

CICS Transaction Server for z/OS
バージョン 4 リリース 1



Web サービス・ガイド

CICS Transaction Server for z/OS
バージョン 4 リリース 1



Web サービス・ガイド

— お願い —

本書および本書で紹介する製品をご使用になる前に、 393 ページの『特記事項』に記載されている情報をお読みください。

本書は、CICS Transaction Server for z/OS パージョン 4 リリース 1 (製品番号 5697-E93)、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： SC34-7020-00

CICS Transaction Server for z/OS

Version 4 Release 1

Web Services Guide

発行： 日本アイ・ピー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2009.7

© Copyright International Business Machines Corporation 2005, 2009.

目次

前書き	vii
本書の内容	vii
本書の対象読者	vii

CICS Transaction Server for z/OS バージョン 4 リリース 1 の変更点 ix

第 1 章 CICS および Web サービス	1
Web サービスとは	2
Web サービスがビジネスに役立つ仕組み	2
Web サービス用語	3

第 2 章 Web サービスのアーキテクチャ	7
Web サービス記述	8
サービスの発行	10

第 3 章 SOAP とは	13
SOAP メッセージの構造	13
SOAP ヘッダー	15
SOAP 本体	17
SOAP 障害	17
SOAP ノード	19
SOAP メッセージ・パス	19

第 4 章 CICS が Web サービスをサポートする仕組み	21
メッセージ・ハンドラーおよびパイプライン	21
トランスポート関連ハンドラー	23
フローの中断	24
サービス・プロバイダー・パイプライン	24
サービス・リクエスター・パイプライン	25
CICS パイプラインおよび SOAP	26
SOAP メッセージおよびアプリケーション・データ構造	27
WSDL およびアプリケーション・データ構造	30
WSDL およびメッセージ交換パターン	32
Web サービスのバインディング・ファイル	34
外部標準	34
SOAP 1.1 および 1.2	35
SOAP 1.1 Binding for MTOM 1.0	35
SOAP Message Transmission Optimization Mechanism (MTOM)	35
Web Services Addressing 1.0	36
Web Services Atomic Transaction バージョン 1.0	36
Web Services Coordination バージョン 1.0	37
Web サービス記述言語バージョン 1.1 および 2.0	37
Web Services Security: SOAP Message Security	38
Web Services Trust Language	38
WSDL 1.1 Binding Extension for SOAP 1.2	38

WS-I Basic Profile バージョン 1.1	39
WS-I Simple SOAP Binding Profile バージョン 1.0	39
XML (Extensible Markup Language) バージョン 1.0	40
XML-binary Optimized Packaging (XOP)	40
XML Encryption Syntax and Processing	40
XML-Signature Syntax and Processing	40
CICS の Web サービス標準への準拠	41

第 5 章 Web サービス入門	47
Web サービス使用の計画立案	47
サービス・プロバイダー・アプリケーションの計画	49
サービス・リクエスター・アプリケーションの計画	51

第 6 章 Web サービスに応じた CICS システムの構成	55
Web サービスのための CICS リソース	55
WebSphere MQ トランスポートを使用するための CICS の構成	58
WebSphere MQ トランスポート	59
サービス・プロバイダーでのローカル・キューの定義	60
サービス・リクエスターでのローカル・キューの定義	62
WMQ トランスポートの URI	62
永続メッセージをサポートするための CICS の構成	63
永続メッセージの処理	64

第 7 章 Web サービス・インフラストラクチャーの作成	67
サービス・プロバイダーに応じた CICS インフラストラクチャーの作成	67
サービス・リクエスターに応じた CICS インフラストラクチャーの作成	68
パイプライン構成ファイル	70
トランスポート関連ハンドラー	72
サービス・プロバイダーのパイプライン定義	75
サービス・リクエスターのパイプライン定義	77
サービス・プロバイダーのみで使用されるエレメント	78
サービス・リクエスターのみで使用されるエレメント	82
サービス・プロバイダーおよびサービス・リクエスターで使用されるエレメント	82
MTOM/XOP に合わせたパイプライン構成	92
WS-Security に合わせたパイプライン構成	97

メッセージ・ハンドラー	109
メッセージ・ハンドラー・プロトコル	110
独自のメッセージ・ハンドラーの提供	113
端末以外のメッセージ・ハンドラーでのメッセージの処理	113
端末メッセージ・ハンドラーでのメッセージの処理	116
エラーの処理	117
メッセージ・ハンドラー・インターフェース	118
SOAP メッセージ・ハンドラー	118
ヘッダー処理プログラム	119
ヘッダー処理プログラム・インターフェース	121
SOAP ハンドラー・インターフェース	123
端末ハンドラーでのインバウンド要求の動的ルーティング	124
パイプラインで使用されるコンテナ	125
制御コンテナ	126
コンテナがパイプライン・プロトコルを制御する方法	133
コンテキスト・コンテナ	135
セキュリティー・コンテナ	146
CICS によって生成されるコンテナ	148
ユーザー・コンテナ	149

第 8 章 Web サービスの作成 151

CICS Web サービス・アシスタント	152
DFHLS2WS: 高水準言語から WSDL への変換	153
DFHWS2LS: WSDL から高水準言語への変換	166
構文表記	180
CICS アシスタントのマッピング・レベル	181
高水準言語と XML のスキーマ・マッピング	186
Web サービス・アシスタントを使用した Web サービス・プロバイダーの作成	231
Web サービス記述を基にしたサービス・プロバイダー・アプリケーションの作成	232
データ構造を基にしたサービス・プロバイダー・アプリケーションの作成	234
チャンネル記述文書の作成	237
生成した Web サービス記述文書のカスタマイズ	238
SOAP 障害の送信	240
Web サービス・アシスタントを使用した Web サービス・リクエスターの作成	242
ツールを使用した Web サービスの作成	244
XML を認識する独自の Web サービス・アプリケーションの作成	245
XML 認識サービス・プロバイダー・アプリケーション	246
XML 認識サービス・リクエスター・アプリケーションの作成	248
SOAP メッセージの検証	250

第 9 章 Web サービスのための実行時処理 253

CICS による、Web サービス・アシスタントを使用して配置したサービス・プロバイダー・プログラムの呼び出し方法	253
---	-----

Web サービス・アシスタントを使用して配置したアプリケーションからの Web サービスの呼び出し	254
Web サービス・アシスタントによって生成されるコードの実行時の制限	255
リクエスター・パイプライン処理を制御するためのオプション	258
リクエスター・パイプライン処理を制御するためのオプション	260
URI を使用したリクエスター・パイプライン処理の制御	261

第 10 章 Web サービス・トランザクションのサポート 265

登録サービス	265
Web サービス・トランザクションに合わせた CICS の構成	268
Web サービス・トランザクションに合わせたサービス・プロバイダーの構成	270
Web サービス・トランザクションに合わせたサービス・リクエスターの構成	271
SOAP メッセージがアトミック・トランザクションの一部であるかどうかの判別	273
アトミック・トランザクションの進行の確認	274

第 11 章 バイナリー・データの MTOM/XOP 最適化のサポート 277

MTOM/XOP および SOAP	277
CICS における MTOM メッセージとバイナリー添付ファイル	279
インバウンド MTOM メッセージの処理	280
アウトバウンド MTOM メッセージの処理	281
MTOM/XOP 使用の際の制限	283
MTOM/XOP をサポートするための CICS の構成	284

第 12 章 Web Services Addressing のサポート 287

Web Services Addressing の概要	288
Web Services Addressing に合わせたパイプラインの構成	291
Web Services Addressing 用に WSDL を構成する	293
DFHWS2LS が WS-Addressing をサポートするために必要とするパラメーター	293
デフォルト EPR の設定	295
WSDL 文書での <wsa:Action> プロパティの定義	295
メッセージ交換	299
WS-Addressing の必須のメッセージ・アドレッシング・プロパティ	301
Web Services Addressing のセキュリティー	303
Web Services Addressing の例	304
Web Services Addressing の用語	308

第 13 章 Web サービスを保護するためのサポート 311

Web Services Security を実装するための前提条件	311
------------------------------------	-----

Web サービスを保護するための計画	312
SOAP メッセージを保護するためのオプション	313
Security Token Service を使用した認証	315
Trust クライアント・インターフェース	317
SOAP メッセージへの署名	318
署名アルゴリズム	318
署名された SOAP メッセージの例	319
暗号化された SOAP メッセージの CICS サポート	320
暗号化アルゴリズム	320
暗号化された SOAP メッセージの例	321
Web Services Security に合わせた RACF の構成	322
Web Services Security に合わせたパイプラインの構成	324
カスタムのセキュリティ・ハンドラーの作成	328
メッセージ・ハンドラーからの Trust クライアントの起動	329

第 14 章 Web サービス・アシスタント と WSRR の相互運用性 333

Web サービス・アシスタントと WSRR で SSL を 使用する方法の例	333
---	-----

第 15 章 問題の診断 335

配置エラーの診断	335
サービス・プロバイダーのランタイム・エラーの診 断	337
サービス・リクエストのランタイム・エラーの診 断	339
MTOM/XOP エラーの診断	341
データ変換エラーの診断	343
データ変換エラーが起こる理由	344
トレース・ポイントにおける変換エラー	345
変換エラーについての SOAP 障害メッセージ	346

第 16 章 CICS カタログ・マネージャー の実例アプリケーション 347

ベース・アプリケーション	347
BMS プレゼンテーション・マネージャー	349
データ・ハンドラー	349
ディスパッチ・マネージャー	349
注文発送エンドポイント	350
在庫マネージャー	350

アプリケーションの構成	350
BMS インターフェースによる実例アプリケーショ ンの実行	350
ベース・アプリケーションのインストールおよびセ ットアップ	352
VSAM データ・セットの作成および定義	352
3270 インターフェースの定義	353
インストールの完了	355
実例アプリケーションの構成	355
実例アプリケーションに対する Web サービス・サ ポート	357
コード・ページ・サポートの構成	362
Web サービス・クライアントおよびラッパー・ プログラムの定義	362
Web サービス・サポートのインストール	363
Web クライアントの構成	368
Web サービス対応アプリケーションの実行	371
実例アプリケーションの配置	375
プログラム・インターフェースの抽出	375
Web サービス・アシスタント・プログラム DFHLS2WS の実行	377
生成される WSDL 文書の例	379
Web サービス・バインディング・ファイルの配 置	380
ベース・アプリケーションのコンポーネント	381
カタログ・マネージャー・プログラム	384
ファイル構造と定義	389
カタログ・ファイル	389
構成ファイル	389

特記事項 393

商標	394
--------------	-----

参考文献 395

CICS Transaction Server for z/OS の CICS ブック	395
CICS Transaction Server for z/OS の CICSplex SM ブック	396
他の CICS 資料	396

アクセシビリティ 397

索引 399

前書き

本書の内容

本書では、CICS® での Web サービスの用法について説明します。

本書の対象読者

本書の対象読者は次のとおりです。

- Web サービス環境での CICS アプリケーションの配置を検討している計画担当者および設計者。
- Web サービスをサポートするために CICS の構成を担当しているシステム・プログラマー。
- Web サービス環境で配置されるアプリケーションを担当するアプリケーション・プログラマー。

CICS Transaction Server for z/OS バージョン 4 リリース 1 の変更点

このリリースに加えられた変更点に関する情報は、インフォメーション・センターの「リリース・ガイド」または以下の資料を参照してください。

- *CICS Transaction Server for z/OS* リリース・ガイド
- *CICS Transaction Server for z/OS V3.2* からのアップグレード
- *CICS Transaction Server for z/OS V3.1* からのアップグレード
- *CICS Transaction Server for z/OS V2.3* からのアップグレード

第 1 章 CICS および Web サービス

Web サービスは、ワールド・ワイド・ウェブ (WWW) がプログラムとエンド・ユーザー間の対話に果たしてきた役割を、プログラム相互間の対話に提供できます。Web サービスを使用することで、これまでに比べ、迅速に効率良く低コストでアプリケーションを統合できます。

CICS Transaction Server for z/OS[®] は、以下に示すような Web サービスの包括的なサポートを提供します。

- CICS アプリケーションは、サービス・リクエスター、サービス・プロバイダー、またはその両方として異種の Web サービス環境に関与できます。
- CICS は、HTTP および WebSphere MQ トランスポート・プロトコルをサポートします。
- CICS Transaction Server for z/OS には、CICS Web サービス・アシスタントが組み込まれています。これは、WSDL サービス記述を高水準プログラム言語のデータ構造にマップしたり、その逆方向にマップしたりするときに役立つ 1 組のユーティリティー・プログラムです。ユーティリティー・プログラムは、以下のプログラム言語をサポートしています。

COBOL

PL/I

C

C++

- Web サービスの CICS サポートは、以下のようなオープン・スタンダードに準拠しています。

SOAP 1.1 および 1.2

HTTP 1.1

WSDL 1.1 および 2.0

- Web サービスの CICS サポートは、WS-I Basic Profile バージョン 1.1 を含む多くの Web サービス仕様に、条件付きでまたは完全に準拠することで、その他の Web サービス実装環境との間で最大限のインターオペラビリティを確保できます。プロファイルは、非専有の一連の Web サービス仕様であり、これらの仕様の説明および改訂も付記されています。これらを総合することにより、Web サービスのさまざまな実装環境間でインターオペラビリティを実現することができます。
- CICS は IBM[®] z/OS XML System Services (XMLSS) パーサーを使用してアプリケーションのバイナリー・データと XML との間の変換を行います。このパーサーは 2GB 境界より上のストレージを使用するため、ユーザー・プログラムが使用できる 2 GB 境界より下のストレージが増え、パフォーマンスが向上します。XMLSS パーサーでは、XML 構文解析の zSeries[®] Application Assist Processor (zAAP) へのオフロードを可能にすることにより、CPU 時間が解放されて、トランザクションのコストが削減されます。zAAP について詳しくは、IBM Redbook の「*zSeries Application Assist Processor (zAAP) Implementation*」(<http://www.redbooks.ibm.com/abstracts/sg246386.html>) を参照してください。

- Web Services Atomic Transaction (WS-AT) では、その SOAP ヘッダーに Web Services Addressing (WS-Addressing) エLEMENTが使用されます。これら WS-Addressing エLEMENTのデフォルトの名前空間の接頭部は、wsa から cicswsa に変更されました。

Web サービスとは

Web サービスとは、ネットワークを介して相互に運用可能なマシン間の対話をサポートする目的で設計されたソフトウェア・システムのことです。Web サービスは、マシン処理可能な形式 (特に、Web サービス記述言語 (WSDL)) で記述されているインターフェースを持ちます。

Web サービスでは、1 つまたは一連の特定のタスクが実行されます。Web サービスは、XML のサービス記述と呼ばれる、標準的で正式な XML の概念を使用して記述されます。このサービス記述は、メッセージ・フォーマット (操作の詳細を記述)、トランスポート・プロトコル、場所など、サービスとの対話に必要な詳細をすべて提供します。

インターフェースの性質上、Web サービスの実装詳細は表示されません。これにより、Web サービス実装先のハードウェアまたはソフトウェアのプラットフォームや、記述に使用したプログラム言語とは独立して使用できるようになります。

この結果、Web サービス・ベースのアプリケーションによる、疎結合状態でコンポーネント指向のテクノロジー相互実装が可能になり、促進されます。Web サービスは、複雑な集計またはビジネス・トランザクションを実行するために、単独または他の Web サービスと組み合わせて使用できます。

Web サービスがビジネスに役立つ仕組み

Web サービスは、ワールド・ワイド・ウェブ (WWW) を介してビジネス機能の配置、およびビジネス機能へのアクセスを提供するテクノロジーです。Web サービスを利用することにより、アプリケーションの統合をかつてないほど迅速、容易、かつ低コストで実現できます。

Web サービスは、以下の点でビジネスに役立つことができます。

- ビジネス実施コストの削減
- ソリューション配置の高速化
- 新規機会の開拓

これらすべてを達成するための鍵は、HTTP、XML、SOAP、WSDL などの既存および先進の標準を基に作成される共通のプログラム間通信モデルです。

CICS が Web サービスをサポートすることにより、再プログラミングの労力を最小限に抑えながら、既存のアプリケーションを新しい方法で配置することが可能になります。

Web サービス用語

Extensible Markup Language (XML)

文書マークアップの標準の 1 つで、単純で人間が読めるタグでデータをマークアップするための汎用構文。この標準は World Wide Web Consortium (W3C) (<http://www.w3.org>) によって承認される。

最初の SOAP 送信側

SOAP メッセージ・パスの開始点で SOAP メッセージを発信する SOAP 送信側

サービス・プロバイダー

Web サービス機能を提供するソフトウェアの集合。

サービス・プロバイダー・アプリケーション

サービス・プロバイダーで使用されるアプリケーションのこと。通常、サービス・プロバイダー・アプリケーションは、サービス・プロバイダーのビジネス・ロジック・コンポーネントを提供する。

サービス・リクエスター

サービス・プロバイダーから Web サービスを要求する役割を持つソフトウェアの集合。

サービス・リクエスター・アプリケーション

サービス・リクエスターで使用されるアプリケーション。通常、サービス・リクエスター・アプリケーションは、サービス・リクエスターのビジネス・ロジック・コンポーネントを提供する。

Simple Object Access Protocol

SOAP を参照。

SOAP 以前は、*Simple Object Access Protocol* の頭字語。非集中の分散環境で情報を交換するための単純なプロトコル。これは、次の 3 つの部分から構成される XML ベースのプロトコルである。

- メッセージの内容およびその処理方法を記述するためのフレームワークを定義するエンベロープ。
- アプリケーション定義のデータ・タイプのインスタンスを表現するための一連のエンコード規則。
- リモート・プロシージャ・コールおよび応答を表現するための規則。

SOAP は、HTTP などの他のプロトコルと併用できる。

SOAP 1.1 の仕様は、<http://www.w3.org/TR/SOAP> で公開される。

SOAP 1.2 の仕様は以下で公開される。

<http://www.w3.org/TR/soap12-part0>

<http://www.w3.org/TR/soap12-part1>

<http://www.w3.org/TR/soap12-part2>

SOAP 中間ノード

SOAP 受信側と SOAP 送信側の両方の機能を備え、SOAP メッセージの内部から宛先を指定できる SOAP ノード。SOAP 中間ノードは、自身を宛先とする SOAP ヘッダー・ブロックを処理し、最終の SOAP 受信側に向けて SOAP メッセージを転送する役割を果たす。

SOAP メッセージ・パス

1 つの SOAP メッセージが通過する一連の SOAP ノードのこと。これには、最初の SOAP 送信側、SOAP 中間ノード (存在しないか 1 つ以上)、最終の SOAP 受信側が含まれる。

SOAP ノード

SOAP メッセージ上で動作する処理ロジック。

SOAP 受信側

SOAP メッセージを受信する SOAP ノード。

SOAP 送信側

SOAP メッセージを送信する SOAP ノード。

最終の SOAP 受信側

SOAP メッセージの最終の宛先となる SOAP 受信側のこと。SOAP 本体およびこれを宛先とするすべての SOAP ヘッダー・ブロックの内容を処理する役割を果たす。

UDDI Universal Description, Discovery and Integration**Universal Description, Discovery and Integration**

Universal Description, Discovery and Integration (UDDI) とは、Web サービスに関する Web ベースの分散情報レジストリーの仕様のこと。UDDI は、企業が自社提供の Web サービスに関する情報を登録できる仕様の一連の実装形態でもあり、これによって他の企業はこの企業の情報を検索できる。仕様は OASIS (<http://www.oasis-open.org>) で公開される。

Web サービス

ネットワークを介した相互に運用可能なマシン間の対話をサポートする目的で設計されたソフトウェア・システムのこと。マシン処理可能な形式 (特に、Web サービス記述言語 (WSDL)) で記述されているインターフェースがある。

Web Services Addressing

Web Services Addressing (WS-Addressing) は、Web サービスおよびメッセージをアドレッシングするための、トランスポートに依存しないメカニズムを提供する。

WS-Addressing の仕様は、以下で公開される。

- <http://www.w3.org/TR/ws-addr-core/>
- <http://www.w3.org/TR/ws-addr-soap/>
- <http://www.w3.org/TR/ws-addr-metadata/>
- <http://www.w3.org/Submission/ws-addressing/>

Web Services Atomic Transaction

全て実行されるか、全く実行されない (all or nothing) プロパティを持つアクティビティの調整に使用される、アトミック・トランザクション調整タイプの定義を提供する仕様。

仕様は <http://www.ibm.com/developerworks/library/specification/ws-tx/#atom> で公開される。

Web サービス・バインディング・ファイル

WEBSERVICE リソースと関連付けられているファイルで、さらに入出力メ

ッセージとアプリケーション・データ構造との間でデータをマップするために CICS が使用する情報が格納されているファイルのこと。

Web サービス記述

サービス・プロバイダーが Web サービスをサービス・リクエスターに呼び出すために仕様をやり取りする場合の手段となる XML 文書。Web サービス記述は、Web サービス記述言語 (WSDL) で記述される。

Web サービス記述言語

Web サービスを記述するための XML アプリケーション。サービスによって提供された抽象機能の記述と、この機能が提供される仕組みや条件など、サービスの具体的な詳細とを分離する目的で設計された。

仕様は <http://www.w3.org/TR/wsdl> で公開される。

Web Services Security

メッセージの保全性と機密性を提供する SOAP メッセージの一連の機能強化。仕様は OASIS (<http://www.oasis-open.org>) によって <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf> で公開される。

WS-Atomic Transaction

Web Services Atomic Transaction

WS-I Basic Profile

非専有の一連の Web サービス仕様であり、これらの仕様の説明および改訂も付記されている。これらを総合することにより、Web サービスのさまざまな実装環境間でインターオペラビリティを実現できる。このプロファイルは Web Services Interoperability Organization (WS-I) によって定義されており、バージョン 1.0 は <http://www.ws-i.org/Profiles/BasicProfile-1.0.html> で入手できる。

WSDL Web サービス記述言語。

WSS Web Services Security

XML Extensible Markup Language。

XML の仕様は以下で公開される。

<http://www.w3.org/TR/soap12-part0>

<http://www.w3.org/TR/soap12-part1>

<http://www.w3.org/TR/soap12-part2>

XML ネーム・スペース

URI 参照によって識別される一まとまりの名前で、エレメント・タイプおよび属性名として XML 文書内で使用される。

XML スキーマ

構造を記述し、他の XML 文書の内容を制約する XML 文書。

XML スキーマ定義言語

XML スキーマを記述するための XML 構文。World Wide Web Consortium (W3C) によって推奨されている。

第 2 章 Web サービスのアーキテクチャー

Web サービスのアーキテクチャーは、サービス・プロバイダー、サービス・リクエスター、およびオプションのサービス・レジストリーの 3 つのコンポーネント間の対話が基本になっています。

サービス・プロバイダー

Web サービス機能を提供するソフトウェアの集合。 Web サービス・プロバイダーには以下のものが含まれています。

- アプリケーション・プログラム
- ミドルウェア
- これらが稼働するプラットフォーム

サービス・リクエスター

サービス・プロバイダーから Web サービスを要求する役割を持つソフトウェアの集合。 Web サービス・リクエスターには以下のものが含まれています。

- アプリケーション・プログラム
- ミドルウェア
- これらが稼働するプラットフォーム

サービス・レジストリー

サービス・レジストリーは、サービス・プロバイダーがサービス記述を発行でき、サービス・リクエスターがこれらのサービス記述を検出できる中心的な場所です。

サービス・リクエスターとプロバイダーがこのレジストリーなしで通信できる状況が多数存在するため、このレジストリーは Web サービス・アーキテクチャーのオプション・コンポーネントです。例えば、サービスを提供する組織はさまざまな方法でサービス記述をサービスのユーザーに直接配布することができます (例えば FTP サイトからのダウンロードとしてサービスを提供できます)。

