCL verwendet ein übergeordnetes Qualifikationsmerkmal *HLQ*.

Der folgende Befehl definiert die Konfigurationsdatei:

```
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) -  
    )
```

Dabei ist *hlq* ein übergeordnetes Qualifikationsmerkmal Ihrer Wahl.

Der folgende Befehl definiert die Katalogdatei:

```
DEFINE CLUSTER (NAME(hlq.EXMPLAPP.catname) -  
    TRK(1 1) -  
    KEYS(4 0) -  
    RECORDSIZE(80,80) -
```

```

SHAREOPTIONS(2 3) -
INDEXED -
) -
DATA (NAME(hlq.EXMPLAPP.catname.DATA) -
) -
INDEX (NAME(hlq.EXMPLAPP.catname.INDEX) -
)

```

Dabei gilt Folgendes:

- *hlq* ist ein übergeordnetes Qualifikationsmerkmal Ihrer Wahl.
 - *catname* ist ein Name Ihrer Wahl. Der in der Beispielanwendung verwendete Name ist EXMPCAT.
3. Führen Sie beide Jobs aus, um die Dateien zu erstellen und zu füllen.
 4. Verwenden Sie CICS Explorer, um eine FILE-Definition für die Katalogdatei zu erstellen.
 - a. Wählen Sie **Definitionen** > **Dateideinitionen** aus. Klicken Sie mit der rechten Maustaste in die Spalte **Name** und klicken Sie auf **Neu**, um eine neue Dateidefinition zu erstellen. Geben Sie EXAMPLE im Textfeld **Ressourcengruppe** und EXMPCAT im Textfeld **Name** ein. Klicken Sie auf **Fertigstellen**, um die FILE-Definition zu definieren. Alternativ können Sie die Dateidefinition aus der von CICS bereitgestellten Gruppe DFH\$EXBS kopieren.
 - b. Doppelklicken Sie auf die neue EXMPCAT-Datei. Wählen Sie im Editor CICS-Beispielanwendung für Dateidefinition (EXMPCAT) die Registerkarte **VSAM** aus. Geben Sie *hlq*.EXMPLAPP.EXMPCAT im Textfeld **Zu verwendender Dateiname** ein.
 - c. Wählen Sie die Registerkarte **Attribute** aus und setzen Sie die Operationen der folgenden Attribute auf **Ja**:
 - Hinzufügen
 - Durchsuchen
 - Löschen
 - Lesen
 - Aktualisieren
 - d. Verwenden Sie die Standardwerte für alle anderen Attribute.
 5. Verwenden Sie CICS Explorer, um eine FILE-Definition für die Konfigurationsdatei zu erstellen.
 - a. Wählen Sie **Definitionen** > **Dateideinitionen** aus. Klicken Sie mit der rechten Maustaste in die Spalte **Name** und klicken Sie auf **Neu**, um eine neue Dateidefinition zu erstellen. Geben Sie EXAMPLE im Textfeld **Ressourcengruppe** und EXMPCONF im Textfeld **Name** ein. Klicken Sie auf **Fertigstellen**, um die FILE-Definition zu definieren. Alternativ können Sie die Dateidefinition aus der von CICS bereitgestellten Gruppe DFH\$EXBS kopieren.
 - b. Doppelklicken Sie auf die neue Datei EXMPCONF. Wählen Sie im Fenster CICS-Beispielanwendung für Dateidefinition (EXMPCONF) die Registerkarte **VSAM** aus. Geben Sie *hlq*.EXMPLAPP.EXMPCONF im Textfeld **Zu verwendender Dateiname** ein.
 - c. Wählen Sie die Registerkarte **Attribute** aus und setzen Sie die Operationen der folgenden Attribute auf **Ja**:
 - Hinzufügen
 - Durchsuchen
 - Löschen
 - Lesen
 - Aktualisieren

- d. Verwenden Sie die Standardwerte für alle anderen Attribute.

Results

Die Dateien werden gefüllt, und die FILE-Definitionen für die Katalogdatei und die Konfigurationsdatei wurden erstellt und können installiert werden.

Optional: Benutzer, die nicht mit CICS Explorer arbeiten, können die CEDA-Transaktion verwenden, um die in den Gruppen DFH\$EXBS und DFH\$EXWS enthaltenen Ressourcen zu ändern und zu installieren.

3270-Schnittstelle definieren

Die Beispielanwendung wird mit einer 3270-Benutzerschnittstelle geliefert, in der Sie die Anwendung ausführen und anpassen können. Die Benutzerschnittstelle besteht aus zwei Transaktionen, EGUI und ECFG. Eine dritte Transaktion, ECLI, wird für den CICS-Web-Service-Client verwendet.

About this task

Optional: Benutzer, die nicht mit CICS Explorer arbeiten, können die CEDA-Transaktion verwenden, um die in den Gruppen DFH\$EXBS und DFH\$EXWS enthaltenen Ressourcen zu ändern und zu installieren.

Procedure

1. Erstellen Sie Transaktionsdefinitionen für die folgenden Transaktionen unter Verwendung von CICS Explorer. Der korrekte Betrieb der Beispielanwendung hängt nicht von den Namen der Transaktionen an, Sie können folglich andere Namen wählen.

EGUI

- a. Kopieren Sie die Definitionen für die Transaktion EGUI aus der von CICS bereitgestellten Gruppe DFH\$EXBS, indem Sie mit der rechten Maustaste in der Ansicht Ressourcengruppendefinitionen auf DFH\$EXBS klicken.
- b. Wählen Sie **Neu > Transaktionsdefinition** aus.
- c. Geben Sie EGUI im Textfeld **Name** ein, geben Sie einen Namen im Textfeld **Ressourcengruppe/CSD-Gruppe** und DFH0XGUI im Textfeld **Programmname** an.
- d. Klicken Sie auf **Fertigstellen**, um die EGUI-Transaktionsdefinition zu erstellen.

Verwenden Sie die Standardwerte für alle anderen Attribute.

ECFG

- a. Kopieren Sie die Definitionen für die Transaktion ECFG aus der von CICS bereitgestellten Gruppe DFH\$EXBS, indem Sie mit der rechten Maustaste im Fenster Ressourcengruppendefinitionen auf DFH\$EXBS klicken.
- b. Wählen Sie **Neu > Transaktionsdefinition** aus.
- c. Geben Sie ECFG im Textfeld **Name** ein, geben Sie einen Namen im Textfeld **Ressourcengruppe/CSD-Gruppe** und DFH0XCFG im Textfeld **Programmname** an.
- d. Klicken Sie auf **Fertigstellen**, um die ECFG-Transaktionsdefinition zu erstellen.

Verwenden Sie die Standardwerte für alle anderen Attribute.

ECLI (optional)

- a. Kopieren Sie die Definitionen für ECLI-Transaktionen aus der von CICS bereitgestellten Gruppe DFH\$EXWS, indem Sie mit der rechten Maustaste in der Ansicht Ressourcengruppendefinitionen auf DFH\$EXWS klicken.
- b. Wählen Sie **Neu > Transaktionsdefinition** aus.
- c. Geben Sie ECLI im Textfeld **Name** ein, geben Sie einen Namen im Textfeld **Ressourcengruppe/CSD-Gruppe** und DFH0XCUI im Textfeld **Programmname** an.
- d. Klicken Sie auf **Fertigstellen**, um die ECLI-Transaktionsdefinition zu erstellen.

Verwenden Sie die Standardwerte für alle anderen Attribute.

2. Optional: Wenn Sie das Programm nicht automatisch installieren möchten, kopieren Sie die PROGRAM-Definitionen für die Basisanwendungsprogramme und die MAPSET-Definitionen für die BMS-Masken aus der von CICS bereitgestellten Gruppe DFH\$EXBS.
 - a. Kopieren Sie die MAPSET-Ressourcendefinitionen für die BMS-Masken in den Membern DFH0XS1, DFH0XS2 und DFH0XS3. Detaillierte Angaben zu den einzelnen Membern finden Sie unter „Komponenten der Basisanwendung“ auf Seite 725.
 - b. Kopieren Sie die PROGRAM-Ressourcendefinitionen für die folgenden COBOL-Programme.

Tabelle 16. SDFHSAMP-Member mit COBOL-Quelle für die Basisanwendung

Membername	Beschreibung
DFH0XCFG	Durch Transaktions-ECFG aufgerufenes Programm zum Lesen und Aktualisieren der VSAM-Konfigurationsdatei.
DFH0XCMN	Controllerprogramm für die Kataloganwendung. Alle Anforderungen durchlaufen das Controllerprogramm.
DFH0XGUI	Durch Transaktion-EGUI aufgerufenes Programm zum Verwalten des Sendens der BMS-Masken an den Terminalbenutzer und des Empfangens der Masken vom Terminalbenutzer. Dieses Programm ist mit dem Programm DFH0XCMN verknüpft.
DFH0XODE	Eine von zwei Versionen des Endpunkts für den Web-Service für die Bestellzustellung. Dies ist die Version, die in CICS ausgeführt wird. Dieses Programm legt den Text "Order in dispatch" (Bestellung in Zustellung) im Rückgabekommunikationsbereich fest.
DFH0XSDS	Eine <i>Testversion</i> des Datenspeicherprogramms, mit der die Anwendung auch dann ausgeführt werden kann, wenn die VSAM-Katalogdatei nicht eingerichtet wurde. DFH0XSDS verwendet im Programm definierte Daten statt in einer VSAM-Datei gespeicherte Daten.
DFH0XSOD	Eine Testversion des Programms zur Bestellzustellung. Es legt den Rückgabecode im Kommunikationsbereich auf '0' fest und kehrt zum Aufrufer zurück. DFH0XSOD wird verwendet, wenn keine ausgehenden Web-Services erforderlich sind.
DFH0XSSM	Eine Testversion des Bestandsmanagerprogramms (Nachbestellungsprogramms). DFH0XSSM legt den Rückgabecode im Kommunikationsbereich auf '0' fest und kehrt zum Aufrufer zurück.

Tabelle 16. SDFHSAMP-Member mit COBOL-Quelle für die Basisanwendung (Forts.)

Membername	Beschreibung
DFH0XVDS	Die VSAM-Version des Datenspeicherprogramms. DFH0XVDS greift auf die VSAM-Datei zu, um Lesevorgänge oder Aktualisierungen des Katalogs auszuführen.
DFH0XWOD	Die Web-Service-Version des Programms zur Bestellzustellung. DFH0XWOD gibt den Befehl EXEC CICS INVOKE WEBSERVICE aus, um einen ausgehenden Web-Service-Aufruf eines Bestelldispatchers zu starten.

Verwenden Sie die Standardwerte für alle anderen Attribute.

- c. Optional: Kopieren Sie die PROGRAM-Definitionen für DFH0XCUI aus der von CICS bereitgestellten Gruppe DFH\$EXWS. Verwenden Sie die Standardwerte für alle anderen Attribute. Dieses Programm ist erforderlich, wenn Sie die Transaktion ECLI verwenden möchten, die den Web-Service-Client startet.

```
DIS G(DFH$EXWS)
ENTER COMMANDS
NAME      TYPE      GROUP
DFH0XCUI  PROGRAM    DFH$EXWS
ECLI      TRANSACTION DFH$EXWS
EXMPPORT  TCIPSERVICE DFH$EXWS
EXPIPE01  PIPELINE    DFH$EXWS
EXPIPE02  PIPELINE    DFH$EXWS
```

Installation abschließen

Um die Installation abzuschließen, installieren Sie die Onlineressourcendefinitionsgruppe, die Ihre Ressourcendefinitionen enthält.

Procedure

Klicken Sie im Fenster Ressourcengruppendefinitionen mit der rechten Maustaste auf die Ressourcengruppe. Wählen Sie **Installieren** aus. Stellen Sie sicher, dass Ihr CICSplex korrekt und Ihre Zielregion ausgewählt ist, und klicken Sie anschließend auf **OK**.

Results

Ihre Onlineressourcendefinition ist jetzt installiert und die Anwendung kann verwendet werden.

Optional: Benutzer, die nicht mit CICS Explorer arbeiten, können die CEDA-Transaktion verwenden, um die in den Gruppen DFH\$EXBS und DFH\$EXWS enthaltenen Ressourcen zu ändern und zu installieren.

Beispielanwendung konfigurieren

Die Basisanwendung umfasst eine Transaktion (ECFG), die Sie verwenden können, um die Beispielanwendung zu konfigurieren.

Before you begin

Die Konfigurationstransaktion verwendet Buchstaben in Groß-/Kleinschreibung. Sie müssen folglich ein Terminal verwenden, das Groß-/Kleinschreibung korrekt verarbeiten kann.

About this task

Sie können eine Reihe von Aspekten der Beispielanwendung angeben. Dazu gehören:

- Die allgemeine Konfiguration der Anwendung, z. B. die Verwendung von Web-Services
- Die Netzadressen, die von den Web-Service-Komponenten der Anwendung verwendet werden
- Die Namen der Ressourcen, z. B. die Datei, die für den Datenspeicher verwendet wird
- Die Namen der Programme, die für die einzelnen Komponenten der Anwendung verwendet werden

Mithilfe der Konfigurationstransaktion können Sie von CICS bereitgestellte Komponenten der Beispielanwendung durch Ihre eigenen ersetzen, ohne die Anwendung erneut starten zu müssen.

Procedure

1. Geben Sie das Transaktions-ECFG ein, um die Konfigurationsanwendung zu starten. CICS öffnet die folgende Anzeige:

CICS-BEISPIELANWENDUNG KONFIGURIEREN

```
Datenspeichertyp ==> VSAM                STUB|VSAM
Ausgehender Web-Service? ==> NO          YES|NO
Katalogmanager ==> DFH0XCMN
Datenspeicher-Stub ==> DFH0XSDS
Datenspeicher-VSAM ==> DFH0XVDS
Bestellzustellung-Stub ==> DFH0XSOD
Bestellzustellung-Web-Service ==> DFH0XWOD
Bestandsmanager ==> DFH0XSSM
VSAM-Dateiname ==> EXMPCAT
Serveradresse und -port ==> mein_server:99999
Ausgehender -Web-Service-URI ==> http://mein_server:80/exampleApp/dispatchOrder
==>
==>
==>
==>
==>
```

PF

3 END

12 CNCL

2. Füllen Sie die Felder aus.

Datenspeichertyp

Geben Sie STUB an, wenn Sie das Datenspeicher-Stubprogramm verwenden wollen.

Geben Sie VSAM an, wenn Sie das VSAM-Datenspeicherprogramm verwenden wollen.

Ausgehender Web-Service?

Geben Sie YES an, wenn Sie einen fernen Web-Service für Ihre Bestellzustellungsfunktion verwenden wollen, d. h. wenn Sie wollen, dass das Katalogmanagerprogramm mit dem Web-Service-Programm für Bestellzustellung verknüpft ist.

Geben Sie N0 an, wenn Sie ein Stubprogramm für Ihre Bestellzustellungsfunktion verwenden wollen, d. h. wenn Sie wollen, dass das Katalogmanagerprogramm mit dem Stubprogramm für Bestellzustellung verknüpft ist.

Katalogmanager

Geben Sie den Namen des Katalogmanagerprogramms an. Das mit der Beispielanwendung bereitgestellte Programm ist DFH0XCMN.

Datenspeicher-Stub

Wenn Sie STUB im Feld **Datenspeichertyp** angegeben haben, geben Sie hier den Namen des Datenspeicher-Stubprogramms an. Das mit der Beispielanwendung bereitgestellte Programm ist DFH0XSDS.

Datenspeicher-VSAM

Wenn Sie VSAM im Feld **Datenspeichertyp** angegeben haben, geben Sie hier den Namen des VSAM-Datenspeicherprogramms an. Das mit der Beispielanwendung bereitgestellte Programm ist DFH0XVDS.

Bestellzustellung-Stub

Wenn Sie N0 im Feld **Ausgehender Web-Service** angegeben haben, geben Sie hier den Namen des Stubprogramms für Bestellzustellung an. Das mit der Beispielanwendung bereitgestellte Programm ist DFH0XSOD.

Bestellzustellung-Web-Service

Wenn Sie YES im Feld **Ausgehender Web-Service** angegeben haben, geben Sie hier den Namen des Programms an, das als Service-Requester fungiert. Das mit der Beispielanwendung bereitgestellte Programm ist DFH0XWOD.

Bestandsmanager

Geben Sie den Namen des Bestandsmanagerprogramms an. Das mit der Beispielanwendung bereitgestellte Programm ist DFH0XSSM.

VSAM-Dateiname

Wenn Sie VSAM im Feld **Datenspeichertyp** angegeben haben, geben Sie hier den Namen der CICS-Dateidefinition an. Der in der Beispielanwendung verwendete Name ist EXMPCAT.

Serveradresse und -port

Wenn Sie den CICS-Web-Service-Client verwenden, geben Sie die IP-Adresse und den Port des Systems an, auf dem die Beispielanwendung als Web-Service implementiert wird.

Ausgehender Web-Service-URI

Wenn Sie YES im Feld **Ausgehender Web-Service** angegeben haben, geben Sie die Position des Web-Service an, der die Funktion zur Bestellzustellung implementiert. Wenn Sie den von CICS bereitgestellten Endpunkt verwenden, legen Sie **Ausgehender Web-Service** auf `http://mein_server:mein_port/exampleApp/dispatchOrder` fest, wobei *mein_server* und *mein_port* Ihre CICS-Serveradresse und der zugehörige Serverport sind.

Beispielanwendung mit der BMS-Schnittstelle ausführen

Die Basisanwendung kann mithilfe der BMS-Schnittstelle ausgeführt werden.

Procedure

1. Führen Sie die Transaktion EGUI über ein CICS-Terminal aus. Das Beispielanwendungsmenü wird angezeigt:

CICS-BEISPIELKATALOGANWENDUNG - Hauptmenü

Wählen Sie eine Aktion aus, drücken Sie dann die Eingabetaste.

Aktion 1. Artikel auflisten
 2. Bestellartikelnummer ____
 3. Beenden

F3=EXIT F12=CANCEL

Sie können die Artikel im Katalog auflisten, einen Artikel bestellen oder die Anwendung schließen.

2. Geben Sie 1 ein und drücken Sie die Eingabetaste, um die Option Artikel auflisten auszuwählen. Die Anwendung zeigt eine Liste der Artikel in dem Katalog an.

CICS-BEISPIELKATALOGANWENDUNG - Katalog abfragen

Wählen Sie einen einzelnen zu bestellenden Artikel mit / aus und drücken Sie dann die Eingabetaste.

Artikel	Beschreibung	Preis	Bestellung
0010	Ball Pens Black 24pk	2,90	/
0020	Ball Pens Blue 24pk	2,90	—
0030	Ball Pens Red 24pk	2,90	—
0040	Ball Pens Green 24pk	2,90	—
0050	Pencil with eraser 12pk	1,78	—
0060	Highlighters Assorted 5pk	3,89	—
0070	Laser Paper 28-lb 108 Bright 500/ream	7,44	—
0080	Laser Paper 28-lb 108 Bright 2500/case	33,54	—
0090	Blue Laser Paper 20lb 500/ream	5,35	—
0100	Green Laser Paper 20lb 500/ream	5,35	—
0110	IBM Network Printer 24 - Toner cart	169,56	—
0120	Standard Diary: Week to view 8 1/4x5 3/4	25,99	—
0130	Wall Planner: Eraseable 36x24	18,85	—
0140	70 Sheet Hard Back wire bound notepad	5,89	—
0150	Sticky Notes 3x3 Assorted Colors 5pk	5,35	—

F3=EXIT F7=BACK F8=FORWARD F12=CANCEL

3. Geben Sie / in der Spalte **Bestellung** ein und drücken Sie die Eingabetaste, um einen Artikel zu bestellen. Die Anwendung zeigt Details des zu bestellenden Artikels an.

CICS-BEISPIELKATALOGANWENDUNG - Details Ihrer Bestellung

Geben Sie Bestelldetails ein und drücken Sie dann die Eingabetaste.

Artikel	Beschreibung	Preis	Bestand	In Bestellung
0010	Ball Pens Black 24pk	2,90	0011	000

Bestellmenge: 5
Benutzername: CHRISB
Abteilung: CICSDEV1

F3=EXIT F12=CANCEL

4. Wenn der Bestand ausreicht, um die Bestellung zu erfüllen, geben Sie die folgenden Informationen ein:
 - a. Füllen Sie das Feld **Bestellmenge** aus. Geben Sie die Anzahl der Artikel an, die Sie bestellen möchten.
 - b. Füllen Sie das Feld **Benutzername** aus. Geben Sie eine 1 bis 8 Zeichen lange Zeichenfolge ein. Die Basisanwendung überprüft den hier eingegebenen Wert nicht.
 - c. Füllen Sie das Feld **Abteilung** aus. Geben Sie eine 1 bis 8 Zeichen lange Zeichenfolge ein. Die Basisanwendung überprüft den hier eingegebenen Wert nicht.
5. Drücken Sie die Eingabetaste, um die Bestellung zu übergeben, und kehren Sie zum Hauptmenü zurück.
6. Drücken Sie F3, um die Anwendung zu beenden.

Web-Service-Unterstützung für die Beispielanwendung

Die Web-Service-Unterstützung erweitert die Beispielanwendung und stellt zwei Java-Versionen eines Web-Server-Front-End-Clients sowie eine Java- und eine COBOL-Version des Web-Service-Endpunkts für die Bestelldispatcherkomponente bereit.

Die beiden Versionen des Front-End-Web-Clients und die eine Version des Web-Service-Endpunkts werden als Java-Webarchivdateien (WARs) bereitgestellt, die in der Java EE 6-Webprofilumgebung ausgeführt werden, die wiederum von der neuesten Version von WebSphere Application Server oder dem aktuellen CICS TS-Liberty-JVM-Server bereitgestellt wird. Die zweite Version des Web-Service-Endpunkts wird als CICS-Service-Provider-Anwendungsprogramm (DFH0XODE) bereitgestellt.

Datei	Beschreibung
ExampleAppClientV855.war	Web-Service-Front-End-Client für den Katalogmanager.
ExampleAppWrapperClientV855.war	Web-Service-Front-End-Client für Web-Service-Wrapper.

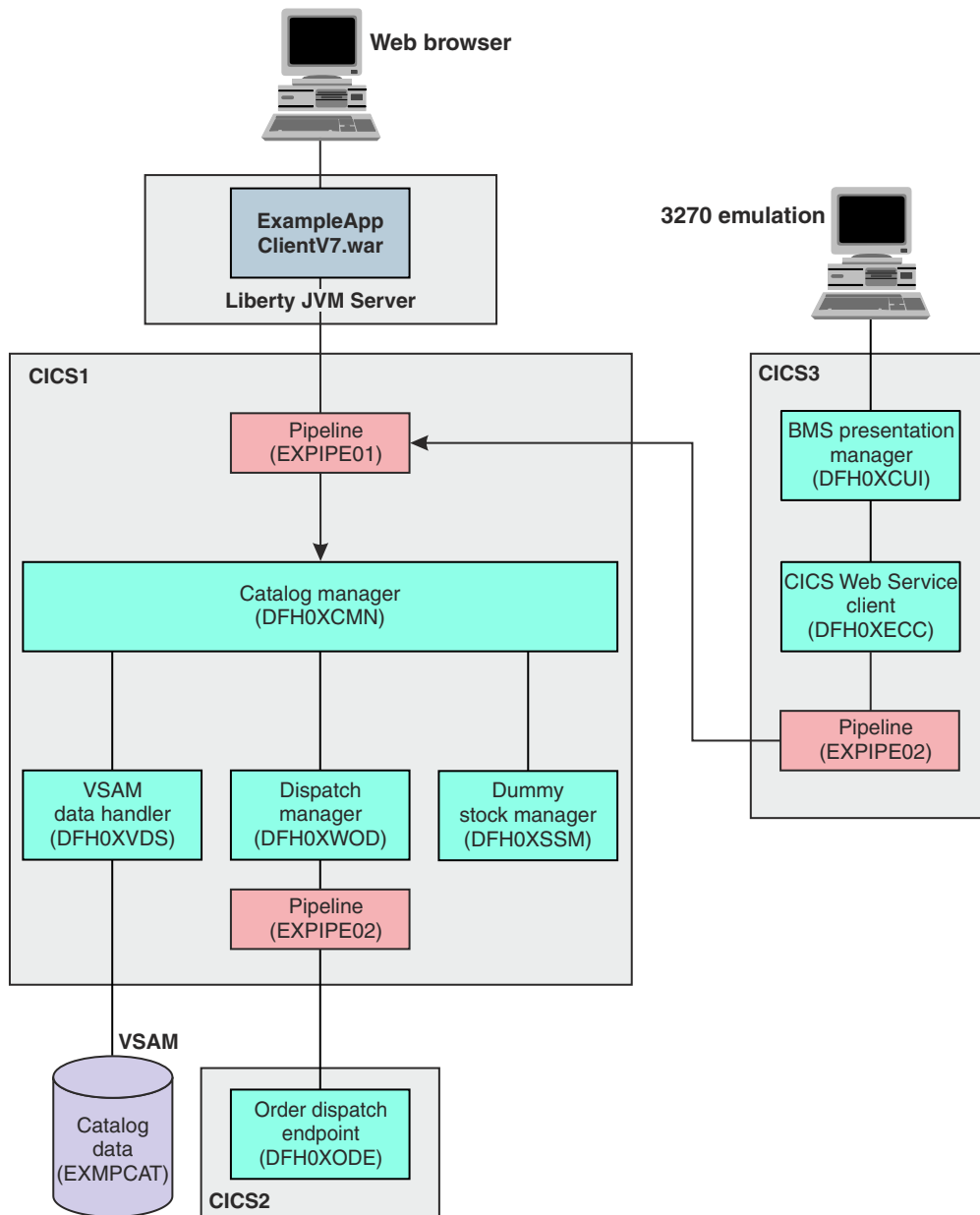
Datei	Beschreibung
ExampleAppDispatchOrderV855.war	Web-Service-Provider-Anwendung für Bestellzustellung

Diese WARs wurden aus Dynamisches Webprojekte exportiert. Informationen zur Implementierung der WAR-Dateien finden Sie unter Anwendungen auf einem JVM-Server implementieren.

Sie müssen das `jax-ws-Feature` auf dem Liberty-JVM-Server aktivieren, z. B. indem Sie die Liberty-Server-Konfigurationsdatei `server.xml` hinzufügen:

```
<feature>jaxws-2.2</feature>
```

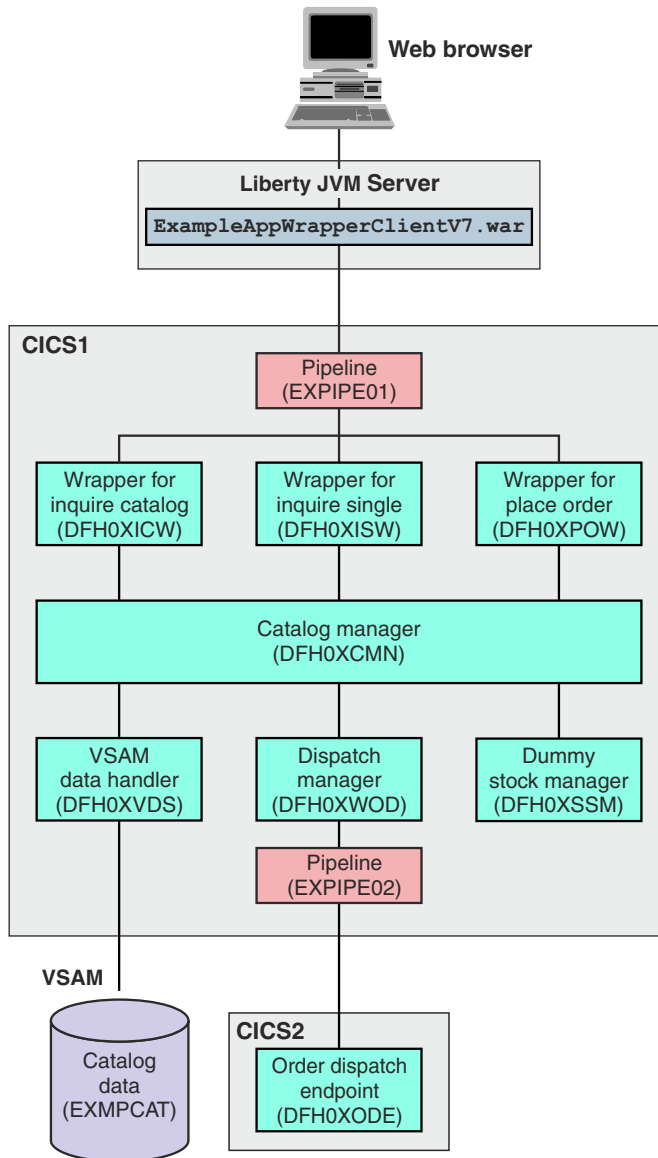
Abbildung 1 zeigt eine Konfiguration der Beispielanwendung mit einer Version des Web-Client-Front-Ends und einem CICS-Service-Provider als Web-Service-Endpunkt für Bestellzustellung. Sie umfasst außerdem einen Web-Service-Client in einem CICS-System.



In dieser Konfiguration erfolgt der Zugriff auf die Anwendung über zwei verschiedene Clients:

- Ein Web-Browser-Client, der mit dem Liberty-JVM-Server verbunden ist, auf dem `ExampleAppClientV855.War` implementiert ist.
- Der CICS-Web-Service-Client `DFH0XECC`. Dieser Client verwendet dieselbe BMS-Darstellungslogik wie die Basisanwendung, aber statt `DFH0XGUI` das Modul `DFH0XCUI`.

Abbildung 2 zeigt eine andere Version des Web-Client-Front-Ends, mit einem CICS-Service-Provider als Web-Service-Endpunkt für Bestellzustellung.



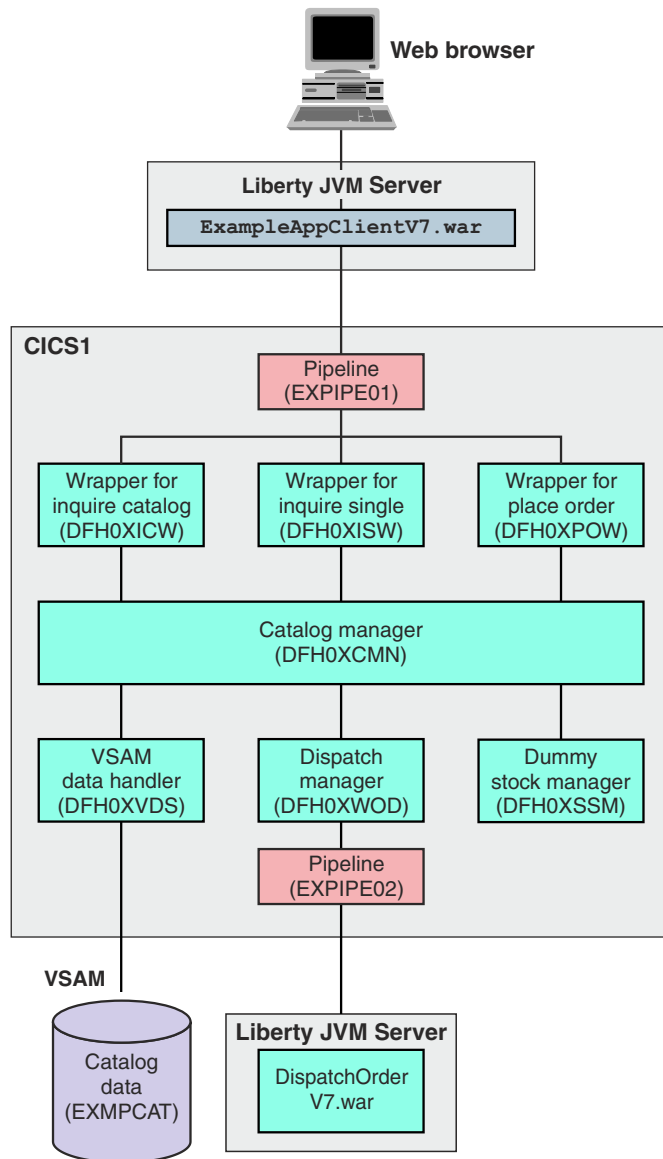
In dieser Konfiguration ist der Web-Browser-Client mit dem Liberty-JVM-Server verbunden, auf dem ExampleAppWrapperClientV855.war implementiert ist. In CICS sind drei Wrapperanwendungen (für die Funktionen zum Abrufen des Katalogs, zum Abrufen einer einzelnen Instanz und zum Erteilen eines Auftrags) als Service-Provider-Anwendungen implementiert. Diese sind wiederum mit der Basisanwendung verknüpft.

Damit der Zustellungsmanager auf Ihrem CICS-System diesen Endpunkt aufruft, müssen Sie die folgende Konfiguration mithilfe der ECFG-Konfigurationstransaktion ändern.

- 'Ausgehender Web-Service?' in YES
- 'Ausgehender Web-Service-URI' in den URI, unter dem der Endpunkt für die Bestellzustellung implementiert wird, z. B. <http://cics2:8080/exampleApp/dispatchOrder>.