サービス・レジストリーを使用することには、リクエスターとプロバイダーの両方にとってさまざまな利点があります。例えば、IBM WebSphere® Service Registry and Repository (WSRR) を使用すると、リクエスターはサービスをより速く見つけることができ、プロバイダーは提供されるサービスのバージョン管理を容易に行うことができます。

CICS は、サービス・リクエスター・コンポーネントとサービス・プロバイダー・コンポーネントを実装するために直接サポートします。ただし、サービス・レジストリーを CICS に配置するには、追加のソフトウェアが必要になります。IBM WebSphere Service Registry and Repository (WSRR) を使用する場合、CICS は Web サービス・アシスタントを介して WSRR をサポートします。または、別のプラットフォームにサービス・レジストリーを配置することもできます。

サービス・プロバイダー、サービス・リクエスター、およびサービス・レジストリー間の対話

サービス・プロバイダー、サービス・リクエスター、およびサービス・レジストリー間の対話では、次のような動作が行われます。

Publish

サービス・レジストリーを使用する際、サービス・プロバイダーは自分のサービス記述をサービス・レジストリーに発行 (publish) して、サービス・リクエスターがそれを見つけられるようにします。

Find サービス・レジストリーを使用する際、サービス・リクエスターはレジストリー内にあるサービス記述を見つけます。

Bind サービス・リクエスターはサービス記述を使用してサービス・プロバイダーをバインドし、Web サービスのインプリメンテーションと対話します。

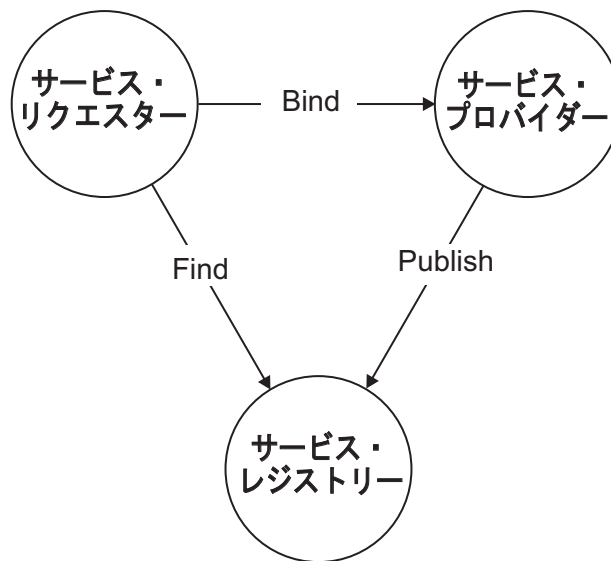


図 1. Web サービスのコンポーネントおよび対話

Web サービス記述

Web サービス記述とは、サービス・プロバイダーがサービス・リクエスターに対して Web サービスを呼び出すための仕様を伝達する基準となる文書です。Web サービス記述は、Web サービス記述言語 (WSDL) と呼ばれる XML アプリケーションで表現されます。

Web サービス記述では、サービス・プロバイダーとサービス・リクエスターとの通信を確保するために必要な共有知識やカスタマイズ・プログラミングの労力を最小限に抑えられるように Web サービスが記述されます。例えば、サービス・プロバ

イダーとサービス・リクエスターは、相手側で稼働しているプラットフォームを認識する必要はなく、相手側で作成されているプログラム言語を認識する必要ありません。

サービス記述は WSDL 1.1 または WSDL 2.0 のいずれかの仕様に適合でき、用語と主要なエレメントの両方に、サービス記述に含めることができる相違点があります。以下の情報は、サービス記述の目的を説明するために、WSDL 1.1 の用語とエレメントを使用します。

WSDL の構造により、サービス記述は次のように区分されます。

- 抽象的なサービス・インターフェース定義。サービスのインターフェースを記述し、サービスの実装と呼び出しを行うプログラムを作成可能にします。
- 具体的なサービス実装定義。プロバイダーの Web サービスのネットワーク (または エンドポイント) の場所および実装に関するその他の具体的な詳細を記述し、サービス・リクエスターからサービス・プロバイダーへの接続を可能にします。

このことは、10 ページの図 2 に図示されています。

WSDL 1.1 文書には、ネットワーク・サービスの定義に以下の主要なエレメントが使用されます。

<types>

一定の型体系 (XML スキーマなど) を使用したデータ・タイプ定義のコンテナ。メッセージ内で使用されるデータ・タイプを定義します。すべてのメッセージが単純なデータ・タイプから構成される場合は、<types> エレメントは必要ありません。

<message>

操作の入力パラメーターおよび出力パラメーターを定義するために使用する XML データ・タイプを指定します。

<portType>

1 つ以上のエンドポイントにサポートされている一連の操作を定義します。<portType> エレメントの内部では、各操作は <operation> エレメントで記述されます。

<operation>

入出力データ・フローで表示できる XML メッセージを指定します。操作は、プログラム言語のメソッド・シグニチャーに相当します。

<binding>

特定の <portType> エレメントに関するプロトコル、データ・フォーマット、セキュリティおよびその他の属性を記述します。

<port> エンドポイントのネットワーク・アドレスを指定し、これを <binding> エレメントに関連付けます。

<service>

Web サービスを一まとまりの関連エンドポイントとして定義します。<service> エレメントには、1 つ以上の <port> エレメントが格納されています。

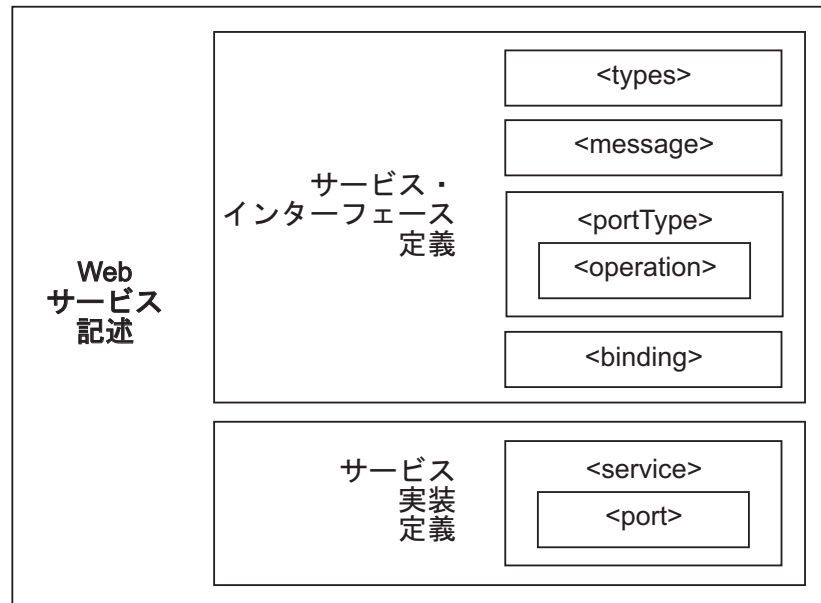


図2. Web サービス記述の構造

Web サービス記述を区分する機能により、完全なサービス記述を作成する責務を分割できます。図のように、業界全体にわたって使用するため標準制定機関によって定義され、その業界内の個々の企業によって実装されるサービスについて考えます。

- 標準制定機関は、以下のエレメントを含むサービス・インターフェース定義を規定します。

```

<types>
<message>
<portType>
<binding>

```

- サービスの実装を提案するサービス・プロバイダーは、以下のエレメントを含むサービス実装定義を規定します。

```

<port>
<service>

```

サービスの発行

サービスの記述は、多数の異なる機構を使用して発行できます。それぞれの機構には異なる機能があり、さまざまな状況での使用に適しています。CICS では、IBM WebSphere Service Registry and Repository (WSRR) を使用してサービス記述を発行することが可能です。または、他の方式を使用してサービス記述を発行することもできます。

WSSR CICS では WSRR を使用してサービス記述を発行することができます。

CICS での WSSR サポートの詳細については、インフォメーション・センターのトピック『Web サービス・アシスタントと WSRR の相互運用性』を参照してください。

以下の機構はいずれも CICS では直接サポートされていませんが、サービス記述を発行するために CICS で使用することが可能です。

直接の発行

これは、サービス記述を発行するための最も単純な機構です。サービス・プロバイダーは E メール添付ファイル、FTP サイト、または CD ROM 配布を使用して、サービス記述をサービス・リクエスターに直接送信します。

DISCO

これらの専有プロトコル群は、動的な発行機能を提供します。サービス・リクエスターは、サービス・プロバイダーによって指定され、URL で識別されるネットワークの場所から Web サービス記述を検索するために、単純な HTTP GET 機構を使用します。

Universal Description, Discovery and Integration (UDDI)

Web サービスに関する Web ベースの分散情報レジストリーの仕様。

UDDI は、企業が自社提供の Web サービスに関する情報を登録できる仕様の一連の実装形態でもあり、これによって他の企業はこの企業の情報を検索できます。

必要に応じて、複数の形式でサービス記述を発行することができます。

第 3 章 SOAP とは

SOAP とは、分散環境で情報を交換するためのプロトコルのことです。SOAP メッセージは XML 文書としてエンコードされ、さまざまな下位プロトコルを使用して交換できます。

以前は *Simple Object Access Protocol* の頭字語であった SOAP は、World Wide Web Consortium (W3C) で開発され、W3C が発行した以下の文書で定義されています。SOAP に関して信頼できる詳細情報が必要な場合は、以下の資料を参照してください。

Simple Object Access Protocol (SOAP) 1.1 (W3C のメモ)

SOAP Version 1.2 Part 0: Primer (W3C 勧告)

SOAP Version 1.2 Part 1: Messaging Framework (W3C 勧告)

SOAP Version 1.2 Part 2: Adjuncts (W3C 勧告)

SOAP 仕様は、SOAP メッセージが SOAP ノード 間に渡される分散処理モデルを記述しています。SOAP メッセージは、SOAP 送信側 から発信され、SOAP 受信側 に送信されます。送信側と受信側の間では、メッセージは 1 つ以上の SOAP 中間ノード によって処理される場合があります。

SOAP メッセージは、SOAP ノード間、つまり SOAP 送信側から SOAP 受信側への片方向伝送ですが、メッセージを組み合わせると、要求と応答、対等の会話などのより複雑な対話を構成できます。

仕様には、以下の内容も記述されています。

- アプリケーション定義のデータ・タイプのインスタンスを表現するための一連のエンコード規則。
- リモート・プロシージャ・コールおよび応答を表現するための規則。

SOAP メッセージの構造

SOAP メッセージは、<Envelope> エレメントから構成される XML 文書としてエンコードされます。このエレメントには、オプションの <Header> エレメントと必須の <Body> エレメントが格納されています。<Body> の内部に格納されている <Fault> エレメントは、エラー報告の用途で使用されます。

SOAP エンベロープ

SOAP <Envelope> は、各 SOAP メッセージのルート・エレメントであり、ここには、オプションの <Header> と必須の <Body> という 2 つの子エレメントが格納されています。

SOAP ヘッダー

SOAP <Header> は、SOAP エンベロープのオプションのサブエレメントであり、SOAP ノードがメッセージ・パスに沿って処理する対象のアプリケーション関連情報を渡すときに使用されます。

SOAP 本体

SOAP <Body> は、SOAP エンベロープの必須のサブエレメントで、ここにはメッセージの最終の受信側を目的とした情報が格納されています。

SOAP 障害

SOAP <Fault> は、SOAP 本体のサブエレメントで、エラー報告のために使用されます。

SOAP メッセージの <Body> に格納されている <Fault> エレメントを除くと、<Header> および <Body> 内部の XML エレメントは、これらのエレメントを使用するアプリケーションによって定義されます。ただし、SOAP 仕様のために、エレメントの構造には何らかの制約が課されます。

図 3 には、SOAP メッセージの主要なエレメントを示します。

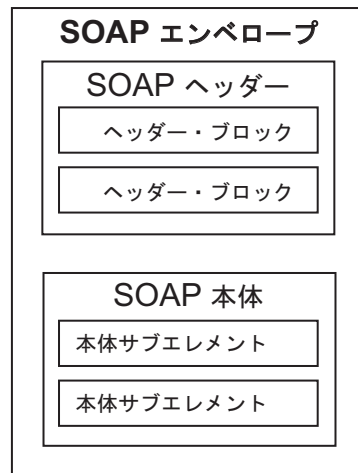


図 3. SOAP メッセージの構造

15 ページの図 4 は、ヘッダー・ブロック (<m:reservation> エレメントと <n:passenger> エレメント) および本体 (<p:itinerary> エレメントと <q:lodging> エレメントを含む) を格納する SOAP メッセージの例です。

```

<?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>

```

図 4. SOAP 1.2 メッセージの例

SOAP ヘッダー

SOAP <Header> は、SOAP メッセージ内部にあるオプションの要素です。この要素は、SOAP ノードがメッセージ・パスに沿って処理する対象のアプリケーション関連情報を渡すときに使用されます。

<Header> 要素の直接の子要素は、ヘッダー・ブロックと呼ばれます。ヘッダー・ブロックは、アプリケーション定義の XML 要素で、送信側から最後の受信側までのメッセージのパスで検出される可能性がある SOAP ノードを先にできるデータの論理グループを表します。

SOAP ヘッダー・ブロックは、SOAP 中間ノードと最終の SOAP 受信側ノードで処理できますが、実際のアプリケーションでは、すべてのヘッダー・ブロックが各ノードで処理されるわけではありません。むしろ、個々のノードは、通常、特定のヘッダー・ブロックを処理するように設計されています。逆に言えば、個々のヘッダー・ブロックが特定のノードによって処理されるようになっています。

SOAP ヘッダーを使用すると、通信相手との間で事前に合意を得る必要なく、機能を非集中方式で SOAP メッセージに追加できます。 SOAP では、誰が機能に対応するか、およびその機能はオプションか必須かを示すために使用できる属性がいくつか定義されます。こうした「管理」情報には、メッセージの処理に関連した指示またはコンテキスト情報の受け渡しなどがあります。これにより、SOAP メッセージをアプリケーション固有の方式で拡張できます。

ヘッダー・ブロックはアプリケーション定義ですが、ヘッダー・ブロックに存在する SOAP 定義の属性は、SOAP ノードによるヘッダー・ブロックの処理方法を示しています。いくつかの重要な属性を以下に示します。

encodingStyle

SOAP メッセージの一部をエンコードするために使用する規則を示します。

SOAP では、XML が許可する非常に柔軟なエンコード方式よりも限られた、一連のデータのエンコード規則を定義します。

role (SOAP 1.2)

actor (SOAP 1.1)

SOAP 1.2 では、あるメッセージに対して特定のノードが稼働するかどうかは **role** 属性によって指定されます。特定のノードに指定された役割が、ヘッダー・ブロックの **role** 属性に一致した場合、このノードはヘッダーを処理します。役割が一致しない場合は、ヘッダー・ブロックを処理しません。 SOAP 1.1 では、**actor** 属性が同じ機能を実行します。

役割はアプリケーションによって定義され、URI で指定されます。例えば、`http://example.com/Log` は、ロギングを実行するノードの役割を指定しています。このノードに処理されるヘッダー・ブロックは、`env:role="http://example.com/Log"` を指定します (ここで、ネーム・スペースの接頭部 `env` は、`http://www.w3.org/2003/05/soap-envelope` の SOAP ネーム・スペース名に関連付けられます)。

SOAP 1.2 仕様では、アプリケーションによって定義されている役割のほかに、以下の 3 つの標準的な役割が定義されています。

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

メッセージ・パス上の SOAP ノードがヘッダー・ブロックを直接処理することはありません。この役割を持つヘッダー・ブロックを使用すると、その他の SOAP ヘッダー・ブロックの処理に必要なデータを搬送できます。

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

メッセージ・パス上にあるすべての SOAP ノードは、ヘッダー・ブロックを検査すると見込まれています (メッセージ・パスの前の方にあるノードによってヘッダーが削除されていないことが前提です)。

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

最終の受信側ノードのみがヘッダー・ブロックを検査すると見込まれています。

mustUnderstand

この属性は、SOAP ノードがアプリケーション全体の目的に重要なヘッダー・ブロックを無視しないようにするために使用します。 SOAP ノードが (**role** 属性または **actor** 属性を使用して) ヘッダー・ブロックを処理することを確認し、かつ **mustUnderstand** 属性の値が `"true"` である場合、この SOAP ノード

はその仕様に整合する方法でヘッダー・ブロックを処理するか、またはまったく処理しない（で障害を通知する）必要があります。ただし、属性の値が "false" である場合は、このノードがヘッダー・ブロックを処理する必要はありません。

mustUnderstand 属性は、実際にはヘッダー・ブロックの処理が必須かオプションかを示しています。

mustUnderstand 属性の値は次のとおりです。

true (SOAP 1.2)

1 (SOAP 1.1)

ノードは、仕様に整合する方法でヘッダー・ブロックを処理するか、またはまったく処理しない（で障害を通知する）必要があります。

false (SOAP 1.2)

0 (SOAP 1.1)

ノードがヘッダー・ブロックを処理する必要はありません。

relay (SOAP 1.2 のみ)

SOAP 中間ノードは、ヘッダー・ブロックを処理すると、SOAP メッセージからヘッダー・ブロックを削除します。デフォルトでは、無視したすべてのヘッダー・ブロックを削除します（これは、**mustUnderstand** 属性の値が "false" であったためです）。ただし、**relay** 属性が "true" という値で指定されている場合、ノードはメッセージ内のヘッダー・ブロックを未処理のまま保存します。

SOAP 本体

<Body> は、SOAP メッセージで伝達される主な終端間情報の搬送媒体となる SOAP エンベロープ内部にある必須エレメントです。

<Body> エレメントとその関連の子エレメントは、最初の SOAP 送信側と最後の SOAP 受信側との間で情報を交換するために使用されます。SOAP では、<Body> に対して 1 つの子エレメント <Fault> を定義します。このエレメントは、エラーを報告するために使用されます。<Body> 内部のその他のエレメントは、それらを使用する Web サービスによって定義されます。

SOAP 障害

SOAP <Fault> エレメントは、SOAP メッセージ内部のエラー情報および状況情報を伝達するときに使用されます。

SOAP <Fault> エレメントは、存在する場合、本文の項目として存在する必要があり、Body エレメント内部に複数存在することはできません。SOAP <Fault> エレメントのサブエレメントは、SOAP 1.1 と SOAP 1.2 とで異なります。

SOAP 1.1

SOAP 1.1 では、SOAP <Fault> エレメントに以下のサブエレメントが格納されています。

<faultcode>

<faultcode> エレメントは、<Fault> エレメント内部の必須エレメントの 1 つです。このエレメントは、ソフトウェアが処理できる形式で障害に関する

情報を提供します。 SOAP は、基本的な SOAP 障害を網羅する SOAP 障害コードの小セットを定義します。このセットはアプリケーションによって拡張できます。

<faultstring>

<faultstring> エLEMENTは、<Fault> ELEMENT内部の必須ELEMENTの 1 つです。このELEMENTは、人間の読み手を対象とする形式で障害に関する情報を提供します。

<faultactor>

<faultactor> ELEMENTには、障害を生成した SOAP ノードの URI が格納されています。最後の SOAP 受信側以外の SOAP ノードは、障害コードの生成時に <faultactor> ELEMENTを包含する必要があります。最後の SOAP 受信側はこのELEMENTを包含することが必須ではありませんが、包含する場合もあります。

<detail>

<detail> ELEMENTは、<Body> ELEMENTに関連したアプリケーション固有のエラー情報を伝達します。このELEMENTが存在する必要があるのは、<Body> ELEMENTの内容を正常に処理できなかった場合です。ヘッダー項目に属するエラー情報の情報を伝達するためにこのELEMENTを使用することはできません。ヘッダー項目に属する詳細なエラー情報は、ヘッダー項目の内部に格納して搬送する必要があります。

SOAP 1.2

SOAP 1.2 では、SOAP <Fault> ELEMENTに以下のサブELEMENTが格納されています。

<Code> <Code> ELEMENTは、<Fault> ELEMENT内部の必須ELEMENTの 1 つです。このELEMENTは、ソフトウェアが処理できる形式で障害に関する情報を提供します。このELEMENTには、<Value> ELEMENTとオプションの <Subcode> ELEMENTが格納されています。

<Reason>

<Reason> ELEMENTは、<Fault> ELEMENT内部の必須ELEMENTの 1 つです。このELEMENTは、人間の読み手を対象とする形式で障害に関する情報を提供します。 <Reason> ELEMENTには、1 つ以上の <Text> ELEMENTが格納されています。このELEMENTのそれぞれには、障害に関する情報がさまざまな言語で記述されています。

<Node> <Node> ELEMENTには、障害を生成した SOAP ノードの URI が格納されています。最後の SOAP 受信側以外の SOAP ノードは、障害コードの生成時に <Node> ELEMENTを包含する必要があります。最後の SOAP 受信側はこのELEMENTを包含することが必須ではありませんが、包含する場合もあります。

<Role> <Role> ELEMENTには、障害が発生した箇所でノードが稼働していた役割を識別する URI が格納されています。

<Detail>

<Detail> ELEMENTは、オプションのELEMENTで、障害を記述する SOAP 障害コードに関連したアプリケーション固有のエラー情報が格納され

ています。<Detail> エレメントの存在は、障害のある SOAP メッセージのどの部分が処理されたかに関しては意味がありません。

SOAP ノード

SOAP ノードとは、SOAP メッセージ上で動作する処理ロジックのことです。

SOAP ノードが可能な処理は次のとおりです。

- SOAP メッセージの送信
- SOAP メッセージの受信
- SOAP メッセージの処理
- SOAP メッセージの中継

SOAP ノードが可能な役割は次のとおりです。

SOAP 送信側

SOAP メッセージを送信する SOAP ノード。

SOAP 受信側

SOAP メッセージを受信する SOAP ノード。

最初の SOAP 送信側

SOAP メッセージ・パスの開始点で SOAP メッセージを発信する SOAP 送信側

SOAP 中間ノード

SOAP 中間ノードとは、SOAP 受信側と SOAP 送信側の両方の機能を備え、SOAP メッセージの内部から宛先を指定できる SOAP ノードのことです。SOAP 中間ノードは、自身を宛先とする SOAP ヘッダー・ブロックを処理し、最終の SOAP 受信側に向けて SOAP メッセージを転送する役割を果たします。

最終の SOAP 受信側

SOAP メッセージの最終の宛先となる SOAP 受信側のことです。SOAP 本体およびこれを宛先とするすべての SOAP ヘッダー・ブロックの内容を処理する役割を果たします。一部の環境では、SOAP 中間ノードでの問題などの理由により、SOAP メッセージが最終の SOAP 受信側に到達できない場合があります。

SOAP メッセージ・パス

SOAP メッセージ・パスとは、1 つの SOAP メッセージが通過する一連の SOAP ノードのことです。これには、最初の SOAP 送信側、SOAP 中間ノード (存在しないか 1 つ以上)、最終の SOAP 受信側が含まれます。

最も簡単なケースでは、SOAP メッセージは 2 つのノード間で送信されます。つまり SOAP 送信側 から SOAP 受信側 までです。しかし、より複雑なケースでは、メッセージは SOAP 中間ノード によって処理されます。このノードでは、SOAP メッセージが受信され、さらに次のノードへ送信されます。20 ページの図 5 に、SOAP メッセージ・パスの例を示します。ここでは、SOAP メッセージが最初の SOAP 送信側ノードから最終の SOAP 受信側ノードに向けて送信され、その経路上

で 2 つの SOAP 中間ノードを通過します。

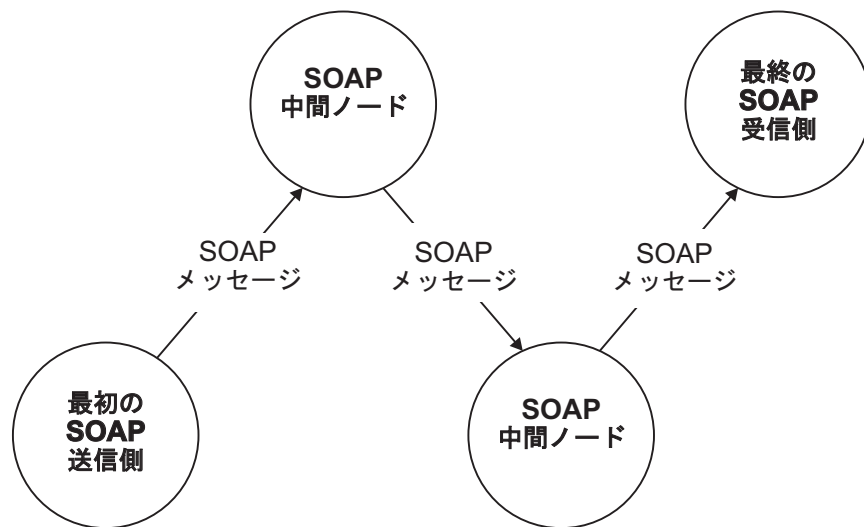


図 5. SOAP メッセージ・パスの例

SOAP 中間ノードは、SOAP 受信側と SOAP 送信側の両方の機能を備えています。SOAP メッセージのヘッダー・ブロックを処理でき、最終の受信側に向かって SOAP メッセージを転送します (ヘッダー・ブロックの処理は必須の場合もあります)。

最終の SOAP 受信側 とは、SOAP メッセージの最終的な宛先です。ヘッダー・ブロックの処理だけでなく、SOAP 本体を処理する役割も果たします。一部の環境では、SOAP 中間ノードでの問題などの理由により、SOAP メッセージが最終の SOAP 受信側に到達できない場合があります。

第 4 章 CICS が Web サービスをサポートする仕組み

CICS は、Web サービス環境で CICS アプリケーションを配置するための 2 つの異なる方法をサポートしています。一方の方法では、プログラミングの労力を最小限に抑えながら迅速な配置を実現できます。もう一方の方法では、各ユーザーの特定の必要性に合わせて記述したコードを使用することにより、Web サービス・アプリケーション全体にわたる柔軟性と制御を実現できます。これら 2 つの方法は、1 つ以上のパイプライン と、Web サービスの要求および応答を対象として動作する 1 つ以上のメッセージ・ハンドラー・プログラムで構成されるインフラストラクチャーによって支えられています。

CICS アプリケーションを Web サービス環境に配置するとき、以下のオプションの中から選ぶことができます。

- CICS Web サービス・アシスタントを使用すると、アプリケーションを配置するために必要なプログラミングの労力を最小限に抑えようとします。

例えば、既存のアプリケーションを Web サービスとして公開する場合は、高水準言語のデータ構造から着手して、Web サービス記述を生成することができます。もう 1 つの方法として、既存の Web サービスと通信する場合は、Web サービス記述から着手し、作成するプログラムに使用可能な高水準言語の構造を生成することができます。

CICS Web サービス・アシスタントは、アプリケーションを配置するために必要な CICS リソースも生成します。さらに、アプリケーションを実行すると、CICS は、出力ではアプリケーション・データを SOAP メッセージに変換し、入力では SOAP メッセージをアプリケーション・データに戻します。

- データの処理を完全に制御するには、独自のコードを記述して、アプリケーション・データと、サービス・リクエスターとサービス・プロバイダーとの間で流れるメッセージをマップします。

例えば、Web サービス・インフラストラクチャーの範囲内で SOAP 以外のメッセージを使用する場合は、独自のコードを記述して、メッセージ・フォーマットとアプリケーションが使用するフォーマットとを変換します。

どちらの方法を採用する場合でも、独自のメッセージ・ハンドラーを使用して要求メッセージおよび応答メッセージに対する追加の処理を実行できます。または、SOAP メッセージの処理専用で設計された CICS 提供のメッセージ・ハンドラーを使用することもできます。

メッセージ・ハンドラーおよびパイプライン

メッセージ・ハンドラー とは、Web サービスの要求および応答について独自の処理を実行できるプログラムのことです。パイプライン とは、順序どおり実行される一連のメッセージ・ハンドラーのことです。

パイプラインの運用には次に示す 2 つの異なる段階があります。

1. 要求段階。CICS がパイプライン内の各ハンドラーを次々と呼び出す段階です。各メッセージ・ハンドラーは、制御を CICS に戻す前に要求を処理できます。
2. この後に応答段階が続きます。この段階でも、CICS は各ハンドラーを次々と呼び出しますが、順序が逆になります。つまり、要求段階で最初に呼び出されるメッセージ・ハンドラーは、応答段階では最後に呼び出されます。各メッセージ・ハンドラーは、この段階のうちに応答を処理できます。

要求の後に必ずしも応答があるわけではありません。つまり、一部のアプリケーションは、サービス・リクエスターからプロバイダーへの片方向のメッセージ・フローを使用します。この場合、処理すべきメッセージがなくても、応答段階で各ハンドラーが順に呼び出されます。

図 6 には、次の 3 つのメッセージ・ハンドラーのパイプラインを示します。

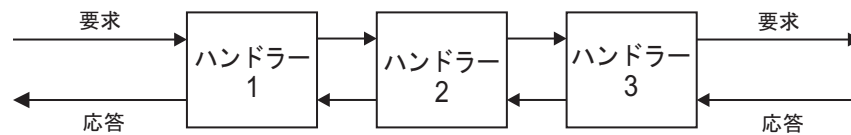


図 6. 一般的な CICS パイプライン

この例では、ハンドラーは次の順序で実行されます。

要求段階の場合

1. ハンドラー 1
2. ハンドラー 2
3. ハンドラー 3

応答段階の場合

1. ハンドラー 3
2. ハンドラー 2
3. ハンドラー 1

サービス・プロバイダーの場合、段階間の移行は、通常、要求を吸収するパイプラインの最後のハンドラー（端末ハンドラー）で実行され、応答が生成されます。サービス・リクエスターの場合、移行が実行されるのは、要求がサービス・プロバイダーで処理されるときです。ただし、要求段階のメッセージ・ハンドラーは、応答段階への即時の移行を強制できます。また、CICS によってエラーが検出された場合にも、即時の移行を実行することができます。

メッセージ・ハンドラーは、メッセージを変更することも、変更せずにそのままの状態にしておくこともできます。例を次に示します。

- 暗号化と復号を実行するメッセージ・ハンドラーは、暗号化されたメッセージを入力で受け取り、復号されたメッセージを次のハンドラーに渡します。出力では、逆の処理が行われます。つまり、非暗号化テキスト・メッセージを受け取り、暗号化されたメッセージを次のハンドラーに渡します。

- ロギングを実行するメッセージ・ハンドラーは、メッセージを調べ、関係のある情報をこのメッセージからログにコピーします。次のハンドラーに渡されるメッセージは変更されません。

重要: CICS TS の SOAP 機能に精通している場合は、このリリースの CICS でのパイプラインの構造が SOAP 機能に使用されているものと同じではないことに留意してください。

トランスポート関連ハンドラー

CICS は、Web サービス・リクエスターとプロバイダー間での 2 つのトランスポート機構の使用をサポートしています。使用中のトランスポート機構がどちらであるかによっては、異なるメッセージ・ハンドラーを呼び出すことが必要な場合があります。例えば、HTTP トランスポートを使用して外部ネットワークと通信する場合には、メッセージの一部を暗号化するメッセージ・ハンドラーが必要になります。しかし、機密保護機能のある内部ネットワークで MQ トランスポートを使用する場合、暗号化は必要ありません。

これをサポートするため、パイプラインを構成することにより、特定のトランスポート (HTTP または MQ) が使用中の場合にのみ呼び出されるハンドラーを指定できます。サービス・プロバイダーの場合は、さらに具体的に、特定の指定リソース (HTTP トランスポートの場合は TCPIPSERVICE、MQ トランスポートの場合は QUEUE) が使用中の場合にのみ呼び出されるハンドラーを指定できます。

このことは、図 7 に図示されています。

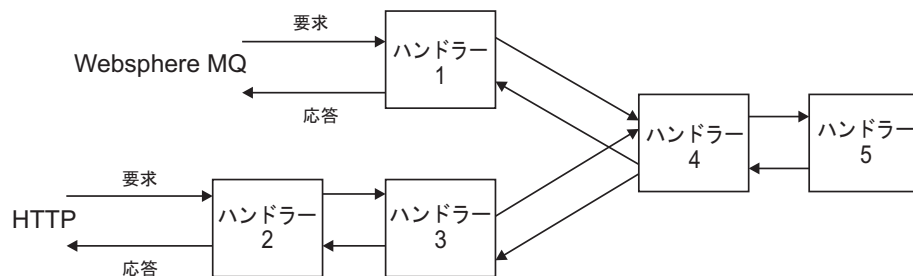


図 7. トランスポート関連ハンドラーを持つパイプライン

この例では、サービス・プロバイダーに適用されるものは次のとおりです。

- ハンドラー 1 は、MQ トランスポートを使用するメッセージの場合に呼び出されます。
- ハンドラー 2 および 3 は、HTTP トランスポートを使用するメッセージの場合に呼び出されます。
- ハンドラー 4 および 5 は、すべてのメッセージを対象として呼び出されます。
- ハンドラー 5 は、端末ハンドラーです。

フローの中断

要求の処理時に、メッセージ・ハンドラーはメッセージを次のハンドラーに渡さない判断をする場合がありますが、応答を生成する場合もあります。メッセージの通常の処理は中断され、パイプライン内の一部のハンドラーは呼び出されません。例えば、図 8 のハンドラー 2 は、セキュリティー検査を実行する役割を持っています。

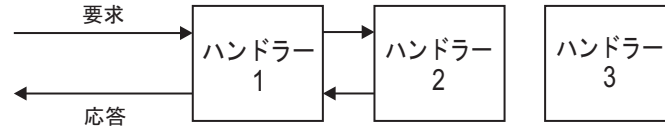


図 8. パイプライン・フローの中断

要求に正しいセキュリティー・クリデンシャルがない場合、ハンドラー 2 は、(要求をハンドラー 3 に渡さずに) 要求を抑止し、適切な応答を作成します。パイプラインは応答段階になっているため、ハンドラー 2 が制御を CICS に戻すと、呼び出される次のハンドラーはハンドラー 1 となり、ハンドラー 3 は完全にバイパスされます。

通常のメッセージ・フローをこのように中断するハンドラーは、メッセージの発信元が応答を期待する場合にのみ、中断する必要があります。例えば、アプリケーションがサービス・リクエスターからプロバイダーへの片方向のメッセージ・フローを使用する場合、ハンドラーは応答を生成してはいけません。

サービス・プロバイダー・パイプライン

サービス・プロバイダー・パイプラインでは、CICS が要求を受け取ります。この要求はパイプラインを介してターゲット・アプリケーション・プログラムに渡されます。アプリケーションからの応答は、同じパイプラインを介してサービス・リクエスターに戻されます。

CICS がサービス・プロバイダーの役割を果たす場合、CICS は以下の操作を実行します。

1. サービス・リクエスターからの要求を受け取る。
2. 要求を調べ、ターゲット・アプリケーション・プログラムに関係のある内容を抽出する。
3. アプリケーション・プログラムを呼び出し、要求から抽出したデータを渡す。
4. アプリケーション・プログラムが制御を戻したら、アプリケーション・プログラムから戻されたデータを使用して応答を作成する。
5. サービス・リクエスターへ応答を送信する。

25 ページの図 9 には、サービス・プロバイダー設定における次の 3 つのメッセージ・ハンドラーのパイプラインを示します。

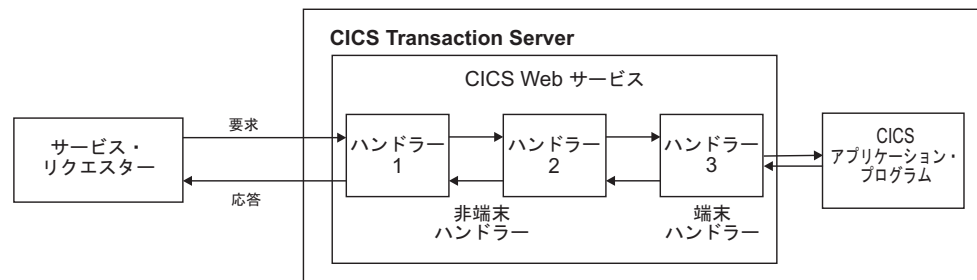


図9. サービス・プロバイダー・パイプライン

1. CICS は、サービス・リクエスターから要求を受け取ります。次に、この要求をメッセージ・ハンドラー 1 に渡します。
2. メッセージ・ハンドラー 1 は何らかの処理を実行して、要求をハンドラー 2 に渡します。(正確には、パイプラインを管理する CICS に制御を戻します。次に CICS は次のメッセージ・ハンドラーに制御を渡します。)
3. メッセージ・ハンドラー 2 はハンドラー 1 から要求を受け取り、何らかの処理を実行して、要求をハンドラー 3 に渡します。
4. メッセージ・ハンドラー 3 は、このパイプラインの端末ハンドラーです。ハンドラー 3 は、要求に記載された情報を使用して、アプリケーション・プログラムを呼び出します。次に、アプリケーション・プログラムからの出力を使用して応答を生成し、この応答をハンドラー 2 に戻します。
5. メッセージ・ハンドラー 2 はハンドラー 3 から応答を受け取り、何らかの処理を実行して、応答をハンドラー 1 に渡します。
6. メッセージ・ハンドラー 1 はハンドラー 2 から応答を受け取り、何らかの処理を実行して、応答をサービス・リクエスターに戻します。

サービス・リクエスター・パイプライン

サービス・リクエスター・パイプラインでは、アプリケーション・プログラムが要求を作成します。この要求はパイプラインを介してサービス・プロバイダーに渡されます。サービス・プロバイダーからの応答は、同じパイプラインを介してアプリケーション・プログラムに戻されます。

CICS がサービス・リクエスターの役割を果たす場合、CICS は以下の操作を実行します。

1. アプリケーション・プログラムから得られたデータを使用して、要求を作成する。
2. 要求をサービス・プロバイダーに送信する。
3. サービス・プロバイダーから応答を受信する。
4. 応答を調べ、オリジナル・アプリケーション・プログラムに関係のある内容を抽出する。
5. アプリケーション・プログラムに制御を戻す。

図 10 には、サービス・リクエスターの設定における次の 3 つのメッセージ・ハンドラーのパイプラインを示します。

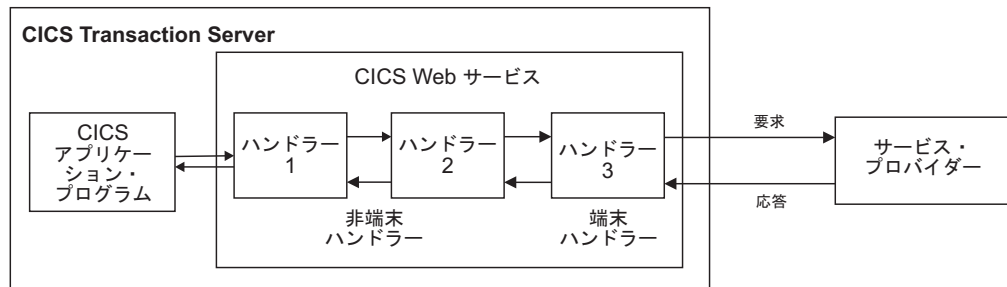


図 10. サービス・リクエスター・パイプライン

1. アプリケーション・プログラムが要求を作成します。
2. メッセージ・ハンドラー 1 は、アプリケーション・プログラムから要求を受け取り、何らかの処理を実行して、要求を手動でハンドラー 2 に渡します。(正確には、パイプラインを管理する CICS に制御を戻します。次に CICS は次のメッセージ・ハンドラーに制御を渡します。)
3. メッセージ・ハンドラー 2 はハンドラー 1 から要求を受け取り、何らかの処理を実行して、要求を手動でハンドラー 3 に渡します。
4. メッセージ・ハンドラー 3 は、ハンドラー 2 から要求を受け取り、何らかの処理を実行して、要求をサービス・プロバイダーに渡します。
5. メッセージ・ハンドラー 3 はサービス・プロバイダーから応答を受け取り、何らかの処理を実行して、応答を手動でハンドラー 2 に渡します。
6. メッセージ・ハンドラー 2 はハンドラー 3 から応答を受け取り、何らかの処理を実行して、応答を手動でハンドラー 1 に渡します。
7. メッセージ・ハンドラー 1 はハンドラー 2 から応答を受け取り、何らかの処理を実行して、応答をアプリケーション・プログラムに戻します。

CICS パイプラインおよび SOAP

Web サービスの要求と応答を処理するために CICS が使用するパイプラインは、各メッセージ・ハンドラーで実行できる処理に関する制約が少ないという点で一般的です。ただし、多くの Web サービス・アプリケーションは SOAP メッセージを使用しており、これらのメッセージを処理する場合は、SOAP 仕様に準拠する必要があります。したがって、CICS には、パイプラインを SOAP ノードとして構成するときに役立つ特殊な SOAP メッセージ・ハンドラー・プログラムが用意されています。

- パイプラインは、サービス・リクエスターまたはサービス・プロバイダーで使用するために、次のように構成できます。
 - サービス・リクエスター・パイプラインは、要求の最初の SOAP 送信側になり、かつ応答の最終の SOAP 受信側になります。
 - サービス・プロバイダー・パイプラインは、要求の最終の SOAP 受信側になり、かつ応答の最初の SOAP 送信側になります。

CICS パイプラインを SOAP 中間ノードとして機能するように構成することはできません。

- サービス・リクエスター・パイプラインは、SOAP 1.1 または SOAP 1.2 をサポートするように構成できますが、両方をサポートするには構成できません。ただし、サービス・プロバイダー・パイプラインは、SOAP 1.1 と SOAP 1.2 の両方をサポートするように構成できます。CICS システム内部には、多数のパイプラインを保持できます。SOAP 1.1 または SOAP 1.2 をサポートするものと、両方をサポートするものがあります。
- CICS パイプラインを構成すると、複数の SOAP メッセージ・ハンドラーを保持できます。
- CICS 提供の SOAP メッセージ・ハンドラーを構成すると、1 つ以上のユーザー作成ヘッダー処理ルーチンを呼び出すことができます。
- CICS 提供の SOAP メッセージ・ハンドラーを構成すると、WS-I Basic Profile バージョン 1.1 に準拠するといういくつかの側面と、SOAP メッセージに特定のヘッダーを存在させることを実現できます。

SOAP メッセージ・ハンドラー、およびそのヘッダー処理ルーチンは、パイプライン構成ファイルで指定します。

SOAP メッセージおよびアプリケーション・データ構造

多くの場合、CICS Web サービス・アシスタントは、アプリケーション・プログラムで使用されている上位データ構造と、SOAP メッセージの <Body> エLEMENTの内容との間でデータを変換するコードを生成できます。これらの場合には、アプリケーション・プログラムを作成するときに、SOAP 本体の解析または構成を行う必要がありません。これらの作業は CICS によって実行されます。

CICS は実行時に、データを変換するためにアプリケーション・データ構造と SOAP メッセージの形式に関する情報を必要とします。この情報は、次の 2 つのファイルに保持されます。

- Web サービスのバインディング・ファイル

このファイルは、ユーティリティー・プログラム DFHLS2WS を使用して、アプリケーション言語のデータ構造を基に CICS Web サービス・アシスタントによって生成されるか、またはユーティリティー・プログラム DFHWS2LS を使用して、Web サービス記述を基に生成されます。CICS はバインディング・ファイルを使用して、Web サービス・アプリケーションが使用するリソースを生成し、アプリケーションのデータ構造と SOAP メッセージ間のマッピングを行います。

- Web サービス記述

これは、既存の Web サービス記述である場合と、ユーティリティー・プログラム DFHLS2WS を使用して、アプリケーション言語のデータ構造を基に生成する場合があります。CICS では、Web サービス記述を使用して、SOAP メッセージの完全な検証を行います。

28 ページの図 11 に、これらのファイルがサービス・プロバイダーで使用される様子を示します。

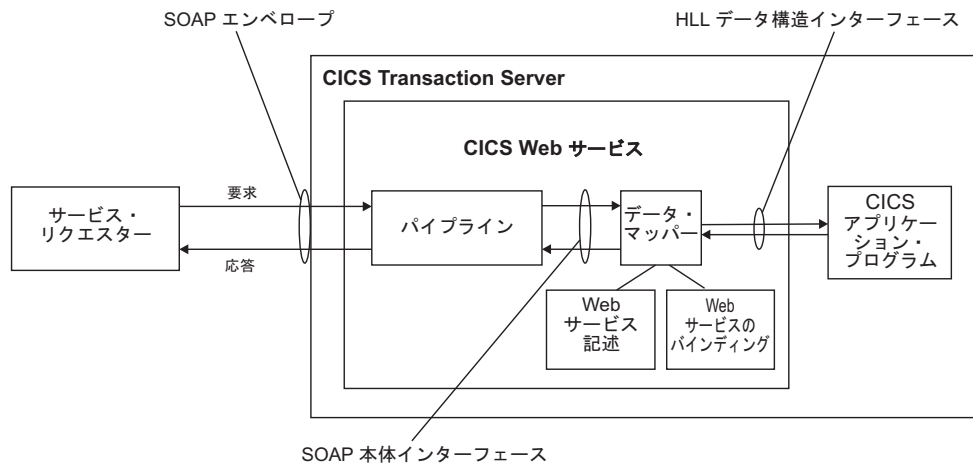


図 11. サービス・プロバイダーでの SOAP 本体からアプリケーション・データ構造へのマッピング

パイプラインのメッセージ・ハンドラー (一般に、CICS 提供の SOAP メッセージ・ハンドラー) は、インバウンド要求から SOAP エンベロープを除去し、SOAP 本体をデータ・マッパー機能に渡します。ここでは、SOAP 本体の内容をアプリケーションのデータ構造にマップするために、Web サービス・バインディング・ファイルを使用します。SOAP メッセージの完全な検証がアクティブである場合は、SOAP 本体が Web サービス記述に対して検証されます。アウトバウンド応答がある場合は、処理が反転します。