Weitere Details zum Einrichten der Beispielanwendung finden Sie unter Configuring the example application.

Abbildung 3 zeigt eine Konfiguration der Beispielanwendung mit beiden Web-Client-Front-Ends und dem Web-Service-Endpunkt für Bestellzustellung auf einem Liberty-JVM-Server.



In dieser Konfiguration ist der Web-Browser-Client mit dem Liberty-JVM-Server verbunden, auf dem 'ExampleAppClientV855.war' implementiert ist. Der Web-Service-Endpunkt für Bestellzustellung 'ExampleAppDispatchOrderV855.war' ist auf dem Liberty-JVM-Server implementiert.

Damit der Zustellungsmanager auf Ihrem CICS-System diesen Endpunkt aufruft, müssen Sie die folgende Konfiguration mithilfe der ECFG-Konfigurationstransaktion ändern:

- 'Ausgehender Web-Service?' in YES
- 'Ausgehender Web-Service-URI' in den URI, unter dem der Endpunkt für die Bestellzustellung implementiert wird, z. B. <http://mylibertyserver:9080/ExampleAppDispatchOrderV855/DispatchOrder>.

Weitere Details zum Einrichten der Beispielanwendung finden Sie unter Configuring the example application.

Codepageunterstützung konfigurieren

Wie angegeben verwendet die Beispielanwendung zwei codierte Zeichensätze. Sie müssen Ihr System konfigurieren, um die Datenkonvertierung zwischen den beiden Zeichensätzen zu aktivieren.

About this task

Die codierten Zeichensätze in der Beispielanwendung sind:

037 EBCDIC-Gruppe 1: USA, Kanada (z/OS), Niederlande, Portugal, Brasilien, Australien, Neuseeland)

1208 UTF-8 Ebene 3

Procedure

Fügen Sie die folgenden Anweisungen zum Konvertierungsimagen für Ihr z/OS-System hinzu:

```
CONVERSION 037,1208;  
CONVERSION 1208,037;
```

Weitere Informationen finden Sie unter Unicode data conversion by z/OS.

Web-Service-Client- und Wrapperprogramme definieren

Wenn Sie keine automatische Installation von Programmen verwenden, müssen Sie Ressourcendefinitionen für die Web-Service-Client- und Wrapperprogramme definieren.

About this task

Optional: Benutzer, die nicht mit CICS Explorer arbeiten, können die CEDA-Transaktion verwenden, um die in den Gruppen DFH\$EXBS und DFH\$EXWS enthaltenen Ressourcen zu ändern und zu installieren.

Procedure

Verwenden Sie CICS Explorer, um PROGRAM-Ressourcendefinitionen für die Wrapperprogramme zu definieren, indem Sie **Definitionen > Programmdefinitionen** auswählen. Klicken Sie mit der rechten Maustaste in die Ansicht Programmdefinitionen und wählen Sie **Neu** aus, um eine neue Programmdefinition zu erstellen. Geben Sie eine CSD-Gruppe im Feld **CSD-Gruppe** ein und geben Sie den Programmnamen im Textfeld **Name** ein. Klicken Sie auf **Fertigstellen**, um die PROGRAM-Definition zu definieren. Erstellen Sie Definitionen für die folgenden COBOL-Programme:

Tabelle 17. SDFHSAMP-Member mit COBOL-Quellcode für die Wrappermodule

Membername	Beschreibung
DFH0XECC	Web-Service-Clientprogramm
DFH0XICW	Wrapperprogramm für den Service inquireCatalog.
DFH0XISW	Wrapperprogramm für den Service inquireSingle.
DFH0XPOW	Wrapperprogramm für den Service purchaseOrder.

Web-Service-Unterstützung installieren

Bevor Sie die Web-Service-Unterstützung für die Beispielanwendung ausführen können, müssen Sie zwei z/OS UNIX-Verzeichnisse erstellen und die erforderlichen CICS-Ressourcen erstellen.

z/OS UNIX-Verzeichnisse:

Web-Service-Support für die Beispielanwendung erfordert ein Ablageverzeichnis und ein Abholverzeichnis in z/OS UNIX.

Das Ablageverzeichnis wird verwendet, um die WEBSERVICE-Bindungsdateien zu speichern, die WEBSERVICE-Ressourcen zugeordnet sind. Jede WEBSERVICE-Ressource ist wiederum einer PIPELINE zugeordnet. Das Ablageverzeichnis wird von der PIPELINE-Ressource verwaltet, und Sie sollten den Inhalt des Verzeichnisses nicht direkt ändern. Verschiedene PIPELINES können dasselbe Ablageverzeichnis verwenden, da CICS eine eindeutige Verzeichnisstruktur des Ablageverzeichnisses für die einzelnen PIPELINES sicherstellt.

Das Abholverzeichnis enthält die WEBSERVICE-Bindungsdateien, die einer PIPELINE zugeordnet sind. Wenn eine PIPELINE installiert oder als Antwort auf einen Befehl **PERFORM PIPELINE SCAN** durchsucht wird, werden die Informationen in den Bindungsdateien verwendet, um die WEBSERVICE- und URIMAP-Definitionen, die der PIPELINE zugeordnet sind, dynamisch zu erstellen.

Die Beispielanwendung verwendet `/var/cicsts` für das Ablageverzeichnis.

Pipelinedefinition erstellen:

Die vollständige Definition einer Pipeline besteht aus einer PIPELINE-Ressource und einer PIPELINE-Konfigurationsdatei. Die Datei enthält die Details der Nachrichtenhandler, die auf Web-Service-Anforderungen und -Antworten reagieren, während diese die Pipeline durchlaufen.

About this task

Optional: Benutzer, die nicht mit CICS Explorer arbeiten, können die CEDA-Transaktion verwenden, um die in den Gruppen DFH\$EXBS und DFH\$EXWS enthaltenen Ressourcen zu ändern und zu installieren.

Die Beispielanwendung verwendet den bereitgestellten SOAP 1.1-Handler, um die SOAP-Envelopes von eingehenden und ausgehenden Anforderungen zu bearbeiten. CICS stellt Beispiele für Pipelinekonfigurationsdateien bereit, die Sie in Ihrem Service-Provider und Service-Requester verwenden können.

Mehrere Web-Services können eine einzige Pipeline gemeinsam nutzen, deshalb müssen Sie nur eine Pipeline für die eingehenden Anforderungen der Beispielanwendung definieren. Sie müssen jedoch eine zweite Pipeline für die ausgehenden Anforderungen definieren, da eine einzige Pipeline nicht so konfiguriert werden kann, dass sie gleichzeitig eine Service-Provider- und eine Service-Requester-Pipeline ist.

Wenn Sie Java-basierte Pipelines verwenden wollen, müssen Sie das Beispiel für die Service-Provider-Konfigurationsdatei, `basicsoap11javaprovider.xml`, statt der Datei `basicsoap11provider.xml` in Schritt 1b angeben. Und Sie müssen das Beispiel für die Service-Requester-Konfigurationsdatei, `basicsoap11javarequester.xml`, statt der Datei `basicsoap11requester.xml` in Schritt 2b angeben. Weitere Informationen

zu Beispielkonfigurationsdateien finden Sie unter Pipeline configuration files. Und wenn Sie den Axis2-Anwendungshandler in Ihrer Java-basierten Pipeline verwenden möchten, müssen Sie EXPIPE01 durch EXPIPE03 in Schritt 1a und EXPIPE02 durch EXPIPE04 in Schritt 2a ersetzen.

Procedure

1. Verwenden Sie CICS Explorer, um eine Pipelinedefinition für den Service-Provider zu erstellen.
 - a. Erstellen Sie mit CICS Explorer eine PIPELINE-Definition für die Wrapperprogramme, indem Sie **Definitionen > Pipelinedefinitionen** auswählen. Klicken Sie mit der rechten Maustaste in die Ansicht Pipelinedefinitionen und wählen Sie **Neu** aus, um eine neue Pipelinedefinition zu erstellen. Geben Sie DFH\$EXWS im Textfeld **Ressourcengruppe** ein und EXPIPE01 im Textfeld **Name**. Klicken Sie auf **Fertigstellen**, um die PIPELINE-Definition zu erstellen. Alternativ können Sie die PIPELINE-Definition aus der von CICS bereitgestellten Gruppe DFH\$EXWS kopieren. Klicken Sie mit der rechten Maustaste in der Ansicht Ressourcengruppendefinition auf DFH\$EXWS und wählen Sie **Neu > Pipelinedefinition** aus.
 - b. Doppelklicken Sie auf die PIPELINE-Definition und wählen Sie die Registerkarte **Attribute** im Editor Pipelinedefinition (EXPIPE01) aus. Unter **Details** muss für die **Konfigurationsdatei** die Position der Beispieldateien `/usr/lpp/cicsts/samples/pipelines/basicsoap11provider.xml` angegeben sein, wobei `/usr/lpp/cicsts` der Pfad der Dateien in Ihrem Verzeichnis ist. **Fach** muss den Wert `/var/cicsts/` haben, **Status** muss ENABLED lauten und für **WS-Verzeichnis** muss `/usr/lpp/cicsts/samples/webservices/wsbind/provider/` angegeben sein.
Bei den z/OS UNIX-Einträgen muss die Groß-/Kleinschreibung beachtet werden und sie nehmen ein CICS z/OS UNIX-Installationsstammverzeichnis `/usr/lpp/cicsts` an.
2. Verwenden Sie CICS Explorer, um eine PIPELINE-Definition für den Service-Requester zu erstellen.
 - a. Erstellen Sie mit CICS Explorer eine PIPELINE-Definition für die Wrapperprogramme, indem Sie **Definitionen > Pipelinedefinitionen** auswählen. Klicken Sie mit der rechten Maustaste in die Ansicht Pipelinedefinitionen und wählen Sie **Neu** aus, um eine neue Pipelinedefinition zu erstellen. Geben Sie DFH\$EXWS im Textfeld **Ressourcengruppe** ein und EXPIPE02 im Textfeld **Name**. Klicken Sie auf **Fertigstellen**, um die PIPELINE-Definition zu erstellen. Alternativ können Sie die PIPELINE-Definition aus der von CICS bereitgestellten Gruppe DFH\$EXWS kopieren.
 - b. Doppelklicken Sie auf die PIPELINE-Definition und wählen Sie die Registerkarte **Attribute** im Editor Pipelinedefinition (EXPIPE02) aus. Unter **Details** muss für **Konfigurationsdatei** die Position der Beispieldateien `/usr/lpp/cicsts/samples/pipelines/basicsoap11requester.xml` angegeben sein, wobei `/usr/lpp/cicsts` der Pfad der Dateien in Ihrem Verzeichnis ist. **Fach** muss den Wert `/var/cicsts/` haben, **Status** muss ENABLED lauten und für **WS-Verzeichnis** muss `/usr/lpp/cicsts/samples/webservices/wsbind/requester/` angegeben sein.

TCP/IP-Service erstellen:

Da der Client eine Verbindung mit Ihren Web-Services über einen HTTP-Transport herstellt, müssen Sie einen TCP/IP-Service definieren, um den eingehenden HTTP-Datenverkehr zu empfangen.

Procedure

Verwenden Sie CICS Explorer, um eine TCPIPSERVICE-Definition zum Verarbeiten von eingehenden HTTP-Anforderungen zu erstellen.

1. Erstellen Sie eine TCPIPSERVICE-Definition, indem Sie **Definitionen > TCP/IP-Servicedefinitionen** auswählen. Klicken Sie mit der rechten Maustaste in die Ansicht TCP/IP-Servicedefinitionen und wählen Sie **Neu** aus, um eine neue Definition zu erstellen. Geben Sie DFH\$EXWS im Textfeld **Ressourcengruppe** ein und EXMPPORT im Textfeld **Name**. Sie müssen eine Portnummer angeben. Geben Sie die Nummer eines nicht verwendeten Ports in Ihrem CICS-System ein. Klicken Sie auf **Fertigstellen**, um die TCPIPSERVICE-Definition zu erstellen.
2. Doppelklicken Sie auf die TCPIPSERVICE-Definition. Legen Sie auf der Registerkarte **Attribute** im Editor TCP/IP-Servicedefinition (EXMPPORT) die folgenden Attribute fest:
 - Urm** muss DFHWBAAX sein.
 - Protokoll** muss HTTP sein.
 - Transaktion** muss CWXN sein.
3. Verwenden Sie die Standardwerte für alle anderen Attribute.

Optional: Benutzer, die nicht mit CICS Explorer arbeiten, können die CEDA-Transaktion verwenden, um die in den Gruppen DFH\$EXBS und DFH\$EXWS enthaltenen Ressourcen zu ändern und zu installieren.

WEBSERVICE- und URIMAP-Ressourcen dynamisch installieren:

Jede Funktion, die als Web-Service zugänglich gemacht wird, erfordert eine WEBSERVICE-Ressource, die die eingehende XML des SOAP-Hauptteils mit der Schnittstelle des Kommunikationsbereichs des Programms abgleicht, und eine URIMAP-Ressource, die eingehende Anforderungen an die richtige Pipeline und den richtigen Web-Service weiterleitet. Obwohl Sie die Onlineresourcendefinition nutzen können, um Ihre WEBSERVICE- und URIMAP-Ressourcen zu definieren und zu installieren, können Sie diese auch von CICS dynamisch erstellen lassen, wenn Sie eine Pipelineressource installieren.

About this task

Optional: Benutzer, die nicht mit CICS Explorer arbeiten, können die CEDA-Transaktion verwenden, um die in den Gruppen DFH\$EXBS und DFH\$EXWS enthaltenen Ressourcen zu ändern und zu installieren.

Procedure

1. Verwenden Sie CICS Explorer, um die PIPELINE-Ressourcen zu installieren.
 - a. Wählen Sie **Definitionen > Pipelinedefinitionen** aus. Klicken Sie mit der rechten Maustaste auf die PIPELINE-Definition EXPIPE01 in der Ansicht Pipelinedefinitionen und wählen Sie **Installieren** aus. Wählen Sie Ihre CICS-Zielregion aus, indem Sie das Kontrollkästchen aktivieren. Klicken Sie auf **OK**, um die PIPELINE zu installieren.

Anmerkung: Wenn Sie unter „Pipelinedefinition erstellen“ auf Seite 712 Java-basierte Pipelinedefinitionen erstellt haben, klicken Sie in der Ansicht Pipelinedefinitionen auf die PIPELINE-Definition EXPIPE03.
 - b. Wiederholen Sie dies für die PIPELINE-Definition EXPIPE02 bzw. EXPIPE04 für Java-basierte Pipelines.

Beim Installieren der einzelnen PIPELINE-Ressourcen durchsucht CICS das im WSDIR-Attribut der PIPELINE angegebene Verzeichnis (das Abholverzeichnis). Für jede WEBSERVICE-Bindungsdatei in dem Verzeichnis, d. h. für jede Datei mit dem Suffix .wsbind, installiert CICS eine WEBSERVICE-Ressource und eine URIMAP-Ressource, falls diese Ressourcen nicht vorhanden sind.

Die URIMAP-Ressource stellt CICS die Informationen für die Zuordnung der WEBSERVICE-Ressource zu einem bestimmten URI bereit. Vorhandene Ressourcen werden ersetzt, wenn die Informationen in der Bindungsdatei neuer sind als die vorhandenen Ressourcen.

Eine zweite optionale URIMAP-Ressource wird installiert, wenn eine WSDL-Datei oder eine WSDL-Archivierungsdatei in das Pickup-Verzeichnis kopiert wurde. Diese URIMAP-Ressource stellt CICS die Informationen für die Zuordnung der WSDL-Archivdatei oder des WSDL-Dokuments zu einem bestimmten URI bereit, sodass die externen Requester mit dem URI die WSDL-Archivdatei oder das WSDL-Dokument erkennen können.

Wenn die PIPELINE später inaktiviert und gelöscht wird, werden alle zugeordneten WEBSERVICE- und URIMAP-Ressourcen ebenfalls gelöscht.

2. Wenn Sie die PIPELINE-Ressource bereits installiert haben, verwenden Sie den Befehl **PERFORM PIPELINE SCAN**, um den Scan des PIPELINE-Abholverzeichnisses zu initialisieren.