図 12 には、これらのファイルがサービス・リクエスターで使われる様子を示します。

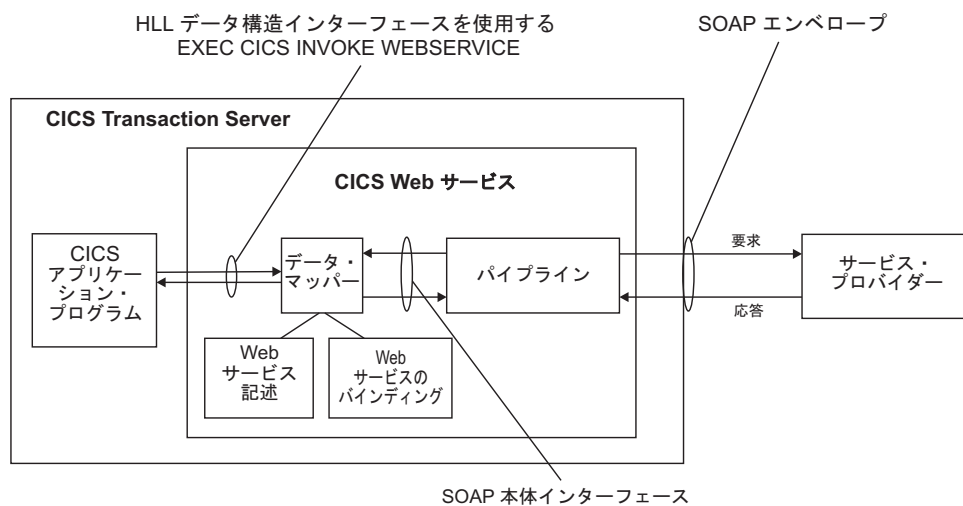


図 12. サービス・リクエスターでの SOAP 本体からアプリケーション・データ構造へのマッピング

アウトバウンド要求では、データ・マッパー機能が、Web サービス・バインディング・ファイルからの情報を使用して、アプリケーションのデータ構造から SOAP 本体の構成を行います。パイプラインのメッセージ・ハンドラー (一般に、CICS 提供の SOAP メッセージ・ハンドラー) は、SOAP エンベロープを追加します。インバウンド応答がある場合は、処理が反転します。SOAP メッセージの完全な検証がアクティブである場合は、インバウンド SOAP 本体が Web サービス記述に対して検証されます。

どちらの場合も、特定の CICS アプリケーション・プログラムが Web サービスの設定で動作できる実行環境は、3 つのオブジェクトによって定義されます。これらは、パイプライン、Web サービス・バインディング・ファイル、および Web サービス記述です。これら 3 つのオブジェクトは、WEBSERVICE リソース定義の属性として CICS に定義されています。

SOAP メッセージを使用している場合でも、次に示すように、CICS Web サービス・アシスタントが生成する変換を使用できない状況がいくつか存在します。

- SOAP メッセージと高水準言語で同じデータが表現できない場合

CICS がサポートしているすべての高水準言語、および XML スキーマでは、さまざまなデータ・タイプがサポートされています。しかし、高水準言語で 사용되는データ・タイプと XML スキーマで 사용되는データ・タイプとの間に 1 対 1 対応は存在しないため、データを一方で表現できても他方では表現できないという場合が存在します。こうした状況では、次のいずれかの手段を検討する必要があります。

- アプリケーション・データ構造を変更します。この方法は、必然的にアプリケーション・プログラム自体の変更が必要になるため、実現は困難です。
- ラッパー・プログラムを作成します。このプログラムは、アプリケーション・データを CICS が処理可能な形式に変換し、さらに SOAP メッセージ本文に変換します。この方法を実行した場合は、アプリケーション・プログラムを変更せずに済みます。この場合は、CICS Web サービス・サポートがラッパー・プログラムと直接対話し、アプリケーション・プログラムとは間接的にのみ対話します。
- アプリケーション・プログラムが、CICS Web サービス・アシスタントでサポートされない言語で記述されている場合

こうした状況では、次のいずれかの手段を検討する必要があります。

- CICS Web サービス・アシスタントがサポートするいずれかの言語 (COBOL、PL/I、C または C++) で記述されたラッパー・プログラムを作成します。
- CICS Web サービス・アシスタントを使用する代わりに、SOAP メッセージとアプリケーション・プログラムのデータ構造間のマッピングを行うプログラムを独自に作成します。

WSDL およびアプリケーション・データ構造

Web サービス記述には、Web サービスが使用する入出力メッセージの抽象表現が含まれています。CICS では、Web サービス記述を使用して、アプリケーション・プログラムが使用するデータ構造を構成します。CICS は、実行時に、アプリケーション・データ構造とメッセージとのマッピングを実行します。

Web サービス記述の代表例は、以下のとおりです。

- 1 つ以上の操作
- 各操作ごとに、入力メッセージ、およびオプションの出力メッセージ。
- メッセージごとに、XML データ・タイプの観点で定義されたメッセージ構造。
メッセージ内で使用される複合データ・タイプは、Web サービス記述内にある `<types>` エレメントに記述されている XML スキーマで定義されます。簡単なメッセージは、`<types>` エレメントを使用しないで記述できます。

WSDL には、操作の抽象定義と関連メッセージが記述されています。WSDL をアプリケーション・プログラム内に直接使用することはできません。操作を実装するには、サービス・プロバイダーが以下の処理を実行する必要があります。

- メッセージの構造を把握するために WSDL の構文解析を行う。
- 入力メッセージを解析して出力メッセージを作成する。
- 入出力メッセージの内容と、アプリケーション・プログラムで使用されているデータ構造とのマッピングを実行する。

サービス・リクエスターは、操作を呼び出すために同じことを行う必要があります。

CICS Web サービス・アシスタントを使用すると、前述の大半の処理がユーザーの代わりに実行されるため、ユーザーは入出力メッセージを構成する方法や WSDL を詳細に理解する必要なくアプリケーション・プログラムを作成できます。

CICS Web サービス・アシスタントは、以下の 2 つのユーティリティー・プログラムで構成されています。

DFHWS2LS

このユーティリティー・プログラムは、Web サービス記述を開始点にしています。このプログラムでは、アプリケーション・プログラムに使用できる高水準言語データ構造を構成するために、メッセージの記述や、メッセージに使用されているデータ・タイプを使用します。

DFHLS2WS

このユーティリティー・プログラムは、高水準言語データ構造を開始点にしています。このプログラムでは、メッセージの記述を格納する Web サービス記述を構成するための構造体と、言語構造から導出されたこれらのメッセージで使用されるデータ・タイプが使用されます。

いずれのユーティリティー・プログラムも、アプリケーション・プログラムのデータ構造と SOAP メッセージ間のマッピングを実行するために CICS が実行時に使用する Web サービス・バインディング・ファイルを生成します。

COBOL から WSDL へのマッピングの例

この例では、COBOL プログラムで使用されているデータ構造が、CICS Web サービス・アシスタントによって生成された Web サービス記述内でどのように表現されているかを示します。

図 13 は、単純な COBOL データ構造を示しています。

```
*   カタログ COMMAREA 構造
      03 CA-REQUEST-ID          PIC X(6).
      03 CA-RETURN-CODE         PIC 9(2).
      03 CA-RESPONSE-MESSAGE    PIC X(79).
*   Place Order (発注) で使用されているフィールド
      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).
```

図 13. WSDL で定義されている入力メッセージの COBOL レコード定義

Web サービス記述の対応するフラグメントでの重要なエレメントを、32 ページの図 14 に示します。

```

<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>

```

図 14. COBOL データ構造から導出された WSDL フラグメント

WSDL およびメッセージ交換パターン

WSDL 2.0 文書には、SOAP 1.2 メッセージを Web サービス・リクエスターと Web サービス・プロバイダーの間で交換する方法を定義する、メッセージ交換パターン (MEP) が含まれます。

CICS は、サービス・プロバイダー・アプリケーションとサービス・リクエスター・アプリケーションの両方について、*WSDL 2.0 Part 2: Adjuncts* 仕様および *WSDL 2.0 Part 2: Additional MEPs* 仕様で定義される 8 つのメッセージ交換パターンのうち 4 つをサポートします。以下の MEP がサポートされています。

In-Only

要求メッセージは Web サービス・プロバイダーに送信されますが、プロバイダーは Web サービス・リクエスターにどのようなタイプの応答を送信することも許可されていません。

- プロバイダー・モードで、CICS が In-Only MEP を使用する Web サービスから要求メッセージを受け取るとき、応答メッセージは戻されません。DFHNORESPONSE コンテナが SOAP ハンドラー・チャンネルに入れられて、パイプラインが応答メッセージを送信してはならないことが示されます。
- リクエスター・モードでは、CICS は要求メッセージを Web サービス・プロバイダーに送信して、応答を待機しません。

In-Out 要求メッセージが Web サービス・プロバイダーに送信され、応答メッセージが Web サービス・リクエスターに戻ります。応答メッセージは通常の SOAP メッセージまたは SOAP 障害です。

- プロバイダー・モードで、CICS が In-Out MEP を使用する Web サービスから要求メッセージを受け取るとき、応答メッセージがリクエスターに戻されます。
- リクエスター・モードでは、CICS は要求メッセージを送信して、応答を待機します。この応答は、正規応答メッセージまたは SOAP 障害メッセージのどちらかです。CICS が応答を待機する時間の長さは、パイプラインで構成されて、そのパイプラインを使用するすべての Web サービスに適用されます。CICS が応答を受け取る前に要求がタイムアウトになる場合、サービス・リクエスター・アプリケーションにエラーが戻されません。

In-Optional-Out

要求メッセージが Web サービス・プロバイダーに送信され、応答メッセージはオプションで Web サービス・リクエスターに戻ります。応答がある場合、応答は通常の SOAP メッセージまたは SOAP 障害のいずれかです。

- プロバイダー・モードでは、SOAP 応答メッセージまたは SOAP 障害のどちらを戻すか、または何も戻さないかの判断は実行時に行われて、サービス・プロバイダーのアプリケーション・ロジックに依存しています。CICS が応答を Web サービス・リクエスターに送信しない場合、DFHNORESPONSE コンテナが SOAP ハンドラー・チャンネルに入れられて、パイプラインが応答メッセージを送信してはならないことが示されます。メッセージが送信されない場合、サービス・プロバイダー・アプリケーションは DFHWS-DATA コンテナをチャンネルから削除する必要があります。
- リクエスター・モードでは、CICS は要求メッセージを送信して、Web サービス・リクエスターからの応答を待機します。応答を受け取る前に要求がタイムアウトになる場合、CICS は、メッセージが正常に受け取られたためにプロバイダーは応答を送信する必要がなかったと想定します。CICS が応答を待機する時間の長さは、パイプラインで構成されて、そのパイプラインを使用するすべての Web サービスに適用されます。

Robust In-Only

要求メッセージが Web サービス・プロバイダーに送信され、エラーが生じた場合にのみ、Web サービス・リクエスターに応答メッセージが戻ります。エラーがある場合は、SOAP 障害メッセージがリクエスターに送信されます。

- プロバイダー・モードでは、パイプラインが要求メッセージをアプリケーションに正常に渡した場合、DFHNORESPONSE コンテナが SOAP ハ

ンドラー・チャンネルに入れられて、パイプラインが応答メッセージを送信してはならないことが示されます。パイプラインでエラーが生じた場合は、SOAP 障害メッセージがリクエスターに戻されます。

- リクエスター・モードでは、CICS は要求メッセージを Web サービス・プロバイダーに送信して、指定された期間待機してからタイムアウトします。CICS が応答を待機する時間の長さは、パイプラインで構成されて、そのパイプラインを使用するすべての Web サービスに適用されます。タイムアウトがある場合、CICS は要求メッセージが正常に受け取れたと想定します。

WSDL 2.0 でのメッセージ交換パターンの詳細については、以下の W3C 仕様を参照してください。

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

関連概念

299 ページの『メッセージ交換』

Web Services Addressing (WS-Addressing) は、片方向、両方向要求/応答、同期要求/応答、および非同期要求/応答のメッセージ交換をサポートしています。

Web サービスのバインディング・ファイル

Web サービスのバインディング・ファイル には、入出力メッセージとアプリケーション・データ構造との間でデータをマップするために CICS が使用する情報が格納されています。

Web サービス記述には、Web サービスが使用する入出力メッセージの抽象表現が含まれています。サービス・プロバイダーまたはサービス・リクエスターのアプリケーションを実行する場合、CICS に必要な情報は、メッセージの内容をアプリケーションが使用するデータ構造へマップする方法になります。この情報は、Web サービスのバインディング・ファイルに保持されます。

Web サービスのバインディング・ファイルの作成方法は、次のとおりです。

- 言語構造を WSDL を基に生成する場合は、ユーティリティー・プログラム DFHWS2LS
- WSDL を言語構造を基に生成する場合は、ユーティリティー・プログラム DFHLS2WS

実行時に、CICS は Web サービスのバインディング・ファイルの情報を使用してアプリケーション・データ構造と SOAP メッセージとのマッピングを行います。

Web サービスのバインディング・ファイルは、WEBSERVICE リソースの WSBIND 属性で、CICS に対して定義されます。

関連情報

WEBSERVICE リソース定義

外部標準

Web サービスの CICS サポートは、多数の業界標準および仕様に準拠しています。

SOAP 1.1 および 1.2

SOAP は、非集中の分散環境で情報を交換するための、XML ベースの単純なプロトコルです。

プロトコルは、次の 3 つの部分から構成されます。

- メッセージの内容およびその処理方法を記述するためのフレームワークを定義するエンベロープ。
- アプリケーション定義のデータ・タイプのインスタンスを表現するための一連のエンコード規則。
- リモート・プロシージャ・コールおよび応答を表現するための規則。

SOAP は、HTTP などの他のプロトコルと併用できる。

SOAP の仕様は、World Wide Web Consortium (W3C) によって公開されています。SOAP 1.1 の仕様は、<http://www.w3.org/TR/SOAP> にメモとして記載されています。この仕様は W3C によって公認されていませんが、SOAP 1.2 仕様の基礎となっています。この仕様は、SOAP 頭字語を Simple Object Access Protocol に展開します。

SOAP 1.2 は W3C 勧告であり、2 つの部分に分けて公開されています。

- Part 1: Messaging Framework は <http://www.w3.org/TR/soap12-part1/> で公開されます。
- Part 2: Adjuncts は <http://www.w3.org/TR/soap12-part2/> で公開されます。

この仕様には、SOAP バージョン 1.2 仕様の機能についての解説を提供する目的で、手引も組み込まれています。これには使用法のシナリオが含まれます。手引は、<http://www.w3.org/TR/soap12-part0/> で公開されます。SOAP 1.2 の仕様は頭字語を展開しません。

SOAP 1.1 Binding for MTOM 1.0

SOAP 1.1 Binding for MTOM 1.0 は、SOAP Message Transmission Optimization Mechanism (MTOM) 仕様と XML-binary Optimized Packaging (XOP) 仕様を SOAP 1.1 で使用する方法を記述する仕様です。

この仕様の目的は、MTOM と XOP の変更内容の定義を最小にし、これらの機能を SOAP 1.1 で相互運用できるようにして、SOAP 1.2 MTOM/XOP 実装を十分に再利用することです。

SOAP 1.1 Binding for MTOM 1.0 仕様は、World Wide Web Consortium (W3C) によって、正式な発信として <http://www.w3.org/Submission/soap11mtom10/> で公開されます。

SOAP Message Transmission Optimization Mechanism (MTOM)

SOAP Message Transmission Optimization Mechanism (MTOM) は、SOAP メッセージの送信と形式を最適化する方法を概念的に定義する、関連した 1 対の仕様の 1 つです。

MTOM で定義するものは次のとおりです。

1. 抽象的な条件で SOAP メッセージの base64binary データの送信を最適化する方法
2. XOP を使用するバインディングに依存しない方法で、SOAP メッセージの最適化された MIME multipart 直列化を実装する方法

MTOM の実装は、関連する XML-binary Optimized Packaging (XOP) 仕様に依存します。この 2 つの仕様は密接にリンクしているので、通常は MTOM/XOP として参照されます。

この仕様は、World Wide Web Consortium (W3C) によって W3C 勧告として <http://www.w3.org/TR/soap12-mtom/> で公開されます。

Web Services Addressing 1.0

Web Services Addressing 1.0 (WS-Addressing) は、Web サービス間でメッセージング情報をやり取りするための、トランスポートに依存しないメカニズムを定義する仕様です。

WS-Addressing 仕様は、2 つの構成体 (メッセージ・アドレッシング・プロパティと、エンドポイント参照) を定義します。これらは、通常、トランスポート・プロトコルやメッセージング・システムによって提供される情報を正規化します。

この仕様は World Wide Web Consortium (W3C) の勧告として、以下の 3 つの部分に分けて W3C により公開されています。

- WS-Addressing 1.0 - コア
- WS-Addressing 1.0 - SOAP バインディング
- WS-Addressing 1.0 - メタデータ

WS-Addressing を CICS で使用する場合、W3C 仕様に従うことをお勧めします。

相互運用性のために、名前空間が <http://schemas.xmlsoap.org/ws/2004/08/addressing> に設定されている場合に限り、CICS は W3C WS-Addressing サブミッション仕様を許容します。

CICS API コマンドは WS-Addressing 勧告仕様に従う MAP および EPR をサポートしますが、WS-Addressing サブミッション仕様に従う MAP および EPR は API コマンドによってサポートされません。

アドレッシング・コンテキストは、すべての MAP を勧告仕様のレベルで保守します。必要に応じて、これらの MAP を SOAP メッセージに適用するときにサブミッション仕様レベルに変換することができ、または SOAP メッセージから抽出するときにサブミッション仕様レベルから MAP に変換することができます。

Web Services Atomic Transaction バージョン 1.0

Web Services Atomic Transaction バージョン 1.0 (または WS-AtomicTransaction) は、短期間のトランザクションについてアトミック・トランザクション調整タイプを定義するプロトコルです。これは、Web Services Coordination バージョン 1.0 (または WS-Coordination) 仕様に記述される拡張可能な調整フレームワークと共に使用されます。

WS-AtomicTransaction 仕様および WS-Coordination 仕様では、複数のトランザクション処理システムを Web サービス環境で相互運用できる、短期間のトランザクション向けプロトコルが定義されます。WS-AtomicTransaction を使用するトランザクションには、原子性、一貫性、独立性および耐久性という *ACID* 特性があります。

WS-AtomicTransaction の仕様は、<http://www.ibm.com/developerworks/library/specification/ws-tx/> で公開されます。

Web Services Coordination バージョン 1.0

Web Services Coordination バージョン 1.0 (または WS-Coordination) は、分散アプリケーションのアクションを調整するプロトコルを提供するための、拡張可能なフレームワークです。これらの調整プロトコルは、多数のアプリケーションをサポートするために使用されます。アプリケーションには、分散アクティビティの結果について一貫した合意に達する必要があるものが含まれます。

このフレームワークによって、アプリケーション・サービスは、他のサービスにアクティビティを伝搬する必要のある状況を生み出し、調整プロトコルについて登録することができます。このフレームワークによって、調整のための既存のトランザクション処理、ワークフロー、およびその他のシステムが、専有プロトコルを隠し、異種の環境で動作するようにできます。

WS-Coordination の仕様は、<http://www.ibm.com/developerworks/library/specification/ws-tx/> で公開されます。

Web サービス記述言語バージョン 1.1 および 2.0

Web サービス記述言語 (WSDL) は、文書指向の情報またはプロシーチャー指向の情報のいずれかを含むメッセージで動作する一連のエンドポイントとして、ネットワーク・サービスを記述するための XML 形式です。

命令およびメッセージは、抽象的に記述されてから、具体的なネットワーク・プロトコルおよびメッセージ・フォーマットにバインドされて、エンドポイントを定義します。関連した具体的なエンドポイントは、抽象的なエンドポイント (サービス) に結合されます。

WSDL は拡張可能なので、通信に使用されるメッセージ・フォーマットやネットワーク・プロトコルに関係なく、エンドポイントおよびそれらのメッセージの記述を許可します。WSDL 1.1 仕様では、WSDL を SOAP 1.1、HTTP GET および POST、および MIME と一緒に使用方法を記述するバインディングのみを定義します。

WSDL 2.0 では、Web サービスの記述に、XML 形式に加えてモデルを使用します。このため、サービスによって提供された抽象機能の記述と、この機能が提供される仕組みや条件などのサービス記述の具体的な詳細とを分離できます。また、メッセージ交換パターンの拡張、SOAP モジュール、および SOAP 1.2 や HTTP についてこのような具体的な詳細を記述する言語も記述します。WSDL 2.0 仕様は、WSDL 1.1 における多数の技術的な問題および制限も解決します。

WSDL 1.1 の仕様は、World Wide Web Consortium (W3C) によって W3C メモとして <http://www.w3.org/TR/wsdl> で公開されます。

WSDL 2.0 の最新の仕様は、W3C 勧告候補として <http://www.w3.org/TR/wsdl20> で公開されます。

Web Services Security: SOAP Message Security

Web Services Security (WSS): SOAP Message Security は、メッセージの保全性と機密性を提供する SOAP メッセージの一連の機能強化です。WSS: SOAP Message Security は拡張可能で、さまざまなセキュリティー・モデルや暗号化テクノロジーに適応できます。

WSS: SOAP Message Security には 3 つの主要なメカニズムがあり、独立して使用することも一緒に使用することもできます。それらは次のとおりです。

- セキュリティー・トークンをメッセージの一部として送信し、セキュリティー・トークンとメッセージの内容を関連付ける機能
- メッセージの内容を無許可で検出されない変更から保護する機能 (メッセージ保全性)
- メッセージの内容を無許可の開示から保護する機能 (メッセージ機密性)

WSS: SOAP Message Security を、他の Web サービスの拡張機能やアプリケーション固有のプロトコルと一緒に使用して、さまざまなセキュリティー要件を満たすことができます。

この仕様は、Organization for the Advancement of Structured Information Standards (OASIS) によって公開され、<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf> に記載されています。

Web Services Trust Language

Web Services Trust Language (または WS-Trust) は、Web Services Security で構築される拡張機能を定義して、セキュリティー・トークンを要求および発行するフレームワークを提供し、信頼関係を仲介します。

WS-Trust は次のものを記述します。

1. セキュリティー・トークンを発行、更新、および検証するメソッド。
2. 信頼関係を確立し、それにアクセスし、仲介する方法。

CICS は、<http://www-128.ibm.com/developerworks/library/specification/ws-trust/> で公開された 2005 年 2 月のバージョンの仕様をサポートします。

WSDL 1.1 Binding Extension for SOAP 1.2

WSDL 1.1 Binding Extension for SOAP 1.2 は、Web サービスのメッセージが SOAP 1.2 プロトコルにバインドされることを示すために必要な、バインディング拡張機能を定義する仕様です。

この仕様の目的は、SOAP 1.1 のバインディングと同程度の機能を提供することです。

この仕様は、World Wide Web Consortium (W3C) によって、正式な実行依頼要求として <http://www.w3.org/Submission/wsdl11soap12/> で公開されます。

WS-I Basic Profile バージョン 1.1

WS-I Basic Profile バージョン 1.1 (WS-I BP 1.1) は、非専有の一連の Web サービス仕様であり、これらの仕様の説明および改訂も付記されています。これらを総合することにより、Web サービスのさまざまな実装環境間でインターオペラビリティを実現することができます。

WS-I BP 1.1 は、Basic Profile バージョン 1.0 の公開済みの正誤表を取り込み、エンベロープの直列化とメッセージ内の表記に関連する要件を分離することによって、Basic Profile バージョン 1.0 から派生しました。これらの要件は Simple SOAP Binding Profile バージョン 1.0 の 2 つの部分になりました。

要約すると、WS-I Basic Profile バージョン 1.0 は 2 つの別々に公開されるプロファイルに分割されました。この内容は次のとおりです。

- WS-I Basic Profile バージョン 1.1
- WS-I Simple SOAP Binding Profile バージョン 1.0

これらの 2 つの Profile の組み合わせが、WS-I Basic Profile バージョン 1.0 を置き換えます。

Basic Profile 1.1 を、Simple SOAP Binding Profile 1.0 を含む、エンベロープの直列化を指定するどのプロファイルとも一緒に構成できるようにするために、この分離が行われました。

WS-I BP 1.1 の仕様は、Web Services Interoperability Organization (WS-I) によって公開され、<http://www.ws-i.org/Profiles/BasicProfile-1.1.html> に記載されています。

WS-I Simple SOAP Binding Profile バージョン 1.0

WS-I Simple SOAP Binding Profile バージョン 1.0 (SSBP 1.0) は、非専有の一連の Web サービス仕様であり、インターオペラビリティを実現するこれらの仕様の説明および改訂も付記されています。

SSBP 1.0 は、エンベロープの直列化とメッセージ内の表記に関連する WS-I Basic Profile 1.0 要件から派生しました。

WS-I Basic Profile 1.0 は 2 つの別々に公開されるプロファイルに分割されました。この内容は次のとおりです。

- WS-I Basic Profile バージョン 1.1
- WS-I Simple SOAP Binding Profile バージョン 1.0

これらの 2 つの Profile の組み合わせが、WS-I Basic Profile バージョン 1.0 を置き換えます。

SSBP 1.0 の仕様は、Web Services Interoperability Organization (WS-I) によって公開され、<http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html> に記載されています。

XML (Extensible Markup Language) バージョン 1.0

Extensible Markup Language (XML) 1.0 は、SGML のサブセットです。現在では HTML で実現できるような方法で、一般的な SGML を Web 上でサービス、受信、および処理できるようにすることが目的です。

XML は、SGML と HTML の両方で実装とインターオペラビリティを簡単にするために設計されました。

XML 1.0 の仕様と正誤表は、World Wide Web Consortium (W3C) によって W3C 勧告として <http://www.w3.org/TR/REC-xml> で公開されます。

XML-binary Optimized Packaging (XOP)

XML-binary Optimized Packaging (XOP) は、特定のタイプの内容を持つ XML Infoset を効率的に直列化する方法を定義する、関連した 1 対の仕様の 1 つです。

XOP は次のようにしてこれを行います。

1. XML をある形式でパッケージ化する。これは *XOP* パッケージと呼ばれます。仕様では MIME Multipart/Related について言及していますが、この形式に制限されているわけではありません。
2. base64binary の内容の全部または一部を再エンコードして、サイズを削減する。
3. base64binary の内容をパッケージ内の他の場所に配置して、これを参照する XML でエンコードされた内容を置き換える。

XOP は、SOAP メッセージの最適化を定義する MTOM 仕様の実装として使用されます。この 2 つの仕様は密接にリンクしているので、通常は MTOM/XOP として参照されます。

この仕様は、World Wide Web Consortium (W3C) によって W3C 勧告として <http://www.w3.org/TR/xop10/> で公開されます。

XML Encryption Syntax and Processing

XML Encryption Syntax and Processing は、データを暗号化し、XML に結果を表示する処理を指します。対象のデータは、任意のデータ (XML 文書を含む)、XML エlement、または XML Element の内容です。データ暗号化の結果は、暗号データを含むか参照する XML Encryption Element です。

XML Encryption Syntax and Processing は World Wide Web Consortium (W3C) の勧告であり、<http://www.w3.org/TR/xmlenc-core> で公開されます。

XML-Signature Syntax and Processing

XML-Signature Syntax and Processing は、XML デジタル署名の規則および構文を指定します。

XML デジタル署名は、署名を含む XML 内にあっても他の場所にあっても、任意のタイプのデータについて、保全性、メッセージ認証、および署名者認証サービスを提供します。

XML-Signature の仕様は、World Wide Web Consortium (W3C) によって <http://www.w3.org/TR/xmlsig-core> で公開されます。

CICS の Web サービス標準への準拠

CICS は、サポートされる Web サービス標準および仕様に準拠しているため、準拠している Web サービスを生成および配置できます。

CICS はこの準拠を強制しないことに注意してください。例えば、WS-I Basic Profile 1.1 仕様のサポートの場合、CICS では、この Profile で概説するインターオペラビリティを失わせる可能性のある追加のサービス品質を Web サービスに適用できます。

CICS が WS-Addressing に準拠する仕組み

CICS は、WS-Addressing 仕様のコアおよび SOAP バインディング部分に準拠しています。CICS は、1 つの例外を除いて仕様のメタデータ部分に準拠しています。

CICS が WS-Addressing 障害を発行するときには、仕様に準拠しません。WS-Addressing 障害を構築する際に CICS はメタデータ仕様に記述されたデフォルト・アクションに関する形式に従いますが、最後の区切り文字と障害名を含みません。

WSDL 1.1 の場合、仕様によると、デフォルト・アクションは次のとおりです。

```
[target namespace][delimiter][port type name][delimiter][operation name][delimiter]Fault[delimiter][fault name]
```

しかし、CICS は障害名を省略して、次のようなデフォルト・アクションを構築します。

```
[target namespace][delimiter][port type name][delimiter][operation name][delimiter]Fault[delimiter]
```

WSDL 2.0 の場合、仕様によると、デフォルト・アクションは次のとおりです。

```
[target namespace][delimiter][interface name][delimiter][fault name]
```

しかし、CICS は障害名を省略して、次のようなデフォルト・アクションを構築します。

```
[target namespace][delimiter][interface name][delimiter]
```

CICS が WSDL 2.0 に準拠する仕組み

CICS は条件付きで WSDL 2.0 に準拠します。サポートは以下の制限に従って行われます。

必須要件

- WSDL で使用されるのは、メッセージ交換パターン in-only、in-out、robust in-only、および in-optional-out のみ。
- 各サービスで許可されるのは 1 つのエンドポイントのみ。
- 少なくとも 1 つの操作が必要。
- エンドポイントは URI で指定されるのみ。
- SOAP バインディングが必要。
- XML スキーマ・タイプ・システムを使用する必要がある。

許容される局面

- 以下の HTTP バインディング・プロパティーは無視される。
 - whttp:location
 - whttp:header
 - whttp:transferCodingDefault
 - whttp:transferCoding
 - whttp:cookies
 - whttp:authenticationType
 - whttp:authenticationRealm
- SOAP ヘッダー情報は DFHWS2LS によって無視される。ただし、独自のメッセージ・ハンドラーをパイプラインに追加して、インバウンド・メッセージおよびアウトバウンド・メッセージについて、必要な SOAP ヘッダー情報を作成および処理できます。

サポートされない局面

- #any および #other メッセージの内容モデル。
- out-only、robust-out-only、out-in および out-optional-in メッセージ交換パターン。
- エンドポイントの WS-Addressing。
- HTTP GET はサポートされない。これは、WSDL 文書の soap-response メッセージ交換パターンを使用して定義されます。ご使用の WSDL がこのメッセージ交換パターンを定義すると、DFHWS2LS がエラー・メッセージを出します。

CICS が Web Services Security 仕様に準拠する仕組み

CICS は、以下の局面をサポートすることで、条件付きで Web Services Security: SOAP Message Security および関連した仕様に準拠します。

Web Services Security: SOAP Message Security との準拠

セキュリティー・ヘッダー

<wsse:Security> ヘッダーは、セキュリティー関連の情報を特定の宛先に向けて SOAP アクターまたは役割の形式で接続するメカニズムを提供します。これがメッセージまたは中間ノードの最終的な宛先になる場合があります。CICS では次の属性がサポートされます。

- S11:actor (中間ノード用)
- S11:mustUnderstand
- S12:role (中間ノード用)
- S12:mustUnderstand

セキュリティー・トークン

セキュリティー・ヘッダーでは以下のセキュリティー・トークンがサポートされます。

- ユーザー名とパスワード
- バイナリー・セキュリティー・トークン (X.509 証明書)

トークン参照

セキュリティー・トークンは一連のクレームを伝えます。これらのクレーム

が外部に常駐していて、受信側のアプリケーションがアクセスする必要がある場合があります。<wsse:SecurityTokenReference> エレメントは、セキュリティー・トークンの参照について、拡張可能なメカニズムを提供します。次のメカニズムがサポートされます。

- 直接参照
- 鍵 ID
- 鍵名
- 組み込み参照

署名アルゴリズム

この仕様は XML Signature を基にして作成されるため、XML Signature 仕様で指定されるものと同じアルゴリズム要件を持ちます。CICS は以下をサポートします。

アルゴリズム・タイプ	アルゴリズム	URI
Digest	SHA1	http://www.w3.org/2000/09/xmlsig#sha1
Signature	SHA1 を使用する DSA (検証のみ)	http://www.w3.org/2000/09/xmlsig#dsa-sha1
Signature	SHA1 を使用する RSA	http://www.w3.org/2000/09/xmlsig#rsa-sha1
Canonicalization	Exclusive XML 正規化 (コメントなし)	http://www.w3.org/2001/10/xml-exc-c14n#

署名部分の署名

CICS では次の SOAP エレメントに署名できます。

- SOAP メッセージの本文
- 宣言 ID として使用される ID トークン (セキュリティー・トークンのタイプ)

暗号化アルゴリズム

次のデータ暗号化アルゴリズムがサポートされます。

アルゴリズム	URI
Triple Data Encryption Standard algorithm (Triple DES)	http://www.w3.org/2001/04/xmlenc#tripledes-cbc
Advanced Encryption Standard (AES) アルゴリズム (鍵の長さは 128 ビット)	http://www.w3.org/2001/04/xmlenc#aes128-cbc
Advanced Encryption Standard (AES) アルゴリズム (鍵の長さは 192 ビット)	http://www.w3.org/2001/04/xmlenc#aes192-cbc
Advanced Encryption Standard (AES) アルゴリズム (鍵の長さは 256 ビット)	http://www.w3.org/2001/04/xmlenc#aes256-cbc

次の鍵暗号化アルゴリズムがサポートされます。

アルゴリズム	URI
鍵トランスポート (公開鍵暗号方式) RSA バージョン 1.5:	http://www.w3.org/2001/04/xmlenc#rsa-1_5

メッセージ部分の暗号化

CICS では次の SOAP エlementを暗号化できます。

- SOAP 本体

タイム・スタンプ

<wsu:Timestamp> Elementは、セキュリティー・セマンティクスの作成時間および満了時間をメッセージに表示するメカニズムを提供します。CICS は、インバウンド SOAP メッセージの Web サービスのセキュリティー・ヘッダー内でのタイム・スタンプの使用を許容します。

エラー処理

CICS は、仕様にリストされる応答コードの標準リストを使用して、SOAP 障害メッセージを生成します。

Web Services Security: UsernameToken Profile 1.0 との準拠

この仕様の次の局面がサポートされます。

パスワード・タイプ

テキスト

トークン参照

直接参照

Web Services Security: X.509 Certificate Token Profile 1.0 との準拠

この仕様の次の局面がサポートされます。

トークン・タイプ

- X.509 バージョン 3: 単一の証明書。 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf>を参照してください。
- X.509 バージョン 3: 証明書取り消しリスト (CRL) のない X509PKIPathv1。 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf>を参照してください。
- X.509 バージョン 3: CRL がある PKCS7、または CRL がない PKCS7。 IBM Software Development Kit (SDK) は両方ともサポートします。 Sun Java Development Kit (JDK) は CRL がない PKCS7 のみをサポートします。

トークン参照

- 鍵 ID - 所有者の鍵 ID
- 直接参照
- カスタム参照 - 発行者の名前およびシリアル番号

サポートされない局面

CICS では次の項目はサポートされません。

- 新鮮さを判断するためのタイム・スタンプの検証
- Nonce
- SOAP 接続についての Web サービス・セキュリティ
- Security Assertion Markup Language (SAML) トークン・プロファイル、WS-SecurityKerberos トークン・プロファイル、および XrML トークン・プロファイル
- Web Services Interoperability (WS-I) Basic Security Profile
- XML エンベロープのデジタル署名
- XML エンベロープのデジタル暗号化
- デジタル署名用の次のトランスポート・アルゴリズムはサポートされません。
 - XSLT: <http://www.w3.org/TR/1999/REC-xslt-19991116>
 - SOAP Message Normalization。詳しくは、<http://www.w3.org/TR/2003/NOTE-soap12-n11n-20031008/>を参照してください。
- 暗号化に使用される Diffie-Hellman 鍵共有アルゴリズムはサポートされません。詳しくは、<http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/Overview.html#sec-DHKeyValue>を参照してください。
- XML 暗号化仕様ではオプションである、暗号化のための次の正規化アルゴリズムは、サポートされません。
 - コメントがある Canonical XML またはコメントがない Canonical XML
 - コメントがある、またはない Exclusive XML 正規化
- Username Token バージョン 1.0 Profile 仕様では、ダイジェストのパスワード・タイプはサポートされません。

CICS が WS-Trust に準拠する仕組み

CICS は条件付きで WS-Trust に準拠します。サポートは以下の制限に従って行われます。

サポートされる局面

- バインディングの検証
- あるトークンが戻される場合のバインディングの実行
- バインディング実行時の AppliesTo

許容される局面

- 要求した参照
- 鍵およびエントロピー
- 計算された鍵の戻り

サポートされない局面

- 複数のセキュリティ・トークンの戻り
- ヘッダーへのセキュリティ・トークンの戻り
- バインディングの更新
- バインディングの取り消し

- ネゴシエーションおよびチャレンジの拡張
- 鍵およびトークンのパラメーターの拡張
- 鍵交換トークンのバインディング

CICS が WS-I Basic Profile 1.1 に準拠する仕組み

CICS は条件付きで WS-I Basic Profile 1.1 に準拠するため、すべての必須 レベル要件に従います。ただし、CICS は特に UDDI レジストリー対応ではないので、この仕様に関する点は無視されます。また、特定のスキーマ・エレメントのマッピングのサポートに制限があるため、Web サービス・アシスタントのジョブおよび関連する実行時環境は、この Profile に完全に準拠するわけではありません。

サポートされないスキーマ・エレメントのリストについては、186 ページの『高水準言語と XML のスキーマ・マッピング』を参照してください。

準拠の対象は、要件が適合する成果物 (SOAP メッセージ、WSDL 記述など) や関係者 (SOAP プロセッサ、エンド・ユーザーなど) を識別します。CICS がサポートする準拠の対象は次のとおりです。

MESSAGE

ENVELOPE をトランスポートするプロトコル・エレメント (SOAP over HTTP メッセージなど)。

ENVELOPE

soap:Envelope エレメントの逐次化およびその内容。

DESCRIPTION

タイプ、メッセージ、インターフェースおよびインターフェースのプロトコルおよびデータ・フォーマット・バインディング、および Web サービスに関連付けられたネットワーク・アクセス・ポイントの記述 (WSDL 記述など)。

INSTANCE

wsdl:port を実装するソフトウェア。

CONSUMER

INSTANCE を呼び出すソフトウェア。

SENDER

関連付けられたプロトコルに応じてメッセージを生成するソフトウェア。

RECEIVER

関連付けられたプロトコルに応じてメッセージをコンシュームするソフトウェア。

第 5 章 Web サービス入門

CICS で Web サービスを始めるには、いくつかの方法があります。最適な方法は、対象の題材に関する習得済みの知識の量や、Web サービスを使用する計画の進捗度によって異なります。

このタスクについて

CICS での Web サービスに関するいくつかの開始点を以下に示します。

- アプリケーションの例をインストールします。CICS には、Web サービス・プロバイダーとして使用できるカタログ管理アプリケーションの例が用意されています。この例には、最小限の作業量でアプリケーションを CICS で動作させるために必要なすべてのコードおよびリソース定義が格納されています。ここには、多くの共通 Web サービス・クライアント上で実行されるサービスと対話するためのコードも収録されています。

CICS で Web サービスを配置できる迅速な「概念実証」デモンストレーションが必要な場合や、CICS での Web サービスを習得するための「実践」方式が必要な場合は、この実例アプリケーションを使用してください。

実例アプリケーションについては、`../com.ibm.cics.ts.exampleapplication.doc/topics/dfhxa_t100.dita`で説明します。

- サービス・プロバイダーまたはサービス・リクエスターとしてアプリケーションを配置する作業の計画にすぐに取り掛かります。CICS で Web サービスを使用することによってアプリケーションおよび関連インフラストラクチャーの計画を開始する方法については、既に十分な知識があることが前提となっています。
- CICS for SOAP 機能からマイグレーションします。この機能を使用する既存のアプリケーションがある場合は、アプリケーションの再配置方法の計画を開始する準備ができています。

Web サービス使用の計画立案

始める前に

CICS で Web サービスを使用する計画を立てるには、その前に以下の問題をアプリケーションごとに検討する必要があります。

サービス・プロバイダーまたはサービス・リクエスターの役割で CICS アプリケーションを配置する計画ですか？

Web サービスの CICS サポートを使用して接続することを求められている 1 組のアプリケーションが存在する場合があります。この場合、一方のアプリケーションはサービス・プロバイダー、もう一方はサービス・リクエスターになります。

既存のアプリケーション・プログラムを使用する計画ですか、それとも新規のアプリケーションを作成する計画ですか？

既存のアプリケーションが、適切に定義されたビジネス・ロジックのインタ

ーフェースを使用して設計されている場合は、このアプリケーションを、サービス・プロバイダーまたはサービス・リクエスターとして Web サービス設定に使用できる確率が高くなります。ただし、ほとんどの場合は、ビジネス・ロジックを Web サービス・ロジックに接続するラッパー・プログラムを作成する必要があります。

新規アプリケーションの作成を計画している場合は、ビジネス・ロジックを Web サービス・ロジックから分離した状態を維持するようにします。さらにこの場合も、この分離状態を実現するためにラッパー・プログラムを作成する必要があります。ただし、アプリケーションが Web サービスを考慮して設計されている場合、ラッパーは簡単に作成できる可能性が高くなります。

SOAP メッセージを使用する予定ですか？

SOAP は、Web サービス・アーキテクチャーの基本であり、CICS で提供されているサポートの多くでは、SOAP の使用が前提となっています。ただし、他のメッセージ・フォーマットを使用したい状況も考えられます。例えば、CICS Web サービス・インフラストラクチャーを使用して配置する独自のメッセージ・フォーマットを作成してある場合などです。CICS では、こうした処理が可能です。ただし、Web サービス・アシスタント、SOAP メッセージ・ハンドラーなど、CICS が提供する機能の一部は使用できなくなります。

SOAP を使用しないことにした場合は、アプリケーション・プログラムには、インバウンド・メッセージの解析とアウトバウンド・メッセージの作成を行う役割が与えられます。

データ構造と SOAP メッセージ間のマッピングを生成するために CICS Web サービス・アシスタントを使用する予定ですか？

Web サービス・アシスタントは、アプリケーションを Web サービス設定に迅速に配置する機能を備えています。その際、追加のプログラミングはほとんど必要ありません。さらに、追加のプログラミングが必要な場合でも、通常は簡単で、既存のビジネス・ロジックを変更せずに済みます。

ただし、Web サービス・アシスタントを使用せずに処理した方がうまく処理できる場合があります。例えば、データ構造を SOAP メッセージにマップする既存のコードがある場合は、Web サービス・アシスタントを使用してアプリケーションを再構築しても、メリットはありません。

CICS Web サービス・アシスタントは、最も一般的なデータ・タイプおよびデータ構造をサポートしますが、サポートされていないデータ・タイプやデータ構造もいくつかあります。この状況では、該当する言語にサポートされていないデータ・タイプと構造のリストをチェックし、アプリケーションのデータを、アシスタントがサポートできる形式にマップするプログラム層を準備することを考慮する必要があります。これが不可能な場合は、自分でメッセージを解析する必要があります。アシスタントがサポートできるものとサポートできないものについて詳しくは、186 ページの『高水準言語と XML のスキーマ・マッピング』を参照してください。

CICS Web サービス・アシスタントを使用しないことにした場合は、Rational Developer for System z などのツールを使用して必要な成果物を作成し、インバウンド・メッセージの解析とアウトバウンド・メッセージの作

成を行うための独自のコードを提供することができます。また、提供されるベンダーのインターフェース API を使用することもできます。

既存のサービス記述を使用する予定ですか、それとも新規のサービス記述を作成する予定ですか？

状況によっては、既存のサービス記述を開始点として使用する必要があります。例を次に示します。

- アプリケーションはサービス・リクエスターであり、既存の Web サービスを呼び出すよう設計されている。
- アプリケーションはサービス・プロバイダーであり、既存の業界標準サービス記述にこのアプリケーションを適合させることを目的としている。

その他の状況では、アプリケーションに応じて新規のサービス記述を作成する必要があります。

次のタスク

- サービス・プロバイダーの計画
- サービス・リクエスターの計画

関連情報

CICS カタログ・マネージャーの実例アプリケーション

サービス・プロバイダー・アプリケーションの計画

一般に、CICS アプリケーションは、ビジネス・ロジックとコミュニケーション・ロジックを確実に分離できるよう構造化する必要があります。この手法に従うと、新規および既存のアプリケーションを Web サービス・プロバイダーで直接的に配置するのに役立ちます。状況によっては、アプリケーション・プログラムと CICS Web サービス・サポートとの間に単純なラッパー・プログラムを介在させる必要があります。

図 15 には、コミュニケーション・ロジックとビジネス・ロジックとを確実に分離するために分割された標準的なアプリケーションを示します。

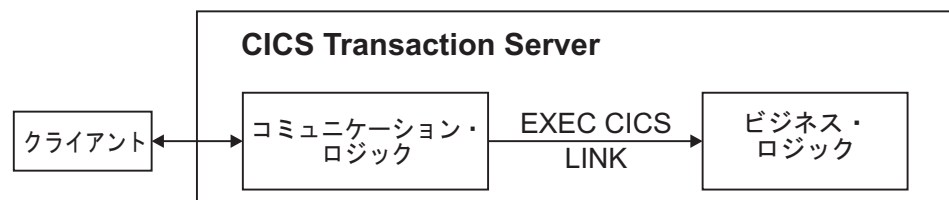


図 15. コミュニケーション・ロジックとビジネス・ロジックに分割されたアプリケーション

多くの場合、ビジネス・ロジックは、サービス・プロバイダー・アプリケーションの場合と同様に直接配置できます。このことは、50 ページの図 16 に図示されています。