Wenn Sie die PIPELINE-Ressourcen installiert haben, werden die folgenden WEBSERVICE-Ressourcen und die zugeordneten URIMAP-Ressourcen für die Provider-Pipeline in Ihrem System installiert:

- dispatchOrder
- dispatchOrderEndpoint
- inquireCatalog
- inquireCatalogClient
- inquireCatalogWrapper
- inquireSingle
- inquireSingleClient
- inquireSingleWrapper
- placeOrder
- placeOrderClient
- placeOrderWrapper

Die Namen der WEBSERVICE-Ressourcen leiten sich ab aus den Namen der WEBSERVICE-Bindungsdateien. Die Namen der URIMAP-Ressourcen werden dynamisch generiert. Für jedes WSDL-Dokument, das im Abholverzeichnis der Pipeline enthalten ist, wird eine zusätzliche URIMAP generiert. Sie können die Ressourcen anzeigen, indem Sie **Operationen > Web-Services** auswählen, um die Ansicht 'Web-Services' zu öffnen. Klicken Sie mit der rechten Maustaste auf die WEBSERVICE-Ressource und wählen Sie **Zugehörige öffnen > URI-Maske** aus.

Die CICS Explorer-Ansicht enthält die Namen der PIPELINE-Ressource, der URIMAP-Ressource und des Zielprogramms, das den einzelnen Web-Services zugeordnet ist. In diesem Beispiel gibt es keine URIMAP bzw. kein Zielprogramm für 'WEBSERVICE(dispatchOrder)', weil die WEBSERVICE-Ressource zu einer ausgehenden Anforderung gehört.

WEBSERVICE(dispatchOrderEndpoint) stellt die lokale CICS-Implementierung des Bestellzustellungsservice dar.

WEBSERVICE-Ressourcen mit Onlineresourcendefinition erstellen:

Alternativ zur Verwendung des Pipelinescanmechanismus zur Installation von WEBSERVICE-Ressourcen können Sie diese mithilfe der Onlineresourcendefinition erstellen und installieren.

Before you begin

Important: Wenn Sie die Onlineresourcendefinition verwenden, um die WEBSERVICE- und URIMAP-Ressourcen zu definieren, müssen Sie sicherstellen, dass ihre Web-Service-Bindungsdateien **nicht** im Pickup-Verzeichnis der PIPELINE vorhanden sind. Auf diese Weise wird sichergestellt, dass die WEBSERVICE- und URIMAP-Ressourcen während eines Pipelinescans des Abholverzeichnis nicht dynamisch installiert werden. Alternativ können Sie sicherstellen, dass kein Wert für WSDIR in der PIPELINE angegeben ist. Wenn Sie jedoch keinen Wert für WSDIR angeben, werden keine Pipelinescans des Abholverzeichnis ausgeführt. Deshalb müssen alle WEBSERVICE- und URIMAP-Ressourcen mithilfe der Onlineresourcendefinition erstellt und installiert werden.

Procedure

1. Verwenden Sie CICS Explorer, um eine WEBSERVICE-Definition für die INQUIRE CATALOG-Definition der Beispielanwendung zu erstellen.
 - a. Erstellen Sie mit CICS Explorer eine WEBSERVICE-Definition, indem Sie **Definitionen > Web-Service-Definition** auswählen.
 - b. Klicken Sie mit der rechten Maustaste in die Ansicht Web-Service-Definitionen und wählen Sie **Neu** aus, um eine neue WEBSERVICE-Definition zu erstellen.
 - c. Geben Sie DFH\$EXWS im Textfeld **Ressourcengruppe** ein, geben Sie EXINQCWS im Textfeld **Name** ein und geben Sie EXPIPE01 im Feld **Pipeline** ein oder geben Sie EXPIPE03 für Java-basierte Pipelines ein. Sie müssen das WSBIND-Attribut eingeben, bevor Sie die WEBSERVICE-Definition erstellen können. Geben Sie im Textfeld **WSBind-Datei** die Zeichenfolge /usr/lpp/cicsts/samples/webservices/wsbinding/provider/inquireCatalog.wsbinding ein.
 - d. Klicken Sie auf **Fertigstellen**, um die WEBSERVICE-Definition zu erstellen.
2. Wiederholen Sie den vorherigen Schritt für alle folgenden Funktionen der Beispielanwendung.

Funktion	WEBSERVICE-Name	PIPELINE-Attribut	WSBind-Attribut
INQUIRE SINGLE ITEM	EXINQSWWS	EXPIPE01 oder EXPIPE03	/usr/lpp/cicsts/samples/webservices/wsbinding/provider/inquireSingle.wsbinding
PLACE ORDER	EXORDRWS	EXPIPE01 oder EXPIPE03	/usr/lpp/cicsts/samples/webservices/wsbinding/provider/placeOrder.wsbinding
DISPATCH STOCK	EXODRQWS	EXPIPE02 oder EXPIPE04	/usr/lpp/cicsts/samples/webservices/wsbinding/requester/dispatchOrder.wsbinding
DISPATCH STOCK-Endpunkt (optional)	EXODEPWS	EXPIPE01 oder EXPIPE03	/usr/lpp/cicsts/samples/webservices/wsbinding/provider/dispatchOrderEndpoint.wsbinding

URIMAP-Ressourcen mit Onlineresourcendefinition erstellen:

Alternativ zur Verwendung des Pipelinescanmechanismus Betriebsmittel zur Installation von URIMAP-Ressourcen können Sie diese mithilfe der Onlineresourcendefinition erstellen und installieren.

Before you begin

Important: Wenn Sie die Onlineresourcendefinition verwenden, um die WEBSERVICE- und URIMAP-Ressourcen zu definieren, müssen Sie sicherstellen, dass ihre Web-Service-Bindungsdateien **nicht** im Pickup-Verzeichnis der PIPELINE vorhanden sind. Auf diese Weise wird sichergestellt, dass die WEBSERVICE- und URIMAP-Ressourcen während eines Pipelinescans des Abholverzeichnis nicht dynamisch installiert werden. Alternativ können Sie sicherstellen, dass kein Wert für WSDIR in der PIPELINE angegeben ist. Wenn Sie jedoch keinen Wert für WSDIR angeben, werden keine Pipelinescans des Abholverzeichnis ausgeführt. Deshalb müssen alle WEBSERVICE- und URIMAP-Ressourcen mithilfe der Onlineresourcendefinition erstellt und installiert werden.

Procedure

1. Verwenden Sie CICS Explorer, um eine URIMAP-Definition für die INQUIRE CATALOG-Definition der Beispielanwendung zu erstellen.
 - a. Erstellen Sie in CICS Explorer eine URIMAP-Definition, indem Sie **Definitionen > URIMAP-Definitionen** auswählen.
 - b. Klicken Sie mit der rechten Maustaste in die Ansicht URIMAP-Definitionen und wählen Sie **Neu** aus, um eine neue URIMAP-Definition zu erstellen.
 - c. Geben Sie INQCURI im Textfeld **Name** ein und geben Sie * im Textfeld **Host** ein. Sie müssen das **Pfadattribut** eingeben, bevor Sie die URIMAP-Definition erstellen können. Geben Sie im Textfeld **Pfad** die Zeichenfolge /exampleApp/inquireCatalog ein. **Verwendung** muss auf **Pipeline** gesetzt sein. Die PIPELINE-Ressource ist EXPIPE01 oder EXPIPE03 für Java-basierte Pipelines.
 - d. Klicken Sie auf **Fertigstellen**, um die URIMAP-Definition fertigzustellen.
 - e. Doppelklicken Sie auf die neue URIMAP-Ressource, um den Editor zu öffnen. Legen Sie auf der Registerkarte **Attribute** im Editor das **Web-Service-Attribut** auf EXINQWS fest und legen Sie **TCP/IP-Service** auf SOAPPORT fest.
2. Wiederholen Sie den vorherigen Schritt für alle verbleibenden Funktionen der Beispielanwendung. Verwenden Sie die folgenden Namen für Ihre URIMAPs:

Funktion	URIMAP-Name
INQUIRE SINGLE ITEM	INQSURI
PLACE ORDER	ORDRURI
DISPATCH STOCK	Nicht erforderlich
DISPATCH STOCK-Endpunkt (optional)	ODEPURI

3. Geben Sie die folgenden eindeutigen Attribute für jede URIMAP an:

Funktion	URIMAP-Name	PATH	WEBSERVICE
INQUIRE SINGLE ITEM	INQSURI	/exampleApp/inquireSingle	EXINQWS
PLACE ORDER	ORDRURI	/exampleApp/placeOrder	EXORDRWS

Funktion	URIMAP-Name	PATH	WEBSERVICE
DISPATCH STOCK-End- punkt (optional)	ODEPURI	/exampleApp/dispatchOrder	EXODEPWS

Installation abschließen:

Um die Installation abzuschließen, installieren Sie die Onlineressourcendefinitionsgruppe, die Ihre Ressourcendefinitionen enthält.

Procedure

Klicken Sie im Fenster Ressourcengruppendefinitionen mit der rechten Maustaste auf die Ressourcengruppe. Wählen Sie **Installieren** aus. Stellen Sie sicher, dass Ihr CICSplex korrekt und Ihre Zielregion ausgewählt ist, und klicken Sie anschließend auf **OK**.

Results

Ihre Onlineressourcendefinition ist jetzt installiert und die Anwendung kann verwendet werden.

Optional: Benutzer, die nicht mit CICS Explorer arbeiten, können die CEDA-Transaktion verwenden, um die in den Gruppen DFH\$EXBS und DFH\$EXWS enthaltenen Ressourcen zu ändern und zu installieren.

Web-Client konfigurieren

Bevor Sie den Web-Client verwenden können, müssen Sie das Java-Webarchiv (WAR) für den Client in einer der unterstützten Umgebungen implementieren und es für den Aufruf der entsprechenden Endpunkte in Ihrem CICS-System konfigurieren.

About this task

Die folgenden Umgebungen werden für die beiden Versionen der Front-End-Anwendung des Web-Clients, ExampleAppClientV855.war und ExampleAppWrapperClientV855.war, unterstützt:

- CICS Liberty-JVM-Server mit dem aktuellen WebSphere Liberty-Profil

Die WAR-Dateien befinden Sie im Verzeichnis *hlq /samples/webservices/client* in z/OS UNIX.

Procedure

1. Um den Web-Client zu starten, geben Sie die folgende URL in Ihren Webbrowser ein, wobei *mylibertyserver* der Hostname des Liberty-JVM-Servers ist, auf dem der Web-Client installiert ist.
 - Verwenden Sie für ExampleAppClientV855.war die URL `http://mylibertyserver:9080/ExampleAppClientV855/`.
 - Verwenden Sie für ExampleAppWrapperClientV855.war die URL `http://mylibertyserver:9080/ExampleAppWrapperClientV855/`.
2. Klicken Sie auf **KONFIGURIEREN**, um die Konfigurationsseite anzuzeigen. Die Konfigurationsseite wird angezeigt.

3. Geben Sie den neuen Endpunkt für den Web-Service INQUIRE CATALOG, INQUIRE ITEM und PLACE ORDER ein.
 - a. Ersetzen Sie in den URLs die Zeichenfolge *myCicsServer* durch den Namen des Systems, auf dem Ihre CICS-Instanz ausgeführt wird.
 - b. Ersetzen Sie die Portnummer 8080 durch die in der TCIPSERVICE-Definitionsressource konfigurierte Portnummer.
4. Klicken Sie auf **ÜBERGEBEN**.

Results

Die Webanwendung kann jetzt ausgeführt werden.

What to do next

Die URL für den Web-Service-Aufruf wird in einer HTTP-Sitzung gespeichert. Deshalb muss dieser Konfigurationsschritt jedesmal wiederholt werden, wenn ein Web-Browser das erste Mal eine Verbindung zum Client herstellt.

Web-Service-aktivierte Anwendung ausführen

Sie können die Beispielanwendung über einen Web-Browser aufrufen.

About this task

Stellen Sie sicher, dass Sie den Web-Client konfiguriert haben, bevor Sie fortfahren. Siehe *Configuring the example application*.

Procedure

1. Geben Sie die folgende URL in Ihren Web-Browser ein: `http://mylibertyserver:9080/ExampleAppClientV855/`, wobei *mylibertyserver* der Hostname des Servers ist, auf dem der Web-Service-Client installiert wird.
2. Klicken Sie auf die Schaltfläche **ABFRAGEN**.
3. Geben Sie eine Artikelnummer ein und klicken Sie auf die Schaltfläche **ÜBERGEBEN**.

Tip: Die Basisanwendung ist mit Elementnummern in der Reihenfolge 0010, 0020,... bis 0210 vorbereitet.

Die Anwendung zeigt eine Liste von Artikeln in dem Katalog an, beginnend mit der von Ihnen eingegebenen Artikelnummer.

4. Wählen Sie den Artikel aus, den Sie bestellen möchten.
 - a. Aktivieren Sie in der Spalte **Auswählen** das Optionsfeld für den Artikel, den Sie bestellen möchten.
 - b. Klicken Sie auf die Schaltfläche **ÜBERGEBEN**.
5. Geben Sie die folgenden Informationen ein, um eine Bestellung aufzugeben.
 - a. Füllen Sie das Feld **Menge** aus. Geben Sie die Anzahl der Artikel an, die Sie bestellen möchten.
 - b. Füllen Sie das Feld **Benutzername** aus. Geben Sie eine 1 bis 8 Zeichen lange Zeichenfolge ein. Die Basisanwendung überprüft den hier eingegebenen Wert nicht.
 - c. Füllen Sie das Feld **Abteilungsname** aus. Geben Sie eine 1 bis 8 Zeichen lange Zeichenfolge ein. Die Basisanwendung überprüft den hier eingegebenen Wert nicht.
 - d. Klicken Sie auf die Schaltfläche **ÜBERGEBEN**.

Die Anwendung bestätigt, dass die Bestellung platziert wurde:

Beispielanwendung implementieren

Mit dem Web-Service-Assistenten können Sie Teile der Beispielanwendung als Web-Service implementieren. Obwohl die Anwendung auch funktioniert, ohne dass Sie diese Task auszuführen, müssen Sie eine ähnliche Task abschließen, wenn Sie Ihre eigenen Anwendungen implementieren wollen, um die Beispielanwendung zu erweitern.

Programmschnittstelle extrahieren

Um ein Programm mit dem CICS-Web-Service-Assistenten zu implementieren, müssen Sie ein Copybook erstellen, das mit dem Kommunikationsbereich oder der Containerschnittstelle übereinstimmt.

About this task

In diesem Beispiel wird die Funktion INQUIRE SINGLE ITEM des zentralen Katalogmanagerprogramms (DFH0XCMN) als Web-Service implementiert. Die Schnittstelle mit diesem Programm ist ein Kommunikationsbereich; die Struktur des Kommunikationsbereichs ist im Copybook DFH0XCP1 definiert:

```
*      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).
*      Fields used in 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).
```

Das Copybook definiert drei separate Schnittstellen für die Funktionen INQUIRE CATALOG, INQUIRE SINGLE ITEM und PLACE ORDER, die in dem Copybook

einander überlagern. Allerdings bietet das Dienstprogramm DFHLS2WS keine Unterstützung für die Anweisung REDEFINES. Deshalb müssen Sie aus dem kombinierten Copybook nur solche Abschnitte extrahieren, die sich auf die INQUIRE SINGLE-Funktion beziehen:

```
* 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).
```

Das neu definierte Element CA-REQUEST-SPECIFIC wurde entfernt und durch den Abschnitt des Copybooks ersetzt, das es für die INQUIRE SINGLE-Funktion neu definiert hat. Das Copybook kann jetzt mit dem Web-Service-Assistenten verwendet werden.

Das Copybook wird mit der Beispielanwendung als Copybook DFH0XCP4 geliefert.

Web-Service-Assistentenprogramm DFHLS2WS ausführen

Der Assistent für CICS-Web-Services umfasst zwei Stapeldienstprogramme, die Sie dabei unterstützen, vorhandene CICS-Anwendungen in Web-Services umzusetzen und CICS-Anwendungen für die Verwendung von Web-Services zu aktivieren, die von externen Providern bereitgestellt werden. Das Programm DFHLS2WS setzt eine Sprachstruktur um, um eine Web-Service-Bindungsdatei und eine Web-Service-Beschreibung zu generieren.