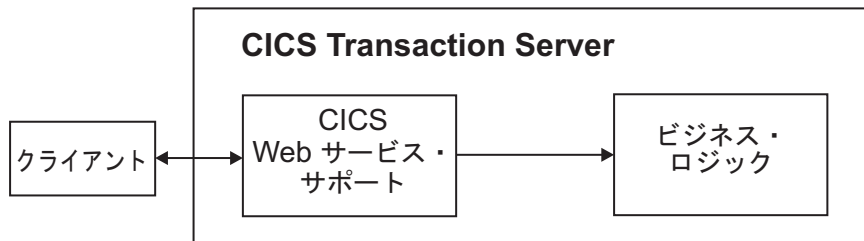


図 16. Web サービス・プロバイダーとしての CICS アプリケーションの単純な配置

この単純なモデルを使用する場合は、以下の条件が適用されます。

CICS Web サービス・アシスタントを使用して SOAP メッセージとアプリケーション・データ構造間のマッピングを生成する場合:

ビジネス・ロジックのインターフェースで使用するデータ・タイプは、CICS Web サービス・アシスタントによってサポートされている必要があります。これに該当しない場合は、CICS Web サービス・サポートとビジネス・ロジックとの間にラッパー・プログラムを介在させる必要があります。

既存のプログラムを配置して既存の Web サービス記述に適合するサービスを提供する場合は、ラッパー・プログラムも必要になります。Web サービス・アシスタントを使用して Web サービス記述を処理すると、結果として得られるデータ構造がビジネス・ロジックのインターフェースと一致する可能性は非常に低くなります。

CICS Web サービス・アシスタントを使用していない場合:

サービス・プロバイダー・パイプラインに存在するメッセージ・ハンドラーは、ビジネス・ロジックと直接対話する必要があります。

ラッパー・プログラムの使用

CICS Web サービス・アシスタントではビジネス・ロジックと直接対話するためのコードを生成できない場合は、ラッパー・プログラムを使用します。例えば、ビジネス・ロジックのインターフェースは、CICS Web サービス・アシスタントが SOAP メッセージに直接マップできないデータ構造を使用する可能性があります。この状況では、ラッパー・プログラムを使用すると、必要なデータ操作を追加できます。

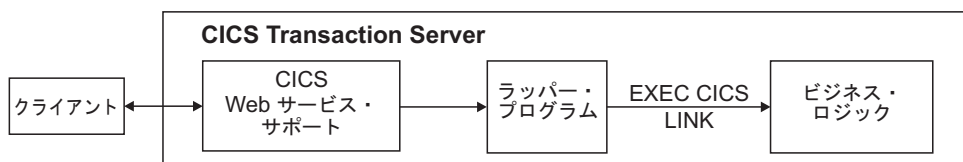


図 17. ラッパー・プログラム使用による、Web サービス・プロバイダーとしての CICS アプリケーションの配置

アシスタントがサポートできる 2 番目のデータ構造を設計して、これをラッパー・プログラムのインターフェースとして使用する必要があります。この結果、ラッパー・プログラムが実行する機能は、以下に示す 2 つの単純な機能となります。

- 2 つのデータ構造間でのデータの移動
- 既存のインターフェースによるビジネス・ロジックの呼び出し

エラー処理

CICS Web サービス・アシスタントを使用する予定がある場合は、エラーが発生したときに変更のロールバックを処理する方法も検討する必要があります。サービス・リクエスターから SOAP 要求メッセージを受信すると、SOAP メッセージはアプリケーション・プログラムに渡される直前に CICS によって変換されます。この変換中にエラーが発生すると、CICS はメッセージで実行された作業を自動的にロールバックしません。例えば、パイプラインでハンドラーを使用して SOAP メッセージに別の処理を追加する予定がある場合、既に実行したりカバリー可能な変更をロールバックするかどうかを決定する必要があります。

アウトバウンド SOAP メッセージでは、サービス・プロバイダー・アプリケーション・プログラムがサービス・リクエスターに応答メッセージを送信するような場合は、応答 SOAP メッセージの生成時に CICS がエラーを検出すると、アプリケーション・プログラムによって行われたリカバリー可能な変更はすべて自動的にバックアウトされます。ご使用のアプリケーション・プログラムにとって同期点を追加することが適切かどうかを検討する必要があります。

プロバイダー・アプリケーションで Web Services Atomic Transaction を使用する予定があり、さらに Web サービス・リクエスターがアトミック・トランザクションをサポートする場合は、CICS がトランザクションをロールバックするエラーが起これば、リモート・リクエスターも変更をロールバックするようになります。

サービス・リクエスター・アプリケーションの計画

一般に、CICS アプリケーションは、ビジネス・ロジックとコミュニケーション・ロジックを確実に分離できるよう構造化する必要があります。この手法に従うと、新規および既存のアプリケーションを Web サービス・リクエスターで直接的に配置するのに役立ちます。ほとんどの状況では、アプリケーション・プログラムと CICS Web サービス・サポートとの間に単純なラッパー・プログラムを介在させる必要があります。

52 ページの図 18 には、コミュニケーション・ロジックとビジネス・ロジックとを確実に分離するために分割された標準的なアプリケーションを示します。このアプリケーションは、Web サービス・リクエスターでビジネス・ロジックを再使用するために最適な構造になっています。

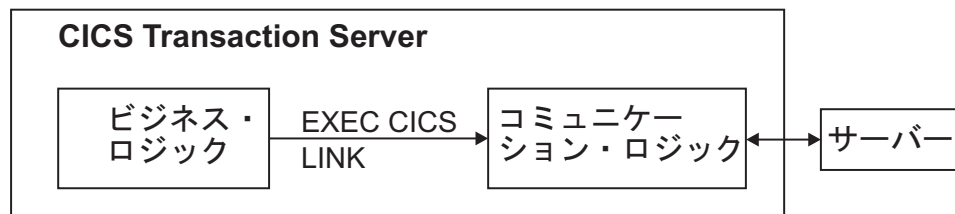


図 18. コミュニケーション・ロジックとビジネス・ロジックに分割されたアプリケーション

この状況では、既存の EXEC CICS LINK コマンドを使用して CICS Web サービス・サポートを呼び出すことはできません。

- CICS Web サービス・アシスタントを使用して SOAP メッセージとアプリケーション・データ構造間のマッピングを生成する場合は、EXEC CICS INVOKE SERVICE コマンドを使用して、アプリケーションのデータ構造を CICS Web サービス・サポートに渡す必要があります。また、ビジネス・ロジックのインターフェースで使用するデータ・タイプは、CICS Web サービス・アシスタントによってサポートされている必要があります。

ただし、アプリケーション・プログラムが呼び出すターゲット WEBSERVICE がプロバイダー・モードである場合、つまり、PROGRAM 属性の値が定義されている場合、CICS は EXEC CICS LINK コマンドを使用して要求を自動的に最適化します。

- CICS Web サービス・アシスタントを使用していない場合は、独自のメッセージを作成して、プログラム DFHPIRT にリンクする必要があります。

このため、いずれの場合でも、プログラムを変更する準備が整っていないかぎり、ビジネス・ロジックは Web サービスを直接呼び出すことができないことになります。Web サービス・アシスタントの場合、このオプションは図 19 に示されていますが、いずれの場合もお勧めできません。

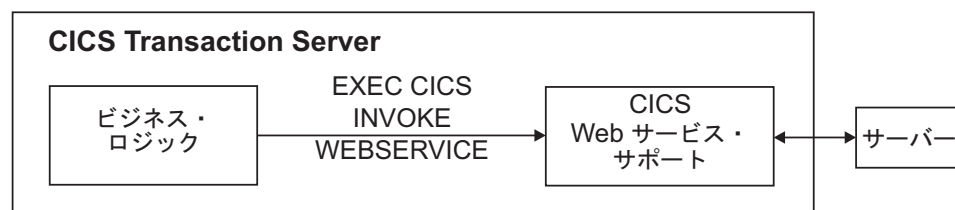


図 19. Web サービス・リクエスターとしての CICS アプリケーションの単純な配置

ラッパー・プログラムの使用

ビジネス・ロジックをほぼ未変更のまま維持する、より優れた解決策は、ラッパー・プログラムを使用することです。この場合、ラッパー・プログラムには次の 2 つの目的があります。

- ラッパー・プログラムは、ビジネス・ロジックの代わりに、EXEC CICS INVOKE SERVICE コマンド、つまり EXEC CICS LINK PROGRAM(DFHPIRT) を発行します。ビジネス・ロジックで変更されるのは、リンク先プログラムの名前だけです。
- CICS Web サービス・アシスタントが SOAP メッセージに直接マップできないデータ構造をアプリケーションが使用する場合、ラッパー・プログラムは、これに必要なデータ操作を必要に応じて提供できます。

Web サービス・アシスタントが使用された場合、この構造は 図 20 に示されます。



図 20. ラッパー・プログラム使用による、Web サービス・リクエスターとしての CICS アプリケーションの配置

エラー処理

CICS Web サービス・アシスタントを使用する予定がある場合は、エラーが発生したときに変更のロールバックを処理する方法も検討する必要があります。サービス・リクエスター・アプリケーションがサービス・プロバイダーから SOAP 障害メッセージを受信した場合は、アプリケーション・プログラムが障害メッセージを処理する方法を決定する必要があります。CICS は、SOAP 障害メッセージの受信時に変更を自動的にロールバックしません。

リクエスター・アプリケーション・プログラムに Web Services Atomic Transaction を実装する予定がある場合、エラー処理は異なります。リモート・サービス・プロバイダーがエラーを検出して、変更をロールバックすると、SOAP 障害メッセージが戻されて、CICS のローカル・トランザクションもロールバックされます。ローカルでの最適化が有効になっている場合は、サービス・リクエスターとサービス・プロバイダーは同じトランザクションを使用します。プロバイダーがエラーを検出すると、リクエスターでトランザクションが行った変更もすべてロールバックされます。

第 6 章 Web サービスに応じた CICS システムの構成

Web サービスを使用するには、CICS システムを正しく構成する必要があります。

1. PL/I のLanguage Environment®サポートをインストール済みであることを確認してください。詳しくは、「*CICS Transaction Server for z/OS インストール・ガイド*」を参照してください。
2. z/OS Support for Unicode を活動化します。z/OS 変換サービスを使用可能にして、CICS を使用して SOAP メッセージとアプリケーション・プログラム間で実行するデータ変換を指定する変換イメージをインストールする必要があります。詳しくは、「*z/OS Support for Unicode: 変換サービスの使用*」を参照してください。

Web サービスのための CICS リソース

CICS で Web サービスをサポートする CICS リソースを以下に示します。

PIPELINE

PIPELINE リソース定義は、あらゆる Web サービスに必要です。これは、サービス要求および応答に対して作用するメッセージ・ハンドラー・プログラムに関する情報を提供します。一般に、単一の PIPELINE 定義で定義されたインフラストラクチャーを、多数のアプリケーションで使用できます。メッセージ・ハンドラーに関する情報は、間接的に指定されます。

PIPELINE はハンドラーとその構成の XML 記述を格納する z/OS UNIX ファイルの名前を指定します。

サービス・リクエスター用に作成された PIPELINE リソースは、サービス・プロバイダーでは使用できず、サービス・プロバイダー用に作成された PIPELINE リソースもサービス・リクエスターでは使用できません。2 種類の PIPELINE は、CONFIGFILE 属性に指定されているパイプライン構成ファイルの内容によって区別されます。サービス・プロバイダーでは、最上位の要素が <provider_pipeline> で、サービス・リクエスターでは <requester_pipeline> です。

WEBSERVICE

WEBSERVICE リソース定義は、アプリケーション・データ構造と SOAP メッセージの間のマッピングが CICS Web サービス・アシスタントを使用して生成されている場合にのみ必要です。このリソース定義では、配置される CICS アプリケーション・プログラムの実行時環境の性質を Web サービスの設定で定義します。

CICS は WEBSERVICE リソースの通常のリソース定義メカニズムを備えています。一般にこのリソースは PIPELINE のピックアップ・ディレクトリーがスキャンされると、Web サービス・バインディング・ファイルから自動的に作成されます。これは、PIPELINE リソースがインストールされたとき、あるいは PERFORM PIPELINE SCAN コマンドの結果として行われます。この場合に WEBSERVICE リソースに適用される属性は、Web サービス・アシスタントによって作成された Web サービス・バインディン

グ・ファイルから取得されます。バインディング・ファイル内の情報は、Web サービス記述から取得されるか、または Web サービス・アシスタントのパラメーターとして提供されます。

サービス・リクエスター用に作成された **WEBSERVICE** リソースは、サービス・プロバイダーでは使用できず、サービス・プロバイダー用に作成された **WEBSERVICE** リソースもサービス・リクエスターでは使用できません。2 種類の **WEBSERVICE** は、**PROGRAM** 属性によって区別されます。サービス・プロバイダーでは、属性の指定が必要です。サービス・リクエスターでは属性を省略する必要があります。

URIMAP

要求を処理する他のリソース (**PIPELINE** など) にインバウンド Web サービス要求の **URI** をマップする情報が **URIMAP** 定義に含まれる場合、この定義はサービス・プロバイダーで必要になります。

URIMAP はサービス要求元のユーザー ID 情報が **HTTP** 認証ヘッダーに組み込まれてサービス・プロバイダーに渡されるように指定しているため、**URIMAP** 定義は、**HTTP** 基本認証を使用する場合にも必要となります。

CICS は通常のリソース定義メカニズムを備えていますが、**CICS Web** サービス・アシスタントを使用して配置されたサービス・プロバイダーでは、**URIMAP** リソースは通常の場合、**PIPELINE** のピックアップ・ディレクトリーがスキャンされると、Web サービス・バインディング・ファイルから自動的に作成されます。このスキャンは、**PIPELINE** リソースがインストールされたとき、あるいは **PERFORM PIPELINE SCAN** コマンドの結果として行われます。この場合に **URIMAP** リソースに適用される属性は、Web サービス・アシスタントによって作成された Web サービス・バインディング・ファイルから取得されます。バインディング・ファイル内の情報は、Web サービス記述から取得されるか、または Web サービス・アシスタントのパラメーターとして提供されます。

TCPIPSERVICE

TCPIPSERVICE 定義は、**HTTP** トランSPORTを使用するサービス・プロバイダーが必要です。この定義には、インバウンド要求を受信するポートに関する情報が含まれます。

特定のアプリケーション・プログラムをサポートするために必要なリソースは、以下の条件によって異なります。

- アプリケーション・プログラムがサービス・プロバイダーであるか、サービス・リクエスターであるか
- アプリケーションが **CICS Web** サービス・アシスタントを使用して配置されるかどうか

サービス・リクエスターまたはサービス・プロバイダー	CICS Web サービス・アシスタントの使用	PIPELINE が必要であるか	WEBSERVICE が必要であるか	URIMAP が必要であるか	TCPIPSERVICE が必要であるか
プロバイダー	はい	はい	はい (ただし、注 1 を参照)	はい (ただし、注 1 を参照)	注 2 を参照
	いいえ	はい	いいえ	はい	注 2 を参照

サービス・リクエスターまたはサービス・プロバイダー	CICS Web サービス・アシスタントの使用	PIPELINE が必要であるか	WEBSERVICE が必要であるか	URIMAP が必要であるか	TCPIPService が必要であるか
リクエスター	はい	はい	はい	いいえ	いいえ
	いいえ	はい	いいえ	いいえ	いいえ
注: 1. CICS Web サービス・アシスタントを使用してアプリケーション・プログラムを配置する場合、PIPELINE のピックアップ・ディレクトリーのスキャン時に WEBSERVICE リソースおよび URIMAP リソースを自動的に作成することができます。このスキャンは、PIPELINE リソースがインストールされたとき、あるいは PERFORM PIPELINE SCAN コマンドの結果として行われます。 2. TCPIPService リソースは、HTTP トランスポートを使用するときに必要です。WebSphere MQ トランスポートの使用時は、TCPIPService リソースは必要ありません。					

一般に、CICS システムに多数の Web サービス・アプリケーションを配置する場合、それぞれのタイプのリソースを複数作成することになります。この場合、アプリケーション間でいくつかのリソースを共用できます。

表 1. 各ファイルおよびリソースで使用可能なリソース

ファイルまたはリソース	リソースの数およびタイプ
パイプライン構成ファイル	このファイルを参照する複数の PIPELINE リソース。
PIPELINE リソース	- PIPELINE を参照する複数の URIMAP リソース。 - PIPELINE を参照する複数の WEBSERVICE リソース。 - PIPELINE のピックアップ・ディレクトリー内の複数の Web サービス・バインディング・ファイル。
Web サービス・バインディング・ファイル	- バインディング・ファイルから自動的に生成される 1 つの URIMAP リソースのみ。ただし、RDO を使用してさらに複数の URIMAP を定義することができます。 - バインディング・ファイルから自動的に生成される 1 つの WEBSERVICE リソースのみ。ただし、RDO を使用してさらに複数の WEBSERVICE を定義することができます。
WEBSERVICE	複数の URIMAP リソース。WEBSERVICE リソースがバインディング・ファイルから自動的に生成される場合は、これに対応する URIMAP リソースが 1 つだけあります。ただし、RDO を使用してさらに複数の URIMAP リソースを定義することができます。
URIMAP	URIMAP リソースで明示的に指定される場合は、1 つの TCPIPService のみ

表 1. 各ファイルおよびリソースで使用可能なリソース (続き)

ファイルまたはリソース	リソースの数およびタイプ
TCPIPService	• 多数の URIMAP リソース

WebSphere MQ トランSPORTを使用するための CICS の構成

CICS で WebSphere MQ トランSPORTを Web サービスと組み合わせて使用するには、それに応じて CICS 領域を構成する必要があります。

1. 以下の WebSphere MQ ライブラリーを CICS プロシーチャーの STEPLIB 連結に組み込みます。正しいコードが確実に使われるようにするために、CICS ライブラリーの後にこのライブラリーを組み込んでください。

`thlqual.SCSQAUTH`

ここで、

`thlqual` は、WMQ ライブラリーの高位修飾子です。

`x` は、各国語の言語を表す文字です。

2. 以下の WebSphere MQ ライブラリーを CICS プロシーチャーの DFHRPL 連結に組み込みます。ここでも、正しいコードが確実に使われるようにするために、CICS ライブラリーの後にこれらのライブラリーを組み込んでください。

`thlqual.SCSQCICS`

`thlqual.SCSQLOAD`

`thlqual.SCSQAUTH`

ここで、

`thlqual` は、WMQ ライブラリーの高位修飾子です。

`x` は、各国語の言語を表す文字です。

CICS-WebSphere MQ API 交差出口 (CSQCAPX) を使用している場合は、プログラムのロード・モジュールを含んでいるライブラリーの名前も追加してください。SCSQCICS ライブラリーは、WebSphere MQ に付属のサンプルを実行する場合に限って必要です。そうでない場合は、CICS プロシーチャーからこれを除去することができます。

3. CICS 領域用の MQCONN リソース定義をインストールします。MQCONN リソース定義は、CICS と WebSphere MQ との接続の属性を指定します。これには、接続のデフォルト WebSphere MQ キュー・マネージャーまたはキュー共有グループの名前が含まれます。これを行う方法については、CICS インフォメーション・センターの『CICS と MQ の統合』セクションの『MQCONN リソース定義のセットアップ』を参照してください。
4. CICS の初期化で CICS と WebSphere MQ との接続を自動的に開始するには、CICS システム初期設定パラメーター **MQCONN=YES** を指定します。

CICS が WebSphere MQ への接続を開始するためには、その前に MQCONN リソース定義をインストールしておく必要があります。CICS の初期設定時に接続を自動開始する際、初期始動またはコールド・スタートの場合には、**GRPLIST** システム初期設定パラメーターで指定されたリスト内のいずれかのグループに MQCONN リソース定義が含まれる必要があります。CICS のウォーム始動また

は緊急開始の場合は、以前の CICS 実行の終了までに MQCONN リソース定義がインストール済みでなければなりません。

5. リモート CICS システムとの領域間通信 (IRC) を行う CICS システムで CICS-WebSphere MQ アダプターを使用する場合には、CICS システム初期設定パラメーター IRCSTRT=YES を指定することにより、アダプター開始前に IRC 機能を必ず OPEN してください。IRC アクセス方式が仮想記憶間として定義されている場合 (つまり ACCESSMETHOD(XM) である場合)、IRC 機能が OPEN している必要があります。
6. ご使用のキュー・マネージャーおよび CICS で使用されるコード化文字セット ID (CCSID)、および UTF-8 と UTF-16 のコード・ページが、z/OS 変換サービスに対して構成されていることを確認します。CICS コード・ページは、**LOCALCCSID** システム初期化パラメーターで指定されます。
7. 次のようにして CICS CSD を更新します。
 - a. CSD を以前のリリースの CICS と共用しない場合は、グループ CSQCAT1 および CSQCKB を CSD から削除します。また、CKQQ TDQUEUE をグループ CSQCAT1 から削除する必要もあります。これで、CKQQ の定義が CICS CSD グループ DFHDCTG で提供されるようになります。
 - b. 以前の CICS リリースとの間で CSD を共有する場合には、CSQCAT1 および CSQCKB が CICS TS 4.1 または CICS TS 3.2 用にインストールされていないことを確認してください。また、CKQQ TDQUEUE をグループ CSQCAT1 から削除する必要もあります。これで、CKQQ の定義が CICS CSD グループ DFHDCTG で提供されるようになります。CICS TS 3.2 より前の CICS TS リリースの場合、グループ DFHMQ を指定変更して必要な定義を正しくインストールするために、DFHLIST のインストール後に CSQCAT1 および CSQCKB グループをグループ・リストに含めてインストールしてください。
8. デッド・レター・キュー、デフォルト伝送キュー、および CICS-WebSphere MQ アダプター・オブジェクトに関する WebSphere MQ 定義を更新します。サンプル CSQ4INYG を使用できますが、CICS 領域の MQINI リソース定義内のデフォルト開始キュー名に一致するよう、開始キュー名を変更する必要があるかもしれません。このメンバーをキュー・マネージャー始動プロシーチャーの CSQINP2 DD 連結で使用できます。または、必要な DEFINE コマンドを発行するために、CSQUTIL ユーティリティの COMMAND 関数の入力としてこのメンバーを使用することもできます。CSQUTIL ユーティリティを使用すると、それ以降 WebSphere MQ を再始動するたびにこれらのオブジェクトを再定義する必要がなくなるため、この方法をお勧めします。

WebSphere MQ トランスポート

CICS は、WMQ トランスポートを使用して、サービス・プロバイダーとサービス・リクエスターの両方の役割で、WebSphereMQ (WMQ) との間で SOAP メッセージを送受信できます。

サービス・プロバイダーとして、CICS は WMQ トリガーを使用して、SOAP メッセージをアプリケーション・キューから処理します。トリガーは、開始キューとローカル・キューを使用することで動作します。ローカル (アプリケーション) キュー定義には、以下が含まれます。

- トリガー・メッセージが生成される時期についての基準。例えば、最初のメッセージがローカル・キューに到着したとき、またはローカル・キューに到着するメッセージごとに、など。CICS SOAP 処理については、最初のメッセージがローカル・キューに到着したときにトリガーが起こるように指定するとよいでしょう。

ローカル・キュー定義では、トリガー・データをターゲット・アプリケーションに渡すように指定することもできます。CICS SOAP 処理 (トランザクション CPIL) の場合は、インバウンド・メッセージで渡されない場合に使用されるデフォルトのターゲット URL を指定します。

- プロセス定義 を識別するプロセス名。プロセス定義では、メッセージを処理する方法が記述されます。CICS SOAP 処理の場合は、CPIL トランザクションが指定されます。
- トリガー・メッセージが送信される開始キューの名前。