Procedure

1. Kopieren Sie die bereitgestellte Beispiel-JCL in eine geeignete Arbeitsdatei. Die JCL wird `samples/webservices/JCL/LS2WS` bereitgestellt.
2. Fügen Sie eine gültige JOB-Karte zur JCL hinzu.
3. Codieren Sie die Parameter für DFHLS2WS. Die folgenden Parameter sind für die Funktion INQUIRE SINGLE ITEM der Beispielanwendung erforderlich:

```
//INPUT.SYSUT1 DD *
LOGFILE=/u/exampleapp/wsbind/inquireSingle.log
PDSLIB=CICSHLQ.SDFHSAMP
REQMEM=DFH0XCP4
RESPMEM=DFH0XCP4
LANG=COBOL
PGMNAME=DFH0XCMN
PGMINT=COMMAREA
URI=mein_cics-server:mein_port/exampleApp/inquireSingle
WSBIND=/u/exampleapp/wsbind/inquireSingle.wsbind
WSDL=/u/exampleapp/wsd1/inquireSingle.wsd1
*/
```

LOGFILE=/u/exampleapp/wsbind/inquireSingle.log

Die Datei, die zum Aufzeichnen von Diagnoseinformationen von

DFHLS2WS verwendet wird. Die Datei wird üblicherweise nur von einer IBM Software Support-Organisation eingesetzt.

PDSLIB=CICSHLQ.SDFHSAMP

Der Name der partitionierten Datei, in der der Web-Service-Assistent nach Copybooks sucht, die die Anforderungs- und Antwortstrukturen definieren. In dem Beispiel ist dies die von CICS installierte Datei SDFHSAMP.

REQMEM=DFH0XCP4

RESPMEM=DFH0XCP4

Diese Parameter definieren die Sprachstruktur für die Anforderung und die Antwort an das Programm. In dem Beispiel haben die Anforderung und die Antwort dieselbe Struktur und sind durch dasselbe Copybook definiert.

LANG=COBOL

Das Zielprogramm und die Datenstrukturen sind in COBOL geschrieben.

PGMNAME=DFH0XCMN

Der Name des Zielprogramms, das gestartet wird, wenn eine Web-Service-Anforderung empfangen wird.

PGMINT=COMMAREA

Das Zielprogramm wird über eine Kommunikationsbereichsschnittstelle aufgerufen.

URI=*mein_cics-server:mein_port/exampleApp/inquireSingle*

Der eindeutige Teil des URI, der in der generierten Web-Service-Definition verwendet wird, und zum Erstellen der URIMAP-Ressource dient, die eingehende Anforderungen dem richtigen Web-Service zuordnet. Der angegebene Wert führt dazu, dass der Service externen Clients zur Verfügung steht:

`http://mein_cics-server:mein_port/exampleApp/inquireSingle`

Dabei sind *mein_cics-server* und *mein_port* die CICS-Serveradresse und der Port, auf dem diese WEBSERVICE-Ressource installiert wurde.

Anmerkung: Der Parameter hat keinen führenden '/'.

WSBIND=/u/exampleapp/wsbind/inquireSingle.wsbind

Die Position unter z/OS UNIX, in die die Web-Service-Bindungsdatei geschrieben wird.

Anmerkung: Wenn die Datei mit dem Pipelinescanmechanismus verwendet wird, muss sie die Erweiterung *.wsbind* haben.

WSDL=/u/exampleapp/wsd1/inquireSingle.wsd1

Die Position unter z/OS UNIX, in die die Datei mit der generierten Web-Service-Beschreibung geschrieben wird. Es ist sinnvoll, übereinstimmende Namen für die Web-Service-Bindungsdatei und die entsprechende Web-Service-Beschreibung zu verwenden.

Herkömmlicherweise haben Dateien, die Web-Service-Beschreibungen enthalten, die Erweiterung *.wsdl*.

Die Web-Service-Beschreibung liefert die Informationen, die ein Client verwenden muss, um auf den Web-Service zuzugreifen. Sie enthält eine XML-Schemadefinition der Anforderung und der Antwort sowie Positionsinformationen für den Service.

4. Führen Sie den Job aus. Eine Web-Service-Beschreibung und eine Web-Service-Bindungsdatei werden an den angegebenen Positionen erstellt.

Beispiel für generiertes WSDL-Dokument

Ein Beispiel für das WSDL-Dokument (WSDL - Web Service Description Language, Sprache für Web-Service-Beschreibung), das generiert wird, wenn das Web-Service-Assistentenprogramm DFHLS2WS ausgeführt wird.

```
<?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.wsdl.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>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </types>
  <xsd:element name="DFH0XCMNOperation" nillable="false" type="tns:ProgramInterface"/>
</definitions>
```

.... 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">
  <!-- Dieses soap:binding-Element weist auf die Verwendung von SOAP 1.1 hin -->
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <!-- Dieses soap:binding-Element weist auf die Verwendung von SOAP 1.2 hin -->
  <!-- <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">
```

```

<!-- Dieses soap:address-Element gibt die Position des Web-Service über HTTP an.
      Ersetzen Sie "my-server" durch den TCP/IP-Hostnamen Ihrer CICS-Region.
      Ersetzen Sie "my-port" durch die Portnummer Ihres CICS-TCP/IP-Service. -->
<soap:address location="http://my-server:my-port/exampleApp/inquireSingles.log"/>
<!-- Dieses soap:address-Element gibt die Position des Web-Service über HTTPS an. -->
<!-- <soap:address location="https://my-server:my-port/exampleApp/inquireSingles.log"/> -->
<!-- Dieses soap:address-Element gibt die Position des Web-Service über Websphere MQSeries an.
      Ersetzen Sie "my-queue" durch den entsprechenden Warteschlangennamen. -->
<!-- <soap:address location="jms:/queue?destination=my-queue&connectionFactory=()&
      targetService=/exampleApp/inquireSingles.log&initialContextFactory=com.ibm.mq.jms.Nojndi" /> -->
</port>
</service>
</definitions>

```

Web-Service-Bindungsdatei implementieren

Die WEBSERVICE-Bindungsdatei, die von DFHLS2WS erstellt wurde, wird während der Installation einer PIPELINE-Ressource dynamisch in Ihrer CICS-Region implementiert.

About this task

Wenn ein Pipelinescanbefehl abgesetzt wird, durchsucht CICS das Abholverzeichnis nach WEBSERVICE-Bindungsdateien mit der Erweiterung .wsbind. Für jede gefundene Bindungsdatei bestimmt CICS, ob eine WEBSERVICE-Ressource installiert werden soll.

Eine URIMAP-Ressource wird ebenfalls erstellt, um den URI, wie in der JCL angegeben, der installierten WEBSERVICE-Ressource und der PIPELINE zuzuordnen, in der der Web-Service installiert wird. Wenn eine gescannte WEBSERVICE-Ressource gelöscht wird, wird die ihr zugeordnete URIMAP-Ressource ebenfalls gelöscht.

Procedure

1. Ändern Sie die PIPELINE-Definition für Ihre Provider-Pipeline PIPELINE(EX-PIPE01) in CICS Explorer, indem Sie **Definitionen > Pipelinedefinitionen** auswählen. Doppelklicken Sie auf EXPIPE01, um den Editor Pipelinedefinition (EXPIPE01) zu öffnen. Ändern Sie auf der Registerkarte **Attribute** den Parameter **WS Directory** in /u/exampleapp/wsbind. Das Abholverzeichnis enthält die WEBSERVICE-Bindungsdatei, die Sie mit DFHLS2WS generiert haben.
2. Kopieren Sie alle anderen WEBSERVICE-Bindungsdateien, die von der Anwendung verwendet werden, in dasselbe Verzeichnis. In diesem Beispiel werden die folgenden Dateien kopiert:

```

inquireCatalog
placeOrder

```

Sie werden im Verzeichnis /usr/lpp/cicsts/samples/webservices/wsbind/provider bereitgestellt.

3. Installieren Sie die PIPELINE-Ressource.

Results

CICS erstellt zwei URIMAP-Ressourcen: die erste URIMAP-Definition ist in einem Service-Provider erforderlich, wenn sie Informationen enthält, die den URI einer eingehenden Web-Service-Anforderung den anderen Ressourcen (z. B. der PIPELINE-Ressource) zuordnet, die der Service anfordert. Die zweite URIMAP enthält Informationen, die den URI einer eingehenden Anforderung nach dem WSDL-Dokument oder nach Dokumenten für den Web-Service zuordnen.

Komponenten der Basisanwendung

In den folgenden Tabellen finden Sie die Komponenten der Basisanwendung und die im SDFHSAMP-Beispiel bereitgestellten Member. Die aufgelisteten SDFH-SAMP-Member enthalten BMS-Masken, die COBOL-Quelle und Copybooks für die Basisanwendung, die Web-Service-Clientanwendung und die Wrappermodule.

Tabelle 18. SDFHSAMP-Member mit BMS-Masken

Membername	Beschreibung
DFH0XS1	BMS-Makros für die Maskengruppe, die aus der Maske (EXMENU) für die Anzeige Hauptmenü und der Maske (EXORDR) für die Anzeige Details Ihrer Bestellung besteht.
DFH0XS2	BMS-Makros für die Maskengruppe, die aus der Maske (EXINQC) für die Anzeige Katalog abfragen besteht.
DFH0XS3	BMS-Makros für die Maskengruppe, die aus der Maske (EXCONF) für die Anzeige CICS-Beispielkataloganwendung konfigurieren besteht.
DFH0XM1	Durch Assemblieren von DFH0XS1 generiertes COBOL-Copybook. DFH0XGUI und DFH0XCUI enthalten dieses Copybook.
DFH0XM2U	COBOL-Copybook, generiert durch Assemblieren von DFH0XS2 und durch Bearbeiten des Ergebnisses auf eine Weise, dass eine indexierte Array-Struktur eingeschlossen wird, die die Programmierung des Copybooks vereinfacht. DFH0XGUI und DFH0XCUI enthalten dieses Copybook.
DFH0XM3	Durch Assemblieren von DFH0XS3 generiertes COBOL-Copybook. DFH0XCFG enthält dieses Copybook.

Tabelle 19. SDFHSAMP-Member mit COBOL-Quelle für die Basisanwendung

Membername	Beschreibung
DFH0XCFG	Durch Transaktions-ECFG aufgerufenes Programm zum Lesen und Aktualisieren der VSAM-Konfigurationsdatei.
DFH0XCMN	Controllerprogramm für die Kataloganwendung. Alle Anforderungen durchlaufen das Controllerprogramm.
DFH0XGUI	Durch Transaktion-EGUI aufgerufenes Programm zum Verwalten des Sendens der BMS-Masken an den Terminalbenutzer und des Empfangens der Masken vom Terminalbenutzer. Dieses Programm ist mit dem Programm DFH0XCMN verknüpft.
DFH0XODE	Eine von zwei Versionen des Endpunkts für den Web-Service für die Bestellzustellung. Dies ist die Version, die in CICS ausgeführt wird. Dieses Programm legt den Text "Order in dispatch" (Bestellung in Zustellung) im Rückgabekommunikationsbereich fest.
DFH0XSDS	Eine <i>Testversion</i> des Datenspeicherprogramms, mit der die Anwendung auch dann ausgeführt werden kann, wenn die VSAM-Katalogdatei nicht eingerichtet wurde. DFH0XSDS verwendet im Programm definierte Daten statt in einer VSAM-Datei gespeicherte Daten.
DFH0XSOD	Eine Testversion des Programms zur Bestellzustellung. Es legt den Rückgabecode im Kommunikationsbereich auf '0' fest und kehrt zum Aufrufer zurück. DFH0XSOD wird verwendet, wenn keine ausgehenden Web-Services erforderlich sind.

Tabelle 19. SDFHSAMP-Member mit COBOL-Quelle für die Basisanwendung (Forts.)

Membername	Beschreibung
DFH0XSSM	Eine Testversion des Bestandsmanagerprogramms (Nachbestellungsprogramms). DFH0XSSM legt den Rückgabecode im Kommunikationsbereich auf '0' fest und kehrt zum Aufrufer zurück.
DFH0XVDS	Die VSAM-Version des Datenspeicherprogramms. DFH0XVDS greift auf die VSAM-Datei zu, um Lesevorgänge oder Aktualisierungen des Katalogs auszuführen.
DFH0XWOD	Die Web-Service-Version des Programms zur Bestellzustellung. DFH0XWOD gibt den Befehl EXEC CICS INVOKE WEBSERVICE aus, um einen ausgehenden Web-Service-Aufruf eines Bestelldispatchers zu starten.

Tabelle 20. SDFHSAMP-Member mit COBOL-Copybooks für die Basisanwendung

Membername	Beschreibung
DFH0XCP1	Definiert eine Kommunikationsbereichsstruktur, die die Anforderung und Antwort für die Funktionen zum Abrufen des Katalogs, zum Abrufen einer einzelnen Instanz und zum Erteilen eines Auftrags einschließt. Die Programme DFH0XCMN, DFH0XCUI, DFH0XECC, DFH0XGUI, DFH0XICW, DFH0XISW, DFH0XPOW, DFH0XSDS und DFH0XVDS schließen dieses Copybook ein.
DFH0XCP2	Definiert eine Kommunikationsbereichsstruktur für die Bestelldispatcher- und Bestandsmanagermodule. Die Programme DFH0XCMN, DFH0XSOD, DFH0XSSM und DFH0XWOD schließen dieses Copybook ein.
DFH0XCP3	Definiert eine Datenstruktur für die Anforderung und Antwort der Funktion zum Abrufen des Katalogs. Wird als Eingabe für DFHLS2WS verwendet, um inquireCatalog.wsdl und inquireCatalog.wsbind zu erzeugen.
DFH0XCP4	Definiert eine Datenstruktur für die Anforderung und Antwort der Funktion zum Abrufen einer einzelnen Instanz. Wird als Eingabe für DFHLS2WS verwendet, um inquireSingle.wsdl und inquireSingle.wsbind zu erzeugen.
DFH0XCP5	Definiert eine Datenstruktur für die Anforderung und Antwort der Funktion zum Erteilen eines Auftrags. Wird als Eingabe für DFHLS2WS verwendet, um placeOrder.wsdl und placeOrder.wsbind zu erzeugen.
DFH0XCP6	Definiert eine Datenstruktur für die Anforderung und Antwort der Funktion zur Bestellzustellung. Wird als Eingabe für DFHLS2WS verwendet, um dispatchOrder.wsdl und dispatchOrder.wsbind zu erzeugen.
DFH0XCP7	Definiert die Datenstruktur für die Anforderung der Funktion zur Bestellzustellung. Die Programme DFH0XODE und DFH0XWOD schließen dieses Copybook ein.
DFH0XCP8	Definiert die Datenstruktur für die Antwort der Funktion zur Bestellzustellung. Die Programme DFH0XODE und DFH0XWOD schließen dieses Copybook ein.

Tabelle 21. SDFHSAMP-Member mit COBOL-Quellcode für die Web-Service-Clientanwendung, die in CICS ausgeführt wird

Membername	Beschreibung
DFH0XCUI	Durch Transaktion-ECLI aufgerufenes Programm zum Verwalten des Sendens der BMS-Masken an den Terminalbenutzer und des Empfangens der Masken vom Terminalbenutzer. Es ist mit dem Programm DFH0XECC verknüpft.
DFH0XECC	Stellt mithilfe des Befehls EXEC CICS INVOKE WEBSERVICE ausgehende Web-Service-Anforderungen an die Basisanwendung. Einer der folgenden Web-Services wird angegeben: inquireCatalogClient inquireSingleClient placeOrderClient

Tabelle 22. SDFHSAMP-Member mit COBOL-Copybooks für die Web-Service-Clientanwendung, die in CICS ausgeführt wird. Sie werden alle von DFHWS2LS generiert und vom Programm DFH0XECC eingeschlossen.

Membername	Beschreibung
DFH0XCPA	Definiert die Datenstruktur für die Anforderung der Funktion zum Abrufen des Katalogs.
DFH0XCPB	Definiert die Datenstruktur für die Antwort der Funktion zum Abrufen des Katalogs.
DFH0XCPC	Definiert die Datenstruktur für die Anforderung der Funktion zum Abrufen einer einzelnen Instanz.
DFH0XCPD	Definiert die Datenstruktur für die Antwort der Funktion zum Abrufen einer einzelnen Instanz.
DFH0XCPE	Definiert die Datenstruktur für die Anforderung der Funktion zum Erteilen eines Auftrags.
DFH0XCPF	Definiert die Datenstruktur für die Antwort der Funktion zum Erteilen eines Auftrags.

Tabelle 23. SDFHSAMP-Member mit COBOL-Quellcode für die Wrappermodule

Membername	Beschreibung
DFH0XECC	Web-Service-Clientprogramm
DFH0XICW	Wrapperprogramm für den Service inquireCatalog.
DFH0XISW	Wrapperprogramm für den Service inquireSingle.
DFH0XPOW	Wrapperprogramm für den Service purchaseOrder.

Tabelle 24. SDFHSAMP-Member mit COBOL-Copybooks für die Wrappermodule

Membername	Beschreibung
DFH0XWC1	Definiert die Datenstruktur für die Anforderung der Funktion zum Abrufen des Katalogs. Das Programm DFH0XICW schließt dieses Copybook ein.
DFH0XWC2	Definiert die Datenstruktur für die Antwort der Funktion zum Abrufen des Katalogs. Das Programm DFH0XICW schließt dieses Copybook ein.
DFH0XWC3	Definiert die Datenstruktur für die Anforderung der Funktion zum Abrufen einer einzelnen Instanz. Das Programm DFH0XISW schließt dieses Copybook ein.

Tabelle 24. SDFHSAMP-Member mit COBOL-Copybooks für die Wrappermodule (Forts.)

Membername	Beschreibung
DFH0XWC4	Definiert die Datenstruktur für die Antwort der Funktion zum Abrufen einer einzelnen Instanz. Das Programm DFH0XISW schließt dieses Copybook ein.
DFH0XWC5	Definiert die Datenstruktur für die Anforderung der Funktion zum Erteilen eines Auftrags. Das Programm DFH0XPOW schließt dieses Copybook ein.
DFH0XWC6	Definiert die Datenstruktur für die Antwort der Funktion zum Erteilen eines Auftrags. Das Programm DFH0XPOW schließt dieses Copybook ein.

Tabelle 25. CICS-Ressourcendefinitionen

Ressourcenname	Ressourcentyp	Kommentar
BEISPIEL	CICS-Ressourcendefinitionsgruppe	CICS-Ressourcendefinitionen für die Beispielanwendung.
EGUI	TRANSACTION	Transaktion zum Aufrufen des Programms DFH0XGUI zum Starten der BMS-Schnittstelle zur Anwendung (anpassbar).
ECFG	TRANSACTION	Transaktion zum Aufrufen des Programms DFH0XCFG zum Starten der BMS-Schnittstelle der Beispielkonfiguration (anpassbar).
EXMPCAT	FILE	Dateidefinition der VSAM-Datei EXMPCAT für den Anwendungskatalog (anpassbar).
EXMPCONF	FILE	Dateidefinition der Anwendungskonfigurationsdatei EXMPCONF.

Das Katalogmanagerprogramm

Der Katalogmanager ist das Steuerprogramm für die Geschäftslogik der Beispielanwendung, und alle Interaktionen mit der Beispielanwendung durchlaufen ihn.

Um sicherzustellen, dass die Programmlogik einfach ist, wird vom Katalogmanager nur eine eingeschränkte Typüberprüfung und Fehlerwiederherstellung ausgeführt.

Der Katalogmanager unterstützt eine Reihe von Operationen. Eingabe- und Ausgabeparameter für jede Operation sind in einer einzelnen Datenstruktur definiert, die an das und von dem Programm in einem Kommunikationsbereich übergeben wird.

Kommunikationsbereichsstrukturen:

Daten werden zwischen dem Beispielfluent und Serverprogrammen unter Verwendung eines CICS-Standardkommunikationsbereichs (COMMAREA) übergeben.

Der folgende Codeauszug zeigt die Kommunikationsbereichsstruktur der Katalogmanageranwendung.