メッセージがローカル・キューに到着すると、キュー・マネージャーがトリガー・メッセージを生成し、指定された開始キューに送信します。トリガー・メッセージには、プロセス定義からの情報が含まれます。トリガー・モニターは開始キューからトリガー・メッセージを取り出し、ローカル・キューでメッセージの処理を開始するように CPIL トランザクションをスケジュールします。トリガーの詳細については、 を参照してください。インフォメーション・センターのトピック *CICS integration with WebSphere MQ* を参照してください。

CICS を構成することで、メッセージがローカル・キューに到着すると、トリガー・モニター (WMQ 提供) が CPIL トランザクションをスケジュールして、ローカル・キューのメッセージを処理し、CICS SOAP パイプラインにキューの SOAP メッセージを処理させることができます。

CICS が Websphere MQ から受け取る SOAP メッセージへの応答を構成するとき、レポート・オプション MQRO_PASS_CORREL_ID が設定されていない限り、入力メッセージのメッセージ ID で相関 ID フィールドが設定されます。このレポート・オプションが設定されていれば、相関 ID は入力メッセージから応答に伝搬されます。

サービス・リクエスターとして、アウトバウンド要求で、ターゲット Web サービスへの応答が特定の応答キューで戻るように指定できます。

どちらの場合も、CICS および WMQ では、必要なリソースおよびキューを定義するための構成が必要です。

サービス・プロバイダーでのローカル・キューの定義

サービス・プロバイダーで WebSphere MQ トランSPORTを使用する場合は、要求メッセージを処理するまで要求メッセージを保管する 1 つ以上のローカル・キューと、要求メッセージを処理する CICS トランザクションを指定する 1 つのトリガー・プロセスを定義する必要があります。

1. 開始キューを定義します。以下のコマンドを使用します。


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

ここで *initiation_queue* は、CICS 領域の MQINI リソース定義の INITQNAME 属性に指定されている値と同じです。MQINI は、MQCONN リソース定義のインストール時に CICS によってインストールされる暗黙的なリソースです。開始キュー名を照会するには、EXEC CICS または CEMT INQUIRE MQINI コマンドを使用してください。

2. ローカル要求キューごとに、QLOCAL オブジェクトを定義します。以下のコマンドを使用します。

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

ここで、

- *queueename* は、ローカル・キュー名です。
 - *processname* は、トリガー・イベント発生時にキュー・マネージャーによって開始されるアプリケーションを示すプロセス・インスタンスの名前です。各 QLOCAL オブジェクトにも、同じ名前を指定します。
 - *initiation_queue* は使用される開始キューの名前です (例えば CICS 領域の MQINI 定義で指定されている開始キュー)。
 - *default_target_service* は、要求にサービスが指定されていない場合、使用されるデフォルトのターゲット・サービスです。ターゲット・サービスの形式は「/string」で、これは URIMAP 定義のパスと突き合わせるために使用されます (例えば /SOAP/test/test1)。先頭文字は必ず「/」にする必要があります。
 - *nnn* は、行われる再試行の回数です。
 - *requeueename* は、障害発生メッセージの送信先キューの名前です。
3. トリガー・プロセスを指定する PROCESS オブジェクトを定義します。以下のコマンドを使用します。

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

ここで、

processname は、プロセスの名前で、要求キューの定義時に使用される名前と同じにする必要があります。

サービス・リクエスターでのローカル・キューの定義

サービス・リクエスターで、アウトバウンド要求に対して WebSphere MQ トランスポートを使用する場合は、事前定義された応答キューに応答が戻ることをターゲット Web サービスの URI で指定できます。 こうする場合は、QLOCAL オブジェクトで各応答キューを定義する必要があります。

このタスクについて

要求に関連した URI が応答キューを指定していない場合、CICS は応答に動的キューを使用します。

オプション: 事前定義の応答キューを指定する各 QLOCAL オブジェクトを定義するには、以下のコマンドを使用します。

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

ここで、

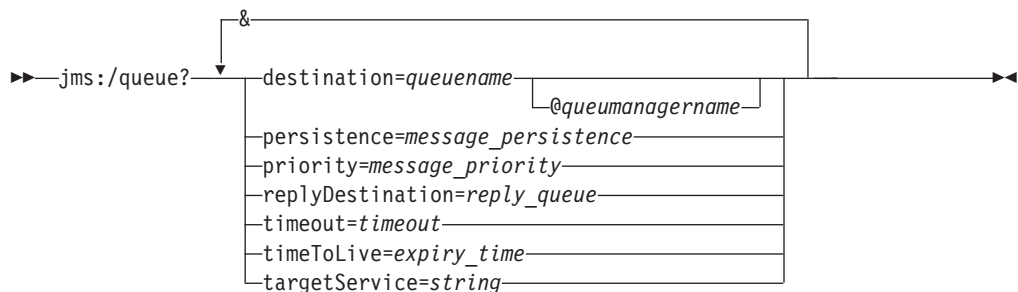
reply_queue は、ローカル・キュー名です。

nnn は、行われる再試行の回数です。

WMQ トランスポートの URI

サービス・リクエスターとサービス・プロバイダーの間の通信に WMQ を使用するとき、宛先の URI は、宛先をキューとして識別する形式であり、WMQ が要求と応答を処理する方法を指定するための情報を含んでいます。

構文



CICS は以下のオプションを使用します。他の Web サービス・プロバイダーは、ここで説明しない追加のオプションを使用することがあります。URI 全体がサービス・プロバイダーに渡されます。ただし、CICS は、CICS がサポートしないオプションおよび URI でコーディングされているオプションを無視します。CICS はオプション名の大/小文字を区別しません。ただし、このスタイルの URI をサポートするその他の実装環境には、大/小文字を区別するものがあります。

destination=queueename [*@queueumanagername*]

queueename は宛先キュー・マネージャーの入力キューの名前です。

`queuemanagername` は宛先キュー・マネージャーの名前です。

persistence=message_persistence

以下のいずれかを指定します。

- 0 永続性はデフォルトのキューの永続性によって定義されます。
- 1 メッセージは永続ではありません。
- 2 メッセージは永続です。

このオプションが指定されないか、誤って指定されると、デフォルトのキューの永続性が使用されます。

priority=message_priority

メッセージ優先順位を、0 から 99999999 の範囲の整数で指定します。

replyDestination=reply_queue

応答メッセージに使用されるキューを指定します。このオプションが指定されていない場合、CICS は応答メッセージに動的キューを使用します。このオプションを使用する前に QLOCAL オブジェクトに応答キューを定義する必要があります。

timeout=timeout

サービス・リクエスターが応答を待つ、ミリ秒単位のタイムアウトです。値ゼロが指定されるか、このオプションが除外される場合は、要求はタイムアウトになりません。

timeToLive=expiry-time

要求の有効期限時刻をミリ秒単位で指定します。このオプションが指定されないか、誤って指定されると、要求の有効期限が切れません。

targetService=string

ターゲット・サービスを識別します。CICS がサービス・プロバイダーである場合は、ターゲット・サービスの形式は `'/string'` になります。これは、URIMAP との突き合わせを試みる際に、CICS がこれをパスとして使用するためです。指定されない場合は、サービス・プロバイダーで入力キューの TRIGDATA に指定された値が使用されます。

この例は WMQ トランスポートの URI を示しています。

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

永続メッセージをサポートするための CICS の構成

CICS は、WMQ トランスポート・プロトコルを使用した永続メッセージの、CICS 領域に配置された Web サービス・プロバイダー・アプリケーションへの送信をサポートしています。

このタスクについて

CICS は、ビジネス・トランザクション・サービス (BTS) を使用して、CICS システム障害の際に永続メッセージが確実に回復されるようにします。これを正しく機能させるためには、以下のステップに従います。

1. IDCAMS (データ操作ユーティリティ) を使用して、MVS へのローカルの要求キューとリポジトリ・ファイルを定義します。ファイル定義のために、STRINGS に適切な値を指定する必要があります。デフォルト値 1 では十分ではないと思われるため、10 を使用することをお勧めします。
2. CICS に対するローカルの要求キューとリポジトリ・ファイルを定義します。CICS に対するローカルの要求キューを定義する方法の詳細は、60 ページの『サービス・プロバイダーでのローカル・キューの定義』で説明しています。ファイル定義に、STRINGS に適切な値を指定する必要があります。デフォルト値 1 では十分ではないと思われるため、10 を使用することをお勧めします。
3. リポジトリ・ファイル名を FILE オプションの値として使用して、DFHMQSOA という名前の PROCESSTYPE リソースを定義します。
4. 永続メッセージの処理中に、最初の暗黙的な同期点が要求される前にプログラムが EXEC CICS SYNCPOINT コマンドを必ず発行することを確認してください。例えば EXEC CICS CREATE TDQUEUE のような SPI コマンドを使用すると、暗黙的に同期点が取られます。EXEC CICS SYNCPOINT コマンドを発行すると、永続メッセージが正常に処理されたことを確認できます。プログラムが暗黙的に同期点を取ることを試みる前に明示的に同期点を要求しない場合、CICS は ASP7 異常終了を発行します。

タスクの結果

次のタスク

片方向の要求メッセージの場合は、Web サービスが異常終了またはバックアウトすると、トランザクションまたはプログラムが障害が発生している要求を再試行したり障害を適切に報告したりするための十分な情報が保持されます。このようなりカバリー・トランザクションまたはプログラムを提供する必要があります。詳しくは、『永続メッセージの処理』を参照してください。

永続メッセージの処理

Web サービス要求が WMQ 永続メッセージで受信されると、CICS は、プロセス・タイプが DFHMQSOA である固有の BTS プロセスを作成します。インバウンド要求に関連するデータは、プロセスに関連付けられた BTS データ・コンテナ内に取り込まれます。

プロセスは、非同期で実行されるようにスケジュールに入れます。Web サービスが正常に完了してコミットすると、CICS は BTS プロセスを削除します。これには、SOAP 障害が Web サービス・リクエスターに生成され戻される場合が含まれます。

エラー処理

必要な BTS プロセスの作成時にエラーが発生すると、Web サービス・トランザクションは異常終了し、インバウンド Web サービス要求は処理されません。BTS が使用不可の場合は、メッセージ DFHPI0117 が発行され、CICS は、既存のチャネル・ベースのコンテナ・メカニズムを使用して、BTS なしで続行します。

Web サービスが開始または処理を完了する前に CICS 障害が発生すると、BTS リカバリーにより、CICS の再起動時にプロセスのスケジュールが変更されます。

Web サービスが異常終了しバックアウトすると、BTS プロセスは、ABENDED 状態で完了したというマークが付けられます。応答を必要とする要求メッセージの場合は、SOAP 障害が Web サービス・リクエスターに戻されます。BTS プロセスは取り消され、CICS は、失敗した要求に関する情報を保持しません。CICS は、一時データ・キュー CSBA ではメッセージ DFHBA0104 を発行し、一時データ・キュー CPIO ではメッセージ DFHPI0117 を発行します。

片方向メッセージの場合、障害に関する情報をリクエスターに戻す方法はないため、BTS プロセスは COMPLETE ABENDED 状態を保ちます。CICS は、一時データ・キュー CSBA ではメッセージ DFHBA0104 を発行し、一時データ・キュー CPIO では DFHPI0116 を発行します。

CBAM トランザクションを使用して COMPLETE ABENDED プロセスを表示することができます。または、リカバリー・トランザクションを指定して、DFHMQSOA の COMPLETE ABENDED プロセスをチェックし、適切な処置をとることができます。

例えば、リカバリー・トランザクションで以下のことが可能です。

1. RESET ACQPROCESS コマンドを使用して BTS プロセスをリセットする。
2. RUN ASYNC コマンドを発行して、障害がある Web サービスを再試行する。
プロセスでの別のデータ・コンテナに再試行カウントを保持して、障害が繰り返されるのを回避することができます。
3. 関連する以下のデータ・コンテナ内の情報を使用して、問題を報告する。

DFHMQORIGINALMSG データ・コンテナには、WMQ から受信したメッセージが含まれ、これには RFH2 ヘッダーが含まれている場合があります。

DFHMQMSG データ・コンテナには、RFH2 ヘッダーが除去された WMQ メッセージが含まれます。

DFHMQDLQ データ・コンテナには、元のメッセージに関連付けられた送達不能キューの名前が含まれます。

DFHMQCONT データ・コンテナには、元のメッセージの MQ GET に関連する WMQ MQMD 制御ブロックが含まれます。

第 7 章 Web サービス・インフラストラクチャーの作成

Web サービスを CICS に配置するには、必要なトランスポート・インフラストラクチャーを作成して、Web サービス要求を処理する 1 つ以上のパイプラインを定義する必要があります。通常は 1 つのパイプラインが多数の異なる Web サービスの要求を処理できるため、CICS システムに新規の Web サービスを配置した場合は、既存のパイプラインを使用できます。

サービス・プロバイダーに応じた CICS インフラストラクチャーの作成

サービス・プロバイダーに応じた CICS インフラストラクチャーを作成するには、パイプライン構成ファイルを作成し、多数の CICS リソースを定義してインストールする必要があります。

このタスクについて

ご使用のサービス・プロバイダーに応じたインフラストラクチャーを作成するには、以下のステップを実行します。

1. トランスポート・インフラストラクチャーを定義します。
 - WMQ トランスポートを使用している場合は、入力メッセージを処理するまで入力メッセージを保管する 1 つ以上のローカル・キューと、入力メッセージを処理する CICS トランザクションを指定する 1 つのトリガー・プロセスを定義する必要があります。詳しくは、58 ページの『WebSphere MQ トランスポートを使用するための CICS の構成』を参照してください。
 - HTTP トランスポートを使用している場合は、インバウンド要求を受信するポートを定義する TCPIP SERVICE リソースを定義する必要があります。詳しくは、55 ページの『Web サービスのための CICS リソース』を参照してください。

必要な各種のトランスポート構成ごとにこのステップを繰り返します。

2. パイプライン構成ファイルを作成します。これは、z/OS UNIX® システム・サービスのファイル・システムに保管される XML ファイルです。このファイルは、インバウンド Web サービス要求および応答を処理するときに使用されるメッセージ・ハンドラー・プログラムを定義します。CICS は、パイプラインで各種オプションを使用可能にするために使用できるメッセージ・ハンドラーの標準セットを備えています。基本的なパイプラインのサンプル `basicoap11provider.xml` は、ライブラリー `/usr/lpp/cicsts/cicsts41/samples/pipelines` にあります。このサンプルは、必要に応じて別のメッセージ・ハンドラーに追加するための基礎として使用することができます。
 - a. パイプライン構成ファイルに含めるメッセージ・ハンドラーを定義します。カスタムのメッセージ・ハンドラー・プログラムを作成する場合にパフォーマンスを最適化するには、プログラムをスレッド・セーフにすることをお勧めします。パイプラインで使用可能にできるオプションについて詳しくは、70 ページの『パイプライン構成ファイル』を参照してください。

- b. パイプライン構成ファイルを z/OS UNIX 内の適切なディレクトリーにコピーします。
- c. パイプライン構成ファイルの権限を変更して、CICS 領域でファイルを読み取ることができるようにします。

必要な各種のパイプライン構成ごとにこのステップを繰り返します。

3. PIPELINE リソースを定義してインストールします。PIPELINE リソースは、パイプライン構成ファイルの場所を定義します。また、Web サービス・バインディング・ファイルと WSDL (オプション) が格納される z/OS UNIX ディレクトリーである、ピックアップ・ディレクトリー を指定します。

必要なパイプライン構成ごとにこのステップを繰り返します。

4. PROGRAM 定義を自動インストールした場合を除き、パイプラインで実行するプログラムごとに PROGRAM リソース定義を提供する必要があります。このようなプログラムには、通常はトランザクション CPIH の下で実行するターゲット・アプリケーション・プログラムがあります。トランザクションは、属性 TASKDATALOC(ANY) で定義されます。したがって、プログラムをリンク・エディットする際は、AMODE(31) オプションを指定する必要があります。

タスクの結果

これで、CICS システムには、各サービス・プロバイダーに必要なインフラストラクチャーが格納されるようになりました。

- 1 つ以上のトランスポート・インフラストラクチャー
- 1 つ以上のパイプライン

次のタスク

追加のトランスポート・インフラストラクチャーを定義するか、または追加のパイプラインを作成する必要がある場合は、構成を拡張できます。

サービス・リクエスターに応じた CICS インフラストラクチャーの作成

サービス・リクエスターに応じた CICS インフラストラクチャーを作成するには、パイプライン構成ファイルを作成し、多数の CICS リソースを定義してインストールする必要があります。

このタスクについて

ご使用のサービス・リクエスターに応じた CICS インフラストラクチャーを作成するには、以下のステップを実行します。

1. パイプライン構成ファイルを作成します。これは、z/OS UNIX システム・サービスのファイル・システムに保管される XML ファイルです。このファイルは、アウトバウンド Web サービス要求および応答を処理するときに使用されるメッセージ・ハンドラー・プログラムとヘッダー処理プログラムを定義します。

CICS は、パイプラインで各種オプション (SOAP 1.1 または SOAP 1.2 メッセージの送信など) を使用可能にするために使用できるメッセージ・ハンドラーとヘッダー処理プログラムの標準セットを備えています。基本的なパイプラインのサンプル basicsoap11requester.xml は、ライブラリー /usr/lpp/cicsts/samples/

pipelines にあります。このサンプルは、必要に応じて別のメッセージ・ハンドラーに追加するための基礎として使用することができます。

- a. CICS 提供のメッセージ・ハンドラーが処理要件を満たしているかどうかを検討します。CICS には次のハンドラーとヘッダー・プログラムがあります。
 - SOAP メッセージ・ハンドラー。SOAP 1.1 または 1.2 のメッセージを処理します。サービス・リクエスター・パイプラインの 1 つのレベルの SOAP だけサポートできます。
 - MTOM ハンドラー。MTOM/XOP 仕様に準拠する MIME Multipart/Related メッセージを処理します。
 - セキュリティー・ハンドラー。セキュア Web サービス・メッセージを処理します。
 - WS-AT ヘッダー処理プログラム。アトミック・トランザクション・メッセージを処理します。
- b. パイプライン構成ファイルに含めるメッセージ・ハンドラーを定義します。パイプラインで使用可能にできるオプションについて詳しくは、70 ページの『パイプライン構成ファイル』を参照してください。

パイプラインで独自の処理を実行したい場合は、メッセージ・ハンドラーまたはヘッダー処理プログラムを作成する必要があります。詳しくは、109 ページの『メッセージ・ハンドラー』を参照してください。カスタムのメッセージ・ハンドラー・プログラムを作成することにした場合は、パフォーマンスを最適化するには、プログラムをスレッド・セーフにすることをお勧めします。

- c. パイプライン構成ファイルを z/OS UNIX 内の適切なディレクトリーにコピーします。
 - d. パイプライン構成ファイルの権限を変更して、CICS 領域でファイルを読み取ることができるようにします。
2. PIPELINE リソースを定義してインストールします。PIPELINE リソースは、パイプライン構成ファイルの場所を定義します。また、Web サービス・バインディング・ファイルと WSDL (オプション) が格納される z/OS UNIX ディレクトリーである、ピックアップ・ディレクトリーを指定します。リクエスター・モード・パイプラインでは、タイムアウトを秒単位で指定することもできます。これは、CICS が Web サービス・プロバイダーからの応答を待機する期間です。必要な各種のパイプライン構成ごとにこのステップを繰り返します。PIPELINE リソースをインストールすると、CICS は、指定したピックアップ・ディレクトリーに格納されているファイルを読み取り、WEBSERVICE リソースを動的に作成します。
 3. PROGRAM 定義を自動インストールした場合を除き、パイプラインで実行するプログラムごとに PROGRAM リソース定義を提供する必要があります。このようなプログラムには、通常はトランザクション CPIH の下で実行するサービス・リクエスター・アプリケーション・プログラムがあります。トランザクションは、属性 TASKDATALOC(ANY) で定義されます。したがって、プログラムをリンク・エディットする際は、AMODE(31) オプションを指定する必要があります。

タスクの結果

これで、CICS システムには、各サービス・リクエスターに必要なインフラストラクチャーが格納されるようになりました。

次のタスク

追加のパイプラインを作成する必要がある場合は、構成を拡張できます。

パイプライン構成ファイル

Web サービス要求を処理するときに使用する、パイプライン構成ファイル と呼ばれるパイプラインの構成は、XML 文書で指定します。

パイプライン構成ファイルは、z/OS UNIX システム・サービスのファイル・システムに保管されます。このファイルの名前は、PIPELINE リソース定義の CONFIGFILE 属性で指定します。パイプライン構成ファイルを使用する場合は、適切な XML エディターまたはテキスト・エディターを使用してください。パイプライン構成ファイルの XML スキーマは、z/OS UNIX 上の /usr/lpp/cicsts/cicsts41/schemas/pipeline/ ディレクトリにあります。構成ファイルを使用する場合は、文字セットのエンコード方式が US EBCDIC (コード・ページ 037) であることを確認してください。

CICS は、Web サービス要求を処理する場合、1 つ以上のメッセージ・ハンドラーからなるパイプラインを使用して Web サービス要求を処理します。パイプラインは、Web Service Security、および Web Service トランザクションのサポートなどの、アプリケーションの異なるカテゴリーに適用する実行環境の局面を提供するために構成されます。一般に、多数のサービス・プロバイダー・アプリケーションまたはサービス・リクエスター・アプリケーションが存在する CICS 領域には、いくつかの異なるパイプライン構成が必要になります。ただし、異なるアプリケーションの要件が類似する場合、これらのアプリケーションが同じパイプライン構成を共有することが可能です。

パイプライン構成は、2 種類あります。1 つはサービス・プロバイダー・パイプラインを記述し、もう 1 つはサービス・リクエスター・パイプラインを記述します。それぞれが独自のスキーマによって定義され、異なるルート・エレメントを持っています。

パイプライン	スキーマ	ルート・エレメント
サービス・プロバイダー	Provider.xsd	<provider_pipeline>
サービス・リクエスター	Requester.xsd	<requester_pipeline>

使用される XML エレメントの多くは両方のパイプライン構成に共通ですが、片方にのみ使用されるエレメントもあるため、プロバイダーとリクエスターの両方に同じ構成ファイルを使用することはできません。

制約事項: パイプライン構成ファイルでは、ネーム・スペースで修飾されたエレメント名はサポートされません。

<provider_pipeline> および <requester_pipeline> エレメントの隣接したサブエレメントは、次のとおりです。

- `<service>` エレメント。これは、すべての要求に対して呼び出されるメッセージ・ハンドラーを指定します。このエレメントは、`<provider_pipeline>` エレメント内で使用する際は必須で、`<requester_pipeline>` エレメント内ではオプションです。
- オプションの `<transport>` エレメント。これは、メッセージ・トランスポートに使用されるリソースに基づいて、実行時に選択されるメッセージ・ハンドラーを指定します。
- `<provider_pipeline>` のみの場合は、`<apphandler>` エレメント。これは、一部の例ではサービスを提供するターゲット・アプリケーション（またはラッパー・プログラム）の指定に使用されます。
- オプションの `<service_parameter_list>` エレメント。パイプライン内のメッセージ・ハンドラーで使用可能なパラメーターを含みます。