```

*   Kommunikationsbereichsstruktur des Katalogs
      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).
*   Felder 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).
*   Felder 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).
*   Felder in 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).

*   Kommunikationsbereichsstruktur des Dispatchers/Bestandsmanagers
      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).
*   Felder im 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).
*   Felder im Bestandsmanager
      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).

```

Rückgabecodes:

Jede Operation des Katalogmanagers kann eine Reihe von Rückgabecodes zurückgeben.

Tabelle 26. Rückgabecodes des Katalogmanagers

Typ	Code	Erläuterung
Allgemein	00	Funktion ohne Fehler abgeschlossen
Katalogdatei	20	Artikelreferenz nicht gefunden
	21	Fehler beim Öffnen, Lesen oder Beenden der Suche in der Katalogdatei
	22	Fehler beim Aktualisieren der Datei
Konfigurationsdatei	50	Fehler beim Öffnen der Konfigurationsdatei
	51	Datenspeichertyp war weder STUB noch VSAM
	52	Ausgehender Web-Service-Wechsel war weder Y noch N
Ferner Web-Service	30	Der Befehl EXEC CICS INVOKE WEBSERVICE hat eine INVREQ-Bedingung zurückgegeben.
	31	Der Befehl EXEC CICS INVOKE WEBSERVICE hat eine NOTFND-Bedingung zurückgegeben
	32	Der Befehl EXEC CICS INVOKE WEBSERVICE hat eine andere Bedingung als INVREQ oder NOTFND zurückgegeben
Anwendung	97	Bestand nicht ausreichend für Bestellung
	98	Bestellmenge war keine positive Zahl
	99	DFH0XCMN hat einen Kommunikationsbereich empfangen, in dem das Feld CA-REQUEST-ID keinen der folgenden Werte hatte: 01INQC, 01INQS oder 01ORDR

Operation INQUIRE CATALOG:

Diese Operation gibt eine Liste mit bis zu 15 Katalogartikeln zurück, angefangen mit dem vom Aufrufenden angegebenen Artikel.

Eingabeparameter

CA-REQUEST-ID

Eine Zeichenfolge, die die Operation angibt. Für den Befehl INQUIRE CATALOG enthält die Zeichenfolge 01INQC.

CA-LIST-START-REF

Die Referenznummer des ersten zurückzugebenden Artikels.

Ausgabeparameter**CA-RETURN-CODE**

Eine Zeichenfolge, die die Operation angibt.

CA-RESPONSE-MESSAGE

Eine lesbare Zeichenfolge, die *anzahl* ITEMS RETURNED enthält, wobei *anzahl* die Anzahl von zurückzugebenden Artikeln ist.

CA-LAST-ITEM-REF

Die Referenznummer des letzten zurückzugebenden Artikels.

CA-ITEM-COUNT

Die Anzahl von zurückgegebenen Artikeln.

CA-CAT-ITEM

Ein Array, das die Liste von zurückgegebenen Katalogartikeln enthält. Das Array hat 15 Elemente. Falls weniger als 15 Artikel zurückgegeben werden, bleiben die restlichen Arrayelemente leer.

Operation INQUIRE SINGLE ITEM:

Diese Operation gibt einen einzelnen Katalogartikel zurück, der von dem Aufrufenden angegeben wird.

Eingabeparameter**CA-REQUEST-ID**

Eine Zeichenfolge, die die Operation angibt. Für den Befehl INQUIRE SINGLE ITEM enthält die Zeichenfolge 01INQS.

CA-ITEM-REF-REQ

Die Referenznummer des zurückzugebenden Artikels.

Ausgabeparameter**CA-RETURN-CODE**

Eine Zeichenfolge, die die Operation angibt.

CA-RESPONSE-MESSAGE

Eine lesbare Zeichenfolge, die RETURNED ITEM: REF=*artikelreferenz* enthält, wobei *artikelreferenz* die Referenznummer des zurückgegebenen Artikels ist.

CA-SINGLE-ITEM

Ein Array, das in seinem ersten Element den zurückgegebenen Katalogartikel enthält.

Operation PLACE ORDER:

Diese Operation gibt eine Bestellung für einen einzelnen Artikel ab. Wenn die erforderliche Menge nicht verfügbar ist, wird eine Nachricht an den Benutzer zurückgegeben. Wenn die Bestellung erfolgreich ist, wird der Bestandsmanager informiert, welcher Artikel bestellt wurde und in welchen Mengen er bestellt wurde.

Eingabeparameter**CA-REQUEST-ID**

Eine Zeichenfolge, die die Operation angibt. Für die Operation PLACE ORDER enthält die Zeichenfolge 01ORDR.

CA-USERID

Eine achtstellige Benutzer-ID, die die Anwendung für den Versand und die Rechnungsstellung verwendet.

CA-CHARGE-DEPT

Eine achtstellige Abteilungs-ID, die die Anwendung für den Versand und die Rechnungsstellung verwendet.

CA-ITEM-REF-NUMBER

Die Referenznummer des zu bestellenden Artikels.

CA-QUANTITY-REQ

Die Anzahl von erforderlichen Artikeln.

Ausgabeparameter**CA-RETURN-CODE**

Eine Zeichenfolge, die die Operation angibt.

CA-RESPONSE-MESSAGE

Eine lesbare Zeichenfolge, die ORDER SUCCESSFULLY PLACED enthält.

Operation DISPATCH STOCK:

Diese Operation ruft das Bestands-Dispatcherprogramm auf, das wiederum die Bestellung an den Kunden sendet.

Eingabeparameter**CA-ORD-REQUEST-ID**

Eine Zeichenfolge, die die Operation angibt. Für die Operation DISPATCH ORDER enthält die Zeichenfolge 01DSP0.

CA-ORD-USERID

Eine achtstellige Benutzer-ID, die die Anwendung für den Versand und die Rechnungsstellung verwendet.

CA-ORD-CHARGE-DEPT

Eine achtstellige Abteilungs-ID, die die Anwendung für den Versand und die Rechnungsstellung verwendet.

CA-ORD-ITEM-REF-NUMBER

Die Referenznummer des zu bestellenden Artikels.

CA-ORD-QUANTITY-REQ

Die Anzahl von erforderlichen Artikeln.

Ausgabeparameter**CA-ORD-RETURN-CODE**

Eine Zeichenfolge, die die Operation angibt.

Operation NOTIFY STOCK MANAGER:

Diese Operation entscheidet anhand von Details der Bestellung, ob ein Auffüllen des Bestands erforderlich ist.

Eingabeparameter**CA-ORD-REQUEST-ID**

Eine Zeichenfolge, die die Operation angibt. Für die Operation NOTIFY STOCK MANAGER enthält die Zeichenfolge 01STK0.

CA-STK-ITEM-REF-NUMBER

Die Referenznummer des zu bestellenden Artikels.

CA-STK-QUANTITY-REQ

Die Anzahl von erforderlichen Artikeln.

Ausgabeparameter**CA-ORD-RETURN-CODE**

Eine Zeichenfolge, die die Operation angibt.

Dateistrukturen und -definitionen

Die Beispielanwendung verwendet zwei VSAM-Dateien: die Katalogdatei, die die Details zu allen Artikeln im Bestand und zur jeweiligen Bestandsmenge enthält, sowie die Konfigurationsdatei, in der die vom Benutzer ausgewählten Optionen für die Anwendung gespeichert sind.

Katalogdatei

Die Katalogdatei ist eine KSDS-VSAM-Datei, die alle Informationen zum Produktbestand enthält.

Katalogdateisätze

Datensätze in der Datei haben die folgende Struktur:

Name	COBOL-Datentyp	Beschreibung
WS-ITEM-REF-NUM	PIC 9(4)	Artikelreferenznummer
WS-DESCRIPTION	PIC X(40)	Artikelbeschreibung
WS-DEPARTMENT	PIC 9(3)	Abteilungskennnummer
WS-COST	PIC ZZZ.99	Artikelpreis
WS-IN-STOCK	PIC 9(4)	Anzahl der Artikel im Bestand
WS-ON-ORDER	PIC 9(3)	Anzahl der Artikel in Bestellung

Konfigurationsdatei

Die Konfigurationsdatei ist eine KSDS-VSAM-Datei, die Informationen zur Konfiguration der Beispielanwendung enthält.

Konfigurationsdateisätze

Die Konfigurationsdatei ist eine KSDS-VSAM-Datei, die vier unterschiedliche Datensätze enthält.

Tabelle 27. Allgemeiner Informationsdatensatz

Name	COBOL-Datentyp	Beschreibung
PROGS-KEY	PIC X(9)	Schlüsselfeld für den allgemeinen Informationsdatensatz, der EXMP-CONF enthält.
Platzhalterelement	PIC X	

Tabelle 27. Allgemeiner Informationsdatensatz (Forts.)

Name	COBOL-Datentyp	Beschreibung
DATASTORE	PIC X(4)	Eine Zeichenfolge, die den Typ des Datenspeicherprogramms angibt, das verwendet werden soll. Werte sind: STUB VSAM
Platzhalterelement	PIC X	
DO-OUTBOUND-WS	PIC X	Ein Zeichen, das angibt, ob der Zustellungsmanager eine ausgehende Web-Service-Anforderung erstellen soll. Werte sind: Y N
Platzhalterelement	PIC X	
CATMAN-PROG	PIC X(8)	Der Name des Katalogmanagerprogramms.
Platzhalterelement	PIC X	
DSSTUB-PROG	PIC X(8)	Der Name des Testdatenhandlerprogramms.
Platzhalterelement	PIC X	
DSVSAM-PROG	PIC X(8)	Der Name des VSAM-Datenhandlerprogramms.
Platzhalterelement	PIC X	
ODSTUB-PROG	PIC X(8)	Der Name des Testdispatchers für Bestellungen.
Platzhalterelement	PIC X	
ODWEBS-PROG	PIC X(8)	Der Name des Dispatcherprogramms für Bestellungen des ausgehenden Web-Service.
Platzhalterelement	PIC X	
STKMAN-PROG	PIC X(8)	Der Name des Bestandsmanagerprogramms.
Platzhalterelement	PIC X(10)	

Tabelle 28. Ausgehender URL-Datensatz

Name	COBOL-Datentyp	Beschreibung
URL-KEY	PIC X(9)	Schlüsselfeld für den allgemeinen Informationsdatensatz, der OUTBNDURL enthält.
Platzhalterelement	PIC X	

Tabelle 28. Ausgehender URL-Datensatz (Forts.)

Name	COBOL-Datentyp	Beschreibung
OUTBOUND-URL	PIC X(255)	Ausgehende URL für die Web-Service-Anforderung des Bestelldispatchers.

Tabelle 29. Katalogdatei-Informationsdatensatz

Name	COBOL-Datentyp	Beschreibung
URL-FILE-KEY	PIC X(9)	Schlüsselfeld für den allgemeinen Informationsdatensatz, der VSAM-NAME enthält.
Platzhalterelement	PIC X	
CATALOG-FILE-NAME	PIC X(8)	Name der CICS-FILE-Resource, die für die Katalogdatei verwendet wird.

Tabelle 30. Serverinformationsdatensatz

Name	COBOL-Datentyp	Beschreibung
WS-SERVER-KEY	PIC X(9)	Schlüsselfeld für den Serverinformationsdatensatz, der WS-SERVER enthält.
Platzhalterelement	PIC X	
CATALOG-FILE-NAME	PIC X(8)	Die IP-Adresse und der Port des Systems, auf dem die Beispielanwendung als Web-Service implementiert ist (nur für den CICS-Web-Service-Client).

JSON-Beispiele

Diese Beispiele erleichtern Ihnen das Verständnis von JSON-Anforderungen.

HTTP GET-Beispielanforderung mithilfe einer Abfragezeichenfolge

Dies ist ein Beispiel einer HTTP GET-Anforderung mithilfe einer Abfragezeichenfolge.

```
GET /genapp/customers?name=Joe%20Bloggs/ 1
Host: www.example.com
```

Dabei ist **1** die Abfragezeichenfolge.

HTTP-Beispielanforderung mit einem JSON-Hauptteil

Dies ist ein Beispiel einer HTTP-Anforderung mit einem JSON-Hauptteil.

```
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
}
}

```

Dabei ist Content-Length: *nn* **1** die Länge Ihrer Anforderung.

Die COBOL-Sprachstrukturzuordnung für dieses Beispiel sähe wie folgt aus:

```

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).

```

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. IBM stellt dieses Material möglicherweise auch in anderen Sprachen zur Verfügung. Für den Zugriff auf das Material in einer anderen Sprache kann eine Kopie des Produkts oder der Produktversion in der jeweiligen Sprache erforderlich sein.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

*IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes 2, avenue Gambetta
92066 Paris La Defense
France*

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuauflage veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

Die Implementierung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesen Informationen beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufs. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden und jede Ähnlichkeit mit Namen und Adressen tatsächlicher Personen oder Unternehmen ist rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellsprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Beispielprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten. Die Beispielprogramme werden ohne Wartung (auf "as-is"-Basis) und ohne jegliche Gewährleistung zur Verfügung gestellt. IBM übernimmt keine Haftung für Schäden, die durch die Verwendung der Beispielprogramme entstehen.

Informationen zu Programmierschnittstellen

Die von CICS zur Verfügung gestellte Dokumentation kann teilweise als Programmierschnittstelle betrachtet werden und zum Teil nicht.

Programmierschnittstellen, die es dem Kunden ermöglichen, Programme zur Nutzung der Services von CICS Transaction Server for z/OS, Version 5 Release 5 zu schreiben, sind in folgenden Abschnitten der Online-Produktdokumentation enthalten:

- Developing applications

- Developing system programs
- Securing overview
- Developing for external interfaces
- Reference: application development
- Reference: system programming
- Reference: connectivity

Informationen, die NICHT zur Verwendung als Programmierschnittstelle von CICS Transaction Server for z/OS, Version 5 Release 5 bestimmt sind, die aber als Programmierschnittstelle missverstanden werden können, sind in folgenden Abschnitten der Online-Produktdokumentation enthalten:

- Troubleshooting and support
- Reference: diagnostics

Wenn Sie auf die CICS-Dokumentation in Handbüchern im PDF-Format zugreifen, sind Programmierschnittstellen, die es dem Kunden ermöglichen, Programme zur Nutzung der Services von CICS Transaction Server for z/OS, Version 5 Release 5 zu schreiben, in den folgenden Handbüchern enthalten:

- Application Programming Guide und Application Programming Reference
- Business Transaction Services
- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- Supplied Transactions
- CICSplex SM Managing Workloads
- CICSplex SM Managing Resource Usage
- CICSplex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

Wenn Sie auf die CICS-Dokumentation in Handbüchern im PDF-Format zugreifen, sind Informationen, die NICHT zur Verwendung als Programmierschnittstelle von CICS Transaction Server for z/OS, Version 5 Release 5 bestimmt sind, die aber als Programmierschnittstelle missverstanden werden können, in den folgenden Handbüchern enthalten:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSplex SM Problem Determination Guide

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation in den USA und/oder anderen Ländern. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite Copyright and trademark information unter www.ibm.com/legal/copytrade.shtml.

Adobe, das Adobe-Logo, PostScript und das PostScript-Logo sind Marken oder eingetragene Marken von Adobe Systems Incorporated in den USA und/oder anderen Ländern.

Intel, das Intel-Logo, Intel Inside, das Intel Inside-Logo, Intel Centrino, das Intel Centrino-Logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium und Pentium sind Marken oder eingetragene Marken der Intel Corporation oder ihrer Tochtergesellschaften in den USA oder anderen Ländern.

Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Nutzungsbedingungen für die Produktdokumentation

Die Berechtigungen zur Nutzung dieser Veröffentlichungen werden Ihnen auf der Basis der folgenden Bedingungen gewährt.

Anwendbarkeit

Diese Bedingungen sind eine Ergänzung der Nutzungsbedingungen auf der IBM Website.

Persönliche Nutzung

Sie dürfen diese Veröffentlichungen für Ihre persönliche, nicht kommerzielle Nutzung unter der Voraussetzung vervielfältigen, dass alle Eigentumsvermerke erhalten bleiben. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM nicht weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Kommerzielle Nutzung

Sie dürfen diese Veröffentlichungen nur innerhalb Ihres Unternehmens und unter der Voraussetzung, dass alle Eigentumsvermerke erhalten bleiben, vervielfältigen, weitergeben und anzeigen. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM außerhalb Ihres Unternehmens weder vervielfältigen, weitergeben oder anzeigen noch abgeleitete Werke davon erstellen.

Rechte

Abgesehen von den hier gewährten Berechtigungen werden keine weiteren Berechtigungen, Lizenzen oder Rechte (veröffentlicht oder stillschweigend) in Bezug auf die Veröffentlichungen oder darin enthaltene Informationen, Daten, Software oder geistiges Eigentum gewährt.

IBM behält sich das Recht vor, die hierin gewährten Berechtigungen nach eigenem Ermessen zurückzuziehen, wenn sich die Nutzung der Veröffentlichungen für IBM als nachteilig erweist oder wenn die obigen Nutzungsbestimmungen nicht genau befolgt werden.

Sie dürfen diese Informationen nur in Übereinstimmung mit allen anwendbaren Gesetzen und Vorschriften, einschließlich aller US-amerikanischen Exportgesetze und Verordnungen, herunterladen und exportieren.

IBM übernimmt keine Gewährleistung für den Inhalt dieser Veröffentlichungen. Diese Veröffentlichungen werden auf der Grundlage des gegenwärtigen Zustands (auf "as-is"-Basis) und ohne eine ausdrückliche oder stillschweigende Gewährleistung für die Handelsüblichkeit, die Verwendbarkeit für einen bestimmten Zweck oder die Freiheit von Rechten Dritter zur Verfügung gestellt.

IBM Online-Datenschutzerklärung

IBM Softwareprodukte, einschließlich Software as a Service-Lösungen („Softwareangebote“), können Cookies oder andere Technologien verwenden, um Informationen zur Produktnutzung zu erfassen, die Endbenutzererfahrung zu verbessern und Interaktionen mit dem Endbenutzer anzupassen oder zu anderen Zwecken. In vielen Fällen werden von den Softwareangeboten keine personenbezogenen Daten erfasst. Einige der IBM Softwareangebote können Sie jedoch bei der Erfassung personenbezogener Daten unterstützen. Wenn dieses Softwareangebot Cookies zur Erfassung personenbezogener Daten verwendet, sind nachfolgend nähere Informationen über die Verwendung von Cookies durch dieses Angebot zu finden:

Für die Webbenutzerschnittstelle von CICSplex System Manager (Hauptschnittstelle):

Abhängig von den implementierten Konfigurationen kann dieses Softwareangebot Sitzungscookies und persistente Cookies zum Erfassen der Benutzernamen und anderer personenbezogener Daten einzelner Benutzer für das Sitzungsmanagement, die Authentifizierung, einen besseren Bedienungskomfort, zur Nutzungsüberwachung und für andere funktionale Zwecke verwenden. Diese Cookies können nicht inaktiviert werden.

Für die Webbenutzerschnittstelle von CICSplex System Manager (Datenschnittstelle): Abhängig von den implementierten Konfigurationen kann dieses Softwareangebot Sitzungscookies und persistente Cookies zum Erfassen der Benutzernamen und anderer personenbezogener Daten einzelner Benutzer für das Sitzungsmanagement, die Authentifizierung, einen besseren Bedienungskomfort, zur Nutzungsüberwachung und für andere funktionale Zwecke verwenden. Diese Cookies können nicht inaktiviert werden.

Für die Webbenutzerschnittstelle von CICSplex System Manager ("hello world"-Seite):

Abhängig von den implementierten Konfigurationen kann dieses Softwareangebot Sitzungscookies verwenden, die keine personenbezogenen Daten erfassen. Diese Cookies können nicht inaktiviert werden.

Für CICS Explorer:

Abhängig von den implementierten Konfigurationen kann dieses Softwareangebot persistente Vorgaben und Sitzungsvorgaben zum Erfassen der Benutzernamen und Kennwörter von Benutzern für das Sitzungsmanagement, die Authentifizierung und zur Single Sign-on-Konfiguration (einmalige Anmeldung) verwenden. Diese Vorgaben können nicht inaktiviert werden, auch wenn die Speicherung eines Benutzerkennworts auf ei-

nem Datenträger in verschlüsselter Form nur aktiviert werden kann, indem der Benutzer bei der Anmeldung explizit ein Kontrollkästchen aktiviert.

Wenn es die für dieses Softwareangebot implementierten Konfigurationen Ihnen als Kunde ermöglichen, personenbezogene Daten von Endbenutzern über Cookies und andere Technologien zu erfassen, müssen Sie sich zu allen gesetzlichen Bestimmungen in Bezug auf eine solche Datenerfassung, einschließlich aller Mitteilungspflichten und Zustimmungsanforderungen, rechtlich beraten lassen.

Weitere Informationen zur Nutzung verschiedener Technologien, einschließlich Cookies, für diese Zwecke finden Sie unter IBM Privacy Policy und in der IBM Online Privacy Statement im Abschnitt „Cookies, Web-Beacons und sonstige Technologien“ sowie auf der Seite IBM Software Products and Software-as-a-Service Privacy Statement.

Index

Sonderzeichen

<addressing>
 Pipelinekonfigurationselement 113
<apphandler_class>
 Pipelinekonfigurationselement 108
<apphandler>
 Pipelinekonfigurationselement 109
<auth_token_type>
 Pipelinekonfigurationselement 146
<authentication>
 Pipelinekonfigurationselement 138
<cics_json_handler_java>
 Pipelinekonfigurationselement 114
<cics_mtom_handler>
 Pipelinekonfigurationselement 131
<cics_soap_1.1_handler>
 Pipelinekonfigurationselement 114
<cics_soap_1.1_handler_java>
 Pipelinekonfigurationselement 116
<cics_soap_1.2_handler>
 Pipelinekonfigurationselement 119
<cics_soap_1.2_handler_java>
 Pipelinekonfigurationselement 121
<default_http_transport_handler_list>
 Pipelinekonfigurationselement 123
<default_mq_transport_handler_list>
 Pipelinekonfigurationselement 124
<default_target>
 Pipelinekonfigurationselement 129
<default_transport_handler_list>
 Pipelinekonfigurationselement 124
<dfhmtom_configuration>
 Pipelinekonfigurationselement 132
<dfhwsse_configuration>
 Pipelinekonfigurationselement 136
<encrypt_body>
 Pipelinekonfigurationselement 148
<handler>
 Pipelinekonfigurationselement 125
<jvmserver>
 Pipelinekonfigurationselement 126
<mime_options>
 Pipelinekonfigurationselement 135
<mtom>
 Pipelinekonfigurationselement 130
<mtom_options>
 Pipelinekonfigurationselement 132
<named_transport_entry>
 Pipelinekonfigurationselement 108
<namespace>
 Pipelinekonfigurationselement 113
<provider_pipeline>
 Pipelinekonfigurationselement 109
<provider_pipeline_json>
 Pipelinekonfigurationselement 110
<repository>
 Pipelinekonfigurationselement 126
<requester_pipeline>
 Pipelinekonfigurationselement 112
<service>
 Pipelinekonfigurationselement 127

<service_handler_list>
 Pipelinekonfigurationselement 127
<service_parameter_list>
 Pipelinekonfigurationselement 128
<sign_body>
 Pipelinekonfigurationselement 148
<sts_authentication>
 Pipelinekonfigurationselement 143
<sts_endpoint>
 Pipelinekonfigurationselement 147
<terminal_handler>
 Pipelinekonfigurationselement 111
<transport>
 Pipelinekonfigurationselement 129
<transport_handler_list>
 Pipelinekonfigurationselement 112
<wsse_handler>
 Pipelinekonfigurationselement 136
<xop_options>
 Pipelinekonfigurationselement 134

A

Adressierung
 Pipelinekonfigurationselement 113
Algorithmus 655, 657
Angepasster Sicherheitshandler 666
Anwendung
 in XML umsetzen 311
Anwendung für Service-Requester erstellen 519
Anwendungsdaten in JSON umsetzen 515
Anwendungshandler
 Pipelinekonfigurationselement 108
App-Handler
 Pipelinekonfigurationselement 109
Assistent, JSON 250, 434
Assistent, Web-Services 522
Assistent, XML 312
Atom-Feed
 XML-Bindung
 erstellen 427
atomare Transaktion 207, 214
 CICS konfigurieren 210
 Registrierungsservices 208
 Service-Provider konfigurieren 212
 Service-Requester konfigurieren 213
 Status 215
Authentifizierung
 Pipelinekonfigurationselement 138
Axis2 62

B

Befehlszeile 688
Beispiel 695
Benutzercontainer 198
Binärdaten in XML umsetzen 430

Binärer Anhang
 Pipelinekonfiguration 130
Bindungsdatei
 XML-Bindung 427

C

C und C++
 Zuordnung zu JSON-Schema 478, 481, 581, 584
 Zuordnung zu XML-Schema 351, 354
cics_json_handler_java
 Pipelinekonfigurationselement 114
cics_mtom_handler
 Pipelinekonfigurationselement 131
cics_soap_1.1_handler
 Pipelinekonfigurationselement 114
cics_soap_1.1_handler_java
 Pipelinekonfigurationselement 116
cics_soap_1.2_handler
 Pipelinekonfigurationselement 119
cics_soap_1.2_handler_java
 Pipelinekonfigurationselement 121
CICS-Web-Services
 Unicode 420, 510, 615
 UTF-16 420, 510, 615
COBOL
 variabel wiederkehrende Inhalte 386, 415
 Zuordnung zu JSON-Schema 461, 470, 565, 573
 Zuordnung zu XML-Schema 335, 344
Container
 DFH-EXIT-HEADER1 176
 DFH-HANDLERPLIST 177
 DFH-SERVICEPLIST 177
 DFHERROR 167
 DFHFUNCTON 169
 DFHHTTPSTATUS 171
 DFHJSON-DATA 692
 DFHJSON-JSON 691
 DFHJSON-JVMSERV 692
 DFHJSON-TRANSFRM 692
 DFHMEDIATYPE 171
 DFHNORESPONSE 172
 DFHREQUEST 172
 DFHRESPONSE 173
 DFHSAML-AnnnVmmm 193
 DFHSAML-ASSQNAME 193
 DFHSAML-ATTRAnnn 194
 DFHSAML-ATTRFnnn 194
 DFHSAML-ATTRNnnn 194
 DFHSAML-ATTRSnnn 194
 DFHSAML-ATTRYnnn 194
 DFHSAML-AUDNRnnn 194
 DFHSAML-AUTHMETH 195
 DFHSAML-CERTIDN 195
 DFHSAML-CERTSDN 195
 DFHSAML-CERTSDNUM 195
 DFHSAML-CONFMETH 195
 DFHSAML-COUNTS 195

- Container (Forts.)
 - DFHSAML-FLAGS 195
 - DFHSAML-ISSUER 195
 - DFHSAML-NAMID 195
 - DFHSAML-NAMIDF 196
 - DFHSAML-NAMIDQ 196
 - DFHSAML-NAMIDSP 196
 - DFHSAML-NAMIDSPQ 196
 - DFHSAML-OUTTOKEN 196
 - DFHSAML-PROXYnnn 196
 - DFHSAML-RESPONSE 196
 - DFHSAML-SAMLID 196
 - DFHSAML-SUBJADDR 196
 - DFHSAML-SUBJDNS 197
 - DFHSAML-TIMES 197
 - DFHWS-APPHANDLER 177, 179
 - DFHWS-CCSID 173
 - DFHWS-CID-DOMAIN 185
 - DFHWS-DATA 178
 - DFHWS-FAULT 178
 - DFHWS-IDTOKEN 191
 - DFHWS-LOCATION 178
 - DFHWS-MEP 179
 - DFHWS-MTOM-IN 186
 - DFHWS-MTOM-OUT 187
 - DFHWS-NODEJSAPP 173
 - DFHWS-PIPELINE 180
 - DFHWS-RESPWAIT 180
 - DFHWS-RESTOKEN 191
 - DFHWS-SERVICEURI 191
 - DFHWS-SOAPLEVEL 180
 - DFHWS-STSACTION 192
 - DFHWS-STSFAULT 192
 - DFHWS-STSREASON 192
 - DFHWS-STSURI 192
 - DFHWS-TOKENTYPE 193
 - DFHWS-TRANID 180
 - DFHWS-URI 181
 - DFHWS-USERID 185
 - DFHWS-WEBSERVICE 185
 - DFHWS-XOP-IN 188
 - DFHWS-XOP-OUT 188
 - Kanalbeschreibung 297, 623
 - Kontextcontainer
 - DFH-HANDLERPLIST 177
 - DFH-SERVICEPLIST 177
 - DFHWS-APPHANDLER 177, 179
 - DFHWS-DATA 178
 - DFHWS-FAULT 178
 - DFHWS-PIPELINE 180
 - DFHWS-SOAPLEVEL 180
 - DFHWS-STSREASON 192
 - DFHWS-TRANID 180
 - DFHWS-URI 181
 - DFHWS-USERID 185
 - DFHWS-WEBSERVICE 185
 - SAML 193
 - Steuercontainer
 - DFHERROR 167
 - DFHFUNCTION 169
 - DFHHTTPSTATUS 171
 - DFHMEDIATYPE 171
 - DFHNORESPONSE 172
 - DFHREQUEST 172
 - DFHRESPONSE 173
 - DFHWS-CCSID 173
 - DFHWS-NODEJSAPP 173

- Container DFH-EXIT-HEADER1 176
- Container DFH-HANDLERPLIST 177
- Container DFH-SERVICEPLIST 177
- Container DFHERROR 167
- Container DFHFUNCTION 169
- Container DFHHTTPSTATUS 171
- Container DFHJSON-DATA 692
- Container DFHJSON-ERROR 692
- Container DFHJSON-ERRORMSG 693
- Container DFHJSON-JSON 691
- Container DFHJSON-JVMSEVR 692
- Container DFHJSON-TRANSFRM 692
- Container DFHMEDIATYPE 171
- Container DFHNORESPONSE 172
- Container DFHREQUEST 172
- Container DFHRESPONSE 173
- Container DFHSAML-AnnnVmmm 193
- Container DFHSAML-ASSQNAME 193
- Container DFHSAML-ATTRAnnn 194
- Container DFHSAML-ATTRFnnn 194
- Container DFHSAML-ATTRNnnn 194
- Container DFHSAML-ATTRSnnn 194
- Container DFHSAML-ATTRYnnn 194
- Container DFHSAML-AUDNRnnn 194
- Container DFHSAML-AUTHMETH 195
- Container DFHSAML-CERTIDN 195
- Container DFHSAML-CERTSDN 195
- Container DFHSAML-CERTSNUM 195
- Container DFHSAML-CONFMETH 195
- Container DFHSAML-COUNTS 195
- Container DFHSAML-FLAGS 195
- Container DFHSAML-ISSUER 195
- Container DFHSAML-NAMID 195
- Container DFHSAML-NAMIDF 196
- Container DFHSAML-NAMIDQ 196
- Container DFHSAML-NAMIDSP 196
- Container DFHSAML-NAMIDSPQ 196
- Container DFHSAML-OUTTOKEN 196
- Container DFHSAML-PROXYnnn 196
- Container DFHSAML-RESPONSE 196
- Container DFHSAML-SAMLID 196
- Container DFHSAML-SUBJADDR 196
- Container DFHSAML-SUBJDNS 197
- Container DFHSAML-TIMES 197
- Container DFHWS-APPHANDLER 177, 179
- Container DFHWS-CCSID 173
- Container DFHWS-CID-DOMAIN 185
- Container DFHWS-DATA 178
- Container DFHWS-FAULT 178
- Container DFHWS-IDTOKEN 191
- Container DFHWS-LOCATION 178
- Container DFHWS-MEP 179
- Container DFHWS-MTOM-IN 186
- Container DFHWS-MTOM-OUT 187
- Container DFHWS-NODEJSAPP 173
- Container DFHWS-PIPELINE 180
- Container DFHWS-RESPWAIT 180
- Container DFHWS-RESTOKEN 191
- Container DFHWS-SERVICEURI 191
- Container DFHWS-SOAPLEVEL 180
- Container DFHWS-STSACTION 192
- Container DFHWS-STSFAULT 192
- Container DFHWS-STSREASON 192
- Container DFHWS-STSURI 192
- Container DFHWS-TOKENTYPE 193
- Container DFHWS-TRANID 180

- Container DFHWS-URI 181
- Container DFHWS-USERID 185
- Container DFHWS-WEBSERVICE 185
- Container DFHWS-WSDL-CTX 188
- Container DFHWS-XOP-IN 188
- Container DFHWS-XOP-OUT 188

D

- default_http_transport_handler_list
 - Pipelinekonfigurationselement 123, 126
- default_mq_transport_handler_list
 - Pipelinekonfigurationselement 124
- default_target
 - Pipelinekonfigurationselement 129
- default_transport_handler_list
 - Pipelinekonfigurationselement 124
- DFHAXIS 692
- DFHJS2LS
 - katalogisierte Prozedur 263, 278, 444
- DFHLS2SC, CICS-XML-Assistentenprogramm 427
- dfhmtom_configuration
 - Pipelinekonfigurationselement 132
- DFHSC2LS
 - katalogisierte Prozedur 321
- DFHWS2LS
 - katalogisierte Prozedur 539
- dfhwsse_configuration
 - Pipelinekonfigurationselement 136
- Dienstprogramm
 - JSON-Assistent 250, 434
 - Web-Service-Assistent 522
 - XML-Assistent 312
- Direktmodus 218
- Dynamisches Routing
 - in einem Service-Provider 164
 - in einem Terminal-Handler 164
- dynamisches Scripting 684

E

- Einschränkungen zur Laufzeit 200
- encrypt_body
 - Pipelinekonfigurationselement 148
- Endpunktreferenz
 - Standard 235
- Endpunktreferenz (EPR) 227
- Envelope, SOAP 21
- EXEC CICS SOAPFAULT CREATE-Befehl 627

F

- Fehler 688
- Fehler, SOAP 21
- Fehlerbehebung 684, 688
- Fehlercontainer
 - DFHJSON-ERROR 692
 - DFHJSON-ERRORMSG 693
- Funktionalität 43

G

Globale Benutzerexits 204
GLUEs 204

H

Handler
 Pipelinekonfigurationselement 125
Hauptteil, SOAP 21
Header, SOAP 21

I

Installation 684
installieren 684

J

Java 62
 Web-Service aufrufen 639
Java-basierte SOAP-Pipelines 231
JCICS-Container
 DFHJSON-DATA 692
 DFHJSON-ERROR 692
 DFHJSON-ERRORMSG 693
 DFHJSON-JSON 691
 DFHJSON-JVMSERVER 692
 DFHJSON-TRANSFRM 692
 JSON 691, 692
JSON 79, 85, 184, 684, 685, 686, 687, 688,
 691, 692, 693, 735
 in Sprachstruktur konvertieren 444
JSON-Assistent 250, 434, 685, 688
JSON-Beispiele 735
JSON in Anwendungsdaten umset-
 zen 517
JSON mithilfe der verknüpfbaren Schnitt-
 stelle umsetzen 433
JSON-Schema 293, 461, 470, 478, 481,
 489, 495, 565, 573, 581, 584, 592, 598
JVM-Server 62, 639

K

Kanalbeschreibung 297, 623
Kanalschnittstelle 297, 623
KatalogbeispielKatalogbeispiel 695
Kompatibilitätsmodus 218
Konfigurationsdatei, Pipeline 95
konfigurieren 79, 85
Kontextcontainer 176
 DFH-EXIT-HEADER1 176
 DFHWS-CID-DOMAIN 185
 DFHWS-IDTOKEN 191
 DFHWS-LOCATION 178
 DFHWS-MEP 179
 DFHWS-MTOM-IN 186
 DFHWS-MTOM-OUT 187
 DFHWS-RESPWAIT 180
 DFHWS-RESTOKEN 191
 DFHWS-SERVICEURI 191
 DFHWS-STSACTION 192
 DFHWS-STSAULT 192
 DFHWS-STSURI 192
 DFHWS-TOKENTYPE 193

Kontextcontainer (*Forts.*)
 DFHWS-WSDL-CTX 188
 DFHWS-XOP-IN 188
 DFHWS-XOP-OUT 188

L

Laufzeiteinschränkungen 200

M

maxOccurs
 in JSON-Schema 304, 395, 505
 in XML-Schema 372, 389, 401, 608
MEP 14
MIME-Nachricht
 Pipelinekonfiguration 130
mime_options
 Pipelinekonfigurationselement 135
minOccurs
 in JSON-Schema 304, 395, 505
 in XML-Schema 372, 389, 401, 608
mtom
 Pipelinekonfigurationselement 130
mtom_options
 Pipelinekonfigurationselement 132
MTOM/XOP
 Pipelinekonfiguration 130

N

Nachrichtenaustauschmuster (Message
 Exchange Pattern, MEP) 14
Nachrichtenhandler
 Nicht-Terminal 155, 156, 157
 vertrauenswürdigen Client aufru-
 fen 668
named_transport_entry
 Pipelinekonfigurationselement 108
Namensbereich
 Pipelinekonfigurationselement 113
Nicht-Terminal-Nachrichtenhandler 155,
 156, 157

P

Persistente Nachricht 55
Pipeline konfigurieren 663
Pipelinedefinition
 Service-Requester 106
Pipelinekonfiguration
 MTOM/XOP 130
 Web Services Security 135
Pipelinekonfigurationsdatei 95
Pipelinekonfigurationselement
 <addressing> 113
 <apphandler> 109
 <auth_token_type> 146
 <authentication> 138
 <cics_json_handler_java> 114
 <cics_mtom_handler> 131
 <cics_soap_1.1_handler> 114
 <cics_soap_1.1_handler_java> 116
 <cics_soap_1.2_handler> 119
 <cics_soap_1.2_handler_java> 121

Pipelinekonfigurationselement (*Forts.*)

 <default_http_transport_handler-
 _list> 123
 <default_mq_transport_handler-
 _list> 124
 <default_transport_handler_list> 124
 <dfhmtom_configuration> 132
 <dfhwsse_configuration> 136
 <encrypt_body> 148
 <handler> 125
 <jvmserver> 126
 <mime_options> 135
 <mtom> 130
 <mtom_options> 132
 <named_transport_entry> 108
 <namespace> 113
 <provider_pipeline> 109
 <provider_pipeline_json> 110
 <repository> 126
 <requester_pipeline> 112
 <service> 127
 <service_handler_list> 127
 <sign_body> 148
 <sts_authentication> 143
 <sts_endpoint> 147
 <terminal_handler> 111
 <transport> 129
 <transport_handler_list> 112
 <wsse_handler> 136
 <xop_options> 134

Pipelineverarbeitung
 anpassen 204
 URI überschreiben 206

Pipelineverarbeitung anpassen 204

PL/I

 Zuordnung zu JSON-Schema 489,
 495, 592, 598
 Zuordnung zu XML-Schema 360, 364
Problembestimmung 684
Probleme diagnostizieren
 Service-Provider 676
 Service-Requester 677
provider_pipeline
 Pipelinekonfigurationselement 109

R

RACF konfigurieren 658
requester_pipeline
 Element der Pipelinedefinition 106
 Pipelinekonfigurationselement 112
Rückgabecode 688

S

SAML
 Container 193
Schema
 Kanalbeschreibung 297, 623
Service
 Pipelinekonfigurationselement 127
service_handler_list
 Pipelinekonfigurationselement 127
service_parameter_list
 Serviceparameterliste 128

- Service-Provider
 - Probleme diagnostizieren 676
- Service-Provider-Anwendung 293
 - atomare Transaktionen verwenden 212
 - aus einer Datenstruktur erstellen 295, 620
- Service-Requester
 - Pipelinedefinition 106
 - Probleme diagnostizieren 677
- Service-Requester-Anwendung
 - atomare Transaktionen verwenden 213
- Serviceparameterliste
 - <service_parameter_list> 128
- Sicherheit für Web-Services 647
- Sicherheitscontainer 191
 - DFHSAML-AnnnVmmm 193
 - DFHSAML-ASSQNAME 193
 - DFHSAML-ATTRAnnn 194
 - DFHSAML-ATTRFnnn 194
 - DFHSAML-ATTRNnnn 194
 - DFHSAML-ATTRSnnn 194
 - DFHSAML-ATTTRYnnn 194
 - DFHSAML-AUDNRnnn 194
 - DFHSAML-AUTHMETH 195
 - DFHSAML-CERTIDN 195
 - DFHSAML-CERTSDN 195
 - DFHSAML-CERTSNUM 195
 - DFHSAML-CONF METH 195
 - DFHSAML-COUNTS 195
 - DFHSAML-FLAGS 195
 - DFHSAML-ISSUER 195
 - DFHSAML-NAMID 195
 - DFHSAML-NAMIDF 196
 - DFHSAML-NAMIDQ 196
 - DFHSAML-NAMIDSP 196
 - DFHSAML-NAMIDSPQ 196
 - DFHSAML-OUTTOKEN 196
 - DFHSAML-PROXYnnn 196
 - DFHSAML-RESPONSE 196
 - DFHSAML-SAMLID 196
 - DFHSAML-SUBJADDR 196
 - DFHSAML-SUBJDNS 197
 - DFHSAML-TIMES 197
- Sicherheitshandler
 - eigenen schreiben 666
- Sicherheitstokenservice
 - vertrauenswürdige Clientschnittstelle 654
- sign_body
 - Pipelinekonfigurationselement 148
- SOAP
 - Envelope 21
 - Fehler 21
 - Hauptteil 21
 - Header 21
 - Übersicht 16
 - Übersicht über SOAP 16
- SOAP-Fehler 627
- SOAP-Nachricht
 - Beispiel 21
 - signieren 654
 - Struktur 21
 - verschlüsseln 656
- SOAP-Nachrichten
 - gegen ein XML-Schema prüfen 641

- SOAP-Nachrichten (Forts.)
 - XML-Schema
 - SOAP-Nachricht überprüfen 641
- SOAP-Nachrichten überprüfen 641
- SOAP-Nachrichtenpfad 8
- SOAP-Pipelines 62
- Sprachstrukturen in XML umsetzen 427
- Standardendpunktreferenz 235
 - WSDL 1.1 235
- Stapeldienstprogramm 250
 - JSON-Assistent 434
 - Web-Service-Assistent 522
 - XML-Assistent 312
- Steuercontainer 167
- sts_authentication
 - Pipelinekonfigurationselement 143
- sts_endpoint
 - Pipelinekonfigurationselement 147

T

- terminal_handler
 - Pipelinekonfigurationselement 111
- TRANSFORM DATATOXML 430
- TRANSFORM XMLTODATA 431
- Transport
 - Pipelinekonfigurationselement 129
- transport_handler_list
 - Pipelinekonfigurationselement 112
- Typ des Authentifizierungstoken
 - Pipelinekonfigurationselement 146

U

- Unterstützung persistenter Nachrichten 56
- URI
 - für WebSphere MQ-Transport 54
- URI überschreiben 206

V

- Variable Arrays 304, 372, 389, 395, 401, 505, 608
- Vertrauenswürdigen Client aufrufen 668
- Vertrauenswürdiger Client
 - aufrufen 668
 - Schnittstelle 654

W

- Web-Service-Adressierung
 - <wsa:Action> 236
 - Adressierungshandler 231, 232
 - DFHWS-URI 227
 - DFHWSADH 231, 232
 - EPR 227
 - explizite Aktionen 234, 236
 - MAP 227
 - Parameter 557
 - Provider-Pipeline-Konfiguration 232
 - Requester-Pipeline-Konfiguration 231
 - Requester-Service 234
 - Spezifikation 226
 - Standardaktionen 234, 237, 239

- Web-Service-Adressierung (Forts.)
 - Standardendpunktreferenz 234, 235
 - Unterstützung 226
 - WSADDR-EPR-ANY 557
 - WSDL 1.1 237
 - WSDL 2.0 239
- Web-Service-Assistent 522
 - Service-Provider-Anwendung erstellen 295, 620
- Web-Service aus Java aufrufen 639
- Web-Service-Erkennung 49
- Web-Service-Fehler 676, 677
- Web Services Security
 - Pipelinekonfiguration 135
- Web Services Security (WSS) 647, 658, 663
- wiederkehrende Inhalte 386, 415
- Workload-Management
 - in einem Service-Provider 164
 - in einem Terminal-Handler 164
- WS-Addressing
 - <wsa:Action> 236
 - Adressierungshandler 231, 232
 - DFHWS-URI 227
 - DFHWSADH 231, 232
 - EPR 227
 - explizite Aktionen 236
 - MAP 227
 - Parameter 557
 - Provider-Pipeline-Konfiguration 232
 - Requester-Pipeline-Konfiguration 231
 - Spezifikation 226
 - Standardaktionen 237, 239
 - Standardendpunktreferenz 235
 - WSADDR-EPR-ANY 557
 - WSDL 1.1 237
 - WSDL 2.0 239
- WS-AT 207
- WSDL
 - abfragen 49
 - in Sprachstruktur konvertieren 263, 278, 321, 539
 - und Anwendungsdatenstruktur 12
 - Web-Service-Adressierung 234
- WSDL 1.1
 - Standardendpunktreferenz 235
- WSDL-Dokument
 - Leerzeichen 393, 418, 612
 - variable Länge 393, 418, 612
- wsse_handler
 - Pipelinekonfigurationselement 136

X

- XML
 - abfragen 422
 - Assistent 311
 - Datentypen 423
 - in Daten umsetzen 311
 - parsen 423
 - umsetzen 423
 - xsd:any-Datentypen 425
- XML abfragen 422
- XML-Assistent 312
 - DFHLS2SC 427
- XML-Bindung 427
- XML in Binärdaten umsetzen 431

- XML parsen 423
- XML-Schema 335, 344, 351, 354, 360, 364
 - Leerzeichen 393, 418, 612
 - variable Länge 393, 418, 612
- XML-Schemas zu Anwendungsdaten zuordnen 429
- XML umsetzen 423
- xop_options
 - Pipelinekonfigurationselement 134
- xsd:any-Datentypen 425

Z

- z/OS Connect 43, 79, 83, 85, 88
 - konfigurieren 84, 91
 - Sicherheit 669
 - Übersicht 38
- zAAP 62
- zosConnectService 79, 85
- Zuordnung zu C und C++ 351, 354, 478, 481, 581, 584
- Zuordnung zu COBOL 335, 344, 461, 470, 565, 573
- Zuordnung zu PL/I 360, 364, 489, 495, 592, 598
- Zuordnungen aus einem JSON-Schema generieren 514
- Zuordnungen aus einer Sprachstruktur generieren 512