一部のエレメントは、そのエレメントに関連付けられる属性を持つことがあります。有効な XML 文書を作成するには、各属性値を引用符で囲む必要があります。

パイプライン構成ファイルに関連しているのは、PIPELINE リソースです。この属性には、z/OS UNIX にあるパイプライン構成ファイルの名前を指定する CONFIGFILE があります。PIPELINE 定義をインストールすると、CICS は、パイプラインを構成するために必要な情報をこのファイルから読み取ります。

CICS は、独自の構成ファイルを作成するための基本として使用できる構成ファイルのサンプルを提供します。これらのファイルは、ライブラリー /usr/lpp/cicts/samples/pipelines にあります。

ファイル

説明

basicsoap11provider.xml

CICS 提供の SOAP 1.1 ハンドラーを使用するサービス・プロバイダーのパイプライン定義。アプリケーションが CICS Web サービス・アシスタントを使用して配置されているときに使用されます。

basicsoap11requester.xml

CICS 提供の SOAP 1.1 ハンドラーを使用するサービス・リクエスターのパイプライン定義。アプリケーションが CICS Web サービス・アシスタントを使用して配置されているときに使用されます。

wsatprovider.xml

Web サービス・トランザクションについての構成情報を basicsoap11provider.xml に追加するパイプライン定義。

wsatrequester.xml

Web サービス・トランザクションについての構成情報を basicsoap11requester.xml に追加するパイプライン定義。

パイプライン構成ファイルの例

以下に、サービス・プロバイダー・パイプライン用の、単純な構成ファイルの例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<provider_pipeline
  xmlns="http://www.ibm.com/software/http/cics/pipeline"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/software/http/cics/pipeline/provider.xsd">
<service>
  <terminal_handler>
    <cics_soap_1.1_handler/>
  </terminal_handler>
</service>
<apphandler>DFHPITP</apphandler>
</provider_pipeline>

```

このパイプラインに含まれるメッセージ・ハンドラーは、CICS 提供の SOAP 1.1 メッセージ・ハンドラー 1 つだけです。このハンドラーはプログラム DFHPITP にリンクします。

- **<provider_pipeline>** エレメントは、サービス・プロバイダー・パイプライン用のパイプライン構成ファイルのルート・エレメントです。
- **<service>** エレメントは、すべての要求に対して呼び出されるメッセージ・ハンドラーを指定します。この例では、メッセージ・ハンドラーは 1 つしかありません。
- **<terminal_handler>** エレメントは、パイプラインの端末メッセージ・ハンドラーの定義を含みます。
- **<cics_soap_1.1_handler>** は、パイプラインの端末ハンドラーが、SOAP 1.1 メッセージについての CICS 提供のハンドラー・プログラムであることを示します。
- **<apphandler>** エレメントは、パイプラインの端末ハンドラーがデフォルトでリンクするプログラムの名前を指定します。このケースでは、プログラムは DFHPITP で、CICS Web サービス・アシスタントを使用して配置されたアプリケーション用の、CICS 提供のターゲット・プログラムです。プログラムが Web サービス・アシスタントを使用して配置されていない場合は、ターゲット・アプリケーション・プログラムの名前になります。

トランスポート関連ハンドラー

各パイプラインの構成ファイルに、複数のメッセージ・ハンドラーの集合を指定できます。CICS は、実行時に、メッセージのトランスポートに使用されているリソースに基づいて、呼び出されるメッセージ・ハンドラーを選択します。

サービス・プロバイダー、およびサービス・リクエスターで、特定のトランスポート (HTTP または WebSphere MQ) が使用中の場合にのみいくつかのメッセージ・ハンドラーを呼び出すように指定することができます。例えば、従業員に対して使用可能にする Web サービスを考えてみます。会社で働く従業員は、保護された社内ネットワーク上で WebSphere MQ トランスポートを使用してサービスにアクセスします。しかし、ビジネス・パートナーの場所で作業する従業員はインターネットを介して HTTP トランスポートを使用してサービスにアクセスします。このような状況では、情報の機密性という性質から、HTTP トランスポートを使用するときはメッセージ・ハンドラーを使用して、メッセージの一部を暗号化することが必要になります。

サービス・プロバイダーでは、インバウンド・メッセージは、指定されたリソース (HTTP トランスポートの場合は TCPIP SERVICE、MQ トランスポートの場合は QUEUE) と関連付けられます。特定のリソースがインバウンド要求について使用される場合のみ、いくつかのメッセージ・ハンドラーを呼び出すように指定できます。

これを可能にするには、パイプライン構成ファイルの次の 2 つの別個の部分にメッセージ・ハンドラーを指定します。

サービス・セクション

パイプラインを実行するたびに呼び出されるメッセージ・ハンドラーを指定します。

トランスポート・セクション

使用中のトランスポート・リソースに応じて呼び出される、または呼び出されない、メッセージ・ハンドラーを指定します。

要確認: 実行時に、メッセージ・ハンドラーがパイプラインの実行を省略することを選択できます。したがって、CICS がパイプライン構成ファイルの内容に基づいて特定のメッセージ・ハンドラーを呼び出すように決定している場合でも、これより前のメッセージ・ハンドラーによってこの決定を無効にできます。

トランスポート・セクションに指定されたメッセージ・ハンドラー (トランスポート関連ハンドラー) は、いくつかのリストにまとめられます。CICS は、実行時に、使用されているトランスポート・リソースに基づいて、これらのうちの 1 つだけのリストから実行するハンドラーを選択します。使用されているトランスポート・リソースに対応するリストが複数ある場合、CICS は最も選択的なリストを使用します。以下に、サービス・プロバイダーとサービス・リクエスターの両方のパイプラインで使用されるリストを示します。

<default_transport_handler_list>

これはトランスポート関連ハンドラーのリストの中で最も選択的でないリストです。このリストに指定されているハンドラーは、以下に示すどのリストも使用されているトランスポート・リソースに対応しない場合に呼び出されます。

<default_http_transport_handler_list>

サービス・リクエスター・パイプラインでは、HTTP トランスポートが使用されているとき、このリストにあるハンドラーが呼び出されます。

サービス・プロバイダー・パイプラインでは、HTTP トランスポートが使用されており、<named_transport_entry> に TCP/IP 接続について TCPIPService が指定されていないときに、このリストにあるハンドラーが呼び出されます。

<default_mq_transport_handler_list>

サービス・リクエスター・パイプラインでは、WebSphere MQ トランスポートが使用されているとき、このリストにあるハンドラーが呼び出されます。

サービス・プロバイダー・パイプラインでは、WebSphere MQ トランスポートが使用されており、<named_transport_entry> にインバウンド・メッセージを受信するメッセージ・キューが指定されていないときに、このリストにあるハンドラーが呼び出されます。

以下のメッセージ・ハンドラーのリストは、サービス・プロバイダー・パイプライン用の構成ファイルでのみ使用されます。

<named_transport_entry>

<named_transport_entry> は、ハンドラーのリストだけでなく、リソースの名前とトランスポート・タイプを指定します。

- HTTP トランスポートの場合、リソース名がインバウンド TCP/IP 接続の TCIPSERVICE の名前と一致したときに、このリストにあるハンドラーが呼び出されます。
- WebSphere MQ トランスポートの場合は、リソース名がインバウンド・メッセージを受信するメッセージ・キューの名前と一致したときに、このリストにあるハンドラーが呼び出されます。

例

以下に、サービス・プロバイダー・パイプライン用のパイプライン構成ファイルの <transport> エレメントの例を示します。

```
<transport>

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

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

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

</transport>
```

これらの定義の効果は、次のとおりです。

- <default_mq_transport_handler_list> は、MQ トランスポートを使用するメッセージがハンドラー HANDLER3 によって処理されるように設定します。
- <named_transport_entry> は、TCIPSERVICE(WS00) に関連付けられた TCP/IP 接続を使用するメッセージがハンドラー HANDLER4 によって処理されるように設定します。
- <default_transport_handler_list> は、残りのすべてのメッセージ、つまり HTTP トランスポートを使用するが TCIPSERVICE(WS00) を使用しないメッセージがハンドラー HANDLER1 と HANDLER2 によって処理されるように設定します。

要確認: トランスポート・セクションに指定されたハンドラーに加えて、パイプライン定義のサービス・セクションに指定されたハンドラーが呼び出されます。

サービス・プロバイダーのパイプライン定義

メッセージ・ハンドラーは、z/OS UNIX に格納されている XML 文書で定義されます。この文書が格納されているファイルの名前は、PIPELINE 定義の CFGFILE 属性で指定します。

パイプライン構成文書のルート・エレメントは、<provider_pipeline> エレメントです。この文書の上位構造を 76 ページの図 21 に示します。

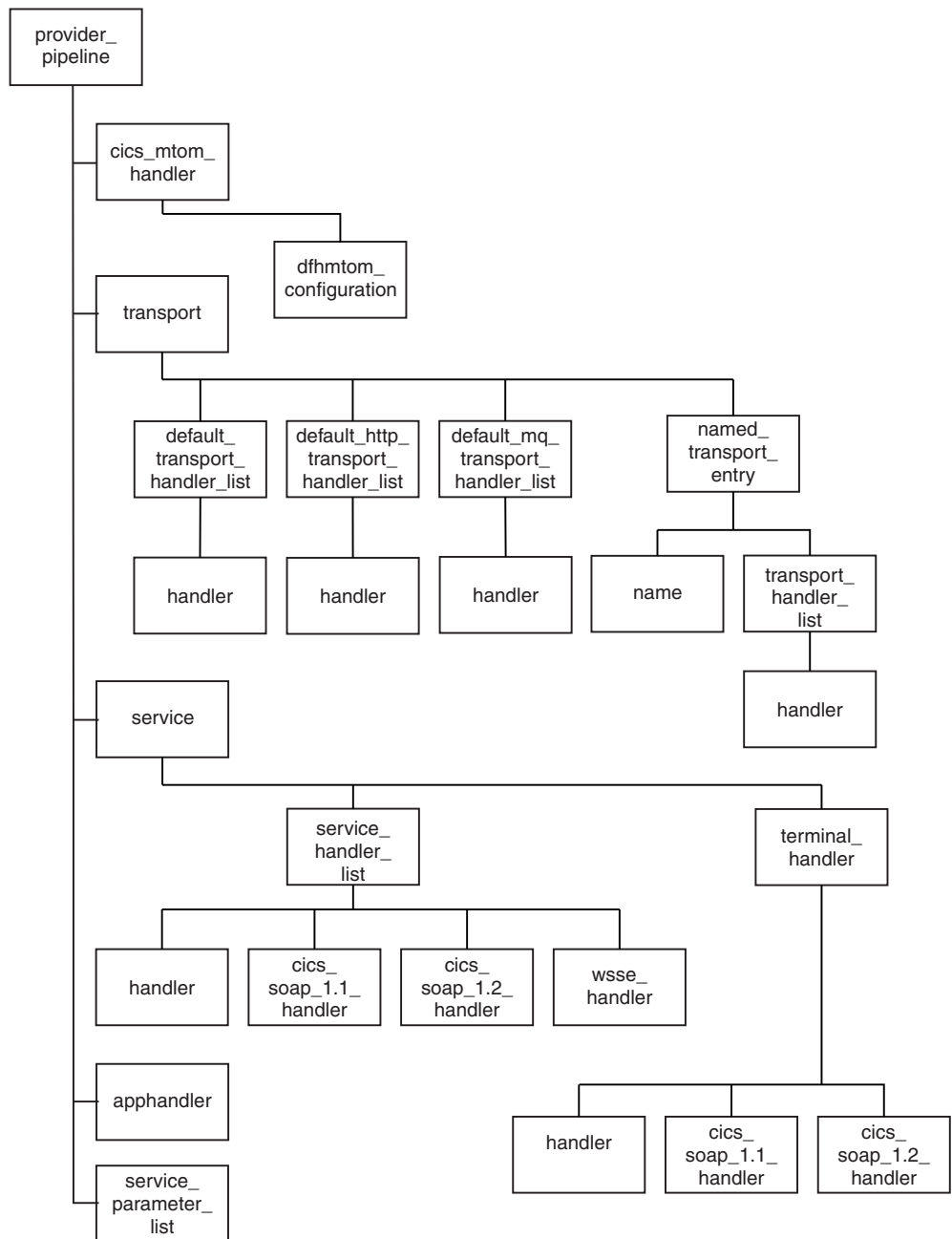


図 21. サービス・プロバイダーのパイプライン定義の構造：

注：図を簡略化するために、<handler>、<cics_soap_1.1_handler>、および<cics_soap_1.2_handler> エレメントの子エレメントは表記していません。

以下のエレメントは、サービス・プロバイダーのパイプライン構成でのみ使用されます。

<named_transport_entry>

<terminal_handler>

その他のエレメントは、サービス・プロバイダーとサービス・リクエスターに共通です。

サービス・リクエスターのパイプライン定義

メッセージ・ハンドラーは、z/OS UNIX に格納されている XML 文書で定義されます。この文書が格納されているファイルの名前は、PIPELINE 定義の CFGFILE 属性で指定します。

パイプライン構成文書のルート・エレメントは、<requester_pipeline> エレメントです。この文書の上位構造を 78 ページの図 22 に示します。

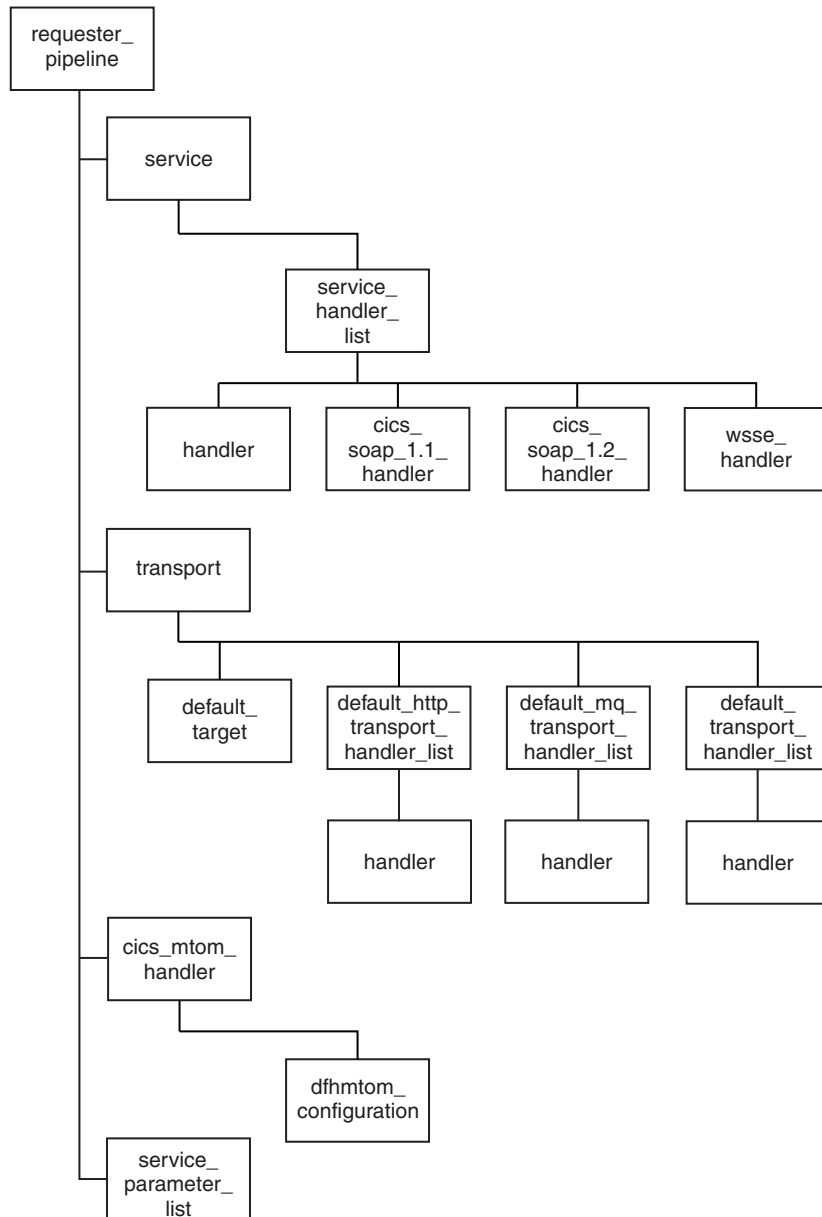


図 22. サービス・リクエスターのパイプライン定義の構造：

注：図を簡略化するために、<handler>、<cics_soap_1.1_handler>、および <cics_soap_1.2_handler> エレメントの子エレメントは表記していません。

サービス・プロバイダーのパイプライン構成で使用するエレメントの一部は、サービス・リクエスターでも使用されます。

サービス・プロバイダーのみで使用されるエレメント

パイプライン構成ファイルで使用する XML エレメントの一部は、サービス・プロバイダー・パイプラインのみに適用されます。

<named_transport_entry>エレメント

指定のトランスポート・リソースがサービス・プロバイダーに使用されると呼び出されるハンドラーのリストが格納されます。

- MQ トランスポートの場合、指定のリソースは要求を受信するローカルの入力キューになります。
- HTTP トランスポートの場合、リソースは要求を受信したポートを定義する TCIPSERVICE になります。

使用先

- サービス・プロバイダー

格納元

<transport>

属性:

名前	説明
type	指定のリソースと関連したトランスポート機構 wmq 指定のリソースはキューです。 http 指定のリソースは TCIPSERVICE です。

内容

1. <name> エレメント。リソースの名前が格納されます。
2. オプションの <transport_handler_list> エレメント。各 <transport_handler_list> には、1 つ以上の <handler> エレメントが格納されます。

<transport_handler_list> エレメントを記述しない場合、指定のトランスポートが使用されると呼び出される唯一のメッセージ・ハンドラーは、<service> エレメントに指定されているメッセージ・ハンドラーになります。

例

```
<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>
```

この例では、指定されたメッセージ・ハンドラー (HANDLER1 および HANDLER2) は、PORT80 という名前で TCIPSERVICE で受信したメッセージに対して呼び出されます。

<provider_pipeline>エレメント

Web サービス・プロバイダーの CICS パイプラインの構成を記述する XML 文書のルート・エレメント。

使用先

- サービス・プロバイダー

内容

1. オプションの `<cics_mtom_handler>` エlement
2. オプションの `<transport>` エlement
3. `<service>` エlement
4. オプションの `<apphandler>` エlement。パイプラインの端末ハンドラーがデフォルトでリンクするプログラムの名前を指定します。

端末ハンドラーが CICS 提供の SOAP メッセージ・ハンドラーのうちの 1 つであるとき、つまり、`<terminal_handler>` エlement に `<cics_soap_1.1_handler>` エlement または `<cics_soap_1.2_handler>` エlement が含まれているときは、`<apphandler>` を使用します。

メッセージ・ハンドラーは実行時に別のプログラムを指定できるため、ここに記述された名前が必ずしもリンク先のプログラムになるわけではありません。`<apphandler>` エlement を記述しなかった場合、いずれかのメッセージ・ハンドラーが DFHWS-APPHANDLER コンテナを使用して、実行時にプログラムの名前を指定する必要があります。

重要: CICS Web サービス・アシスタントを使用して、サービス・プロバイダーを配置する場合は、`<apphandler>` エlement (または DFHWS-APPHANDLER コンテナ) で、ターゲット・アプリケーションやラッパー・プログラムの名前ではなく、DFHPITP を指定する必要があります。この場合、DFHWS2LS または DFHLS2WS の実行時に PGMNAME パラメーターにプログラムの名前を指定します。

5. オプションの `<service_parameter_list>` エlement。コンテナ DFH-SERVICEPLIST のパイプラインに存在するすべてのメッセージ・ハンドラーで使用可能になる XML エlement が格納されます。

例

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

`<terminal_handler>` エlement

サービス・プロバイダー・パイプラインの端末メッセージ・ハンドラーの定義が格納されます。

使用先

- サービス・プロバイダー

格納元

- `<service>` エlement

内容

以下のいずれかのエレメント

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

ただし、<cics_soap_1.1_handler> エレメントと <cics_soap_1.2_handler> エレメントは同じパイプラインに定義しないでください。パイプラインが SOAP 1.1 メッセージと SOAP 1.2 メッセージの両方を処理することを期待する場合は、CICS 提供の SOAP 1.2 メッセージ・ハンドラーを使用してください。

要確認: サービス・プロバイダーでは、<terminal_handler> エレメントの場合と同様に、以下のハンドラーを <service_handler_list> エレメントに指定できます。

例

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

<transport_handler_list>エレメント

指定のリソースが使用されると呼び出されるメッセージ・ハンドラーのリストが格納されます。

- MQ トランスポートの場合、指定のリソースは、ローカル入力キューの名前になります。
- HTTP トランスポートの場合、リソースは要求を受信したポートを定義する TCPIP SERVICE になります。

使用先

- サービス・プロバイダー

格納元

- <named_transport_entry> エレメント

内容

- 1 つ以上の <handler> エレメント。

例

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


サービス・リクエスターのみで使用されるエレメント

パイプライン構成ファイルで使用される XML エレメントの一部は、サービス・リクエスター・パイプラインのみに適用されます。

<requester_pipeline>エレメント

サービス・リクエスターのパイプラインの構成を記述する XML 文書のルート・エレメント。

使用先

- サービス・リクエスター

内容

- オプションの <service> エレメント
- オプションの <transport> エレメント
- オプションの <cics_mtom_handler> エレメント
- オプションの <service_parameter_list> エレメント。コンテナ DFH-SERVICEPLIST のメッセージ・ハンドラーで使用可能になる XML エレメントが格納されます。

例

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

サービス・プロバイダーおよびサービス・リクエスターで使用されるエレメント

パイプライン構成ファイルで使用される XML エレメントの一部は、サービス・プロバイダーとサービス・リクエスターの両方のパイプラインに適用されます。

<cics_soap_1.1_handler>エレメント

SOAP 1.1 メッセージに対応する CICS 提供のハンドラー・プログラムの属性を定義します。

使用先

- サービス・リクエスター
- サービス・プロバイダー

格納元

<service_handler_list> エレメント
<terminal_handler> エレメント

内容

存在しないか 1 つ以上の <headerprogram> エレメント。各 <headerprogram> の内容は、以下のとおりです。

1. <program_name> エlement。ヘッダー処理プログラムの名前が格納されています。
2. <namespace> エlement。ヘッダー処理プログラムが処理する SOAP メッセージのヘッダー・ブロックを調べるときに、次の <localname> エlementと組み合わせて使用します。<namespace> エlementには、ヘッダー・ブロックのネーム・スペースの URI (Universal Resource Identifier) が格納されます。
3. <localname> エlement。ヘッダー処理プログラムが処理する SOAP メッセージのヘッダー・ブロックを調べるときに、前の <namespace> エlementと組み合わせて使用します。<localname> には、ヘッダー・ブロックのElement名が格納されます。

例えば、次のヘッダー・ブロックの場合を考えます。

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

- ネーム・スペース名は http://mynamespace です。
- Element名は myheaderblock です。

ヘッダー・プログラムをこのヘッダー・ブロックに一致させるには、<namespace> Elementおよび <localname> Elementを次のように記述します。

```
<namespace>http://mynamespace</namespace>
<localname>myheaderblock</localname>
```

<localname> Elementにアスタリスク (*) を記述すると、特定の文字ストリングで始まる名前を持つネーム・スペース内のすべてのヘッダー・ブロックを処理するように指定できます。例を次に示します。

```
<namespace>http://mynamespace</namespace>
<localname>myhead*</localname>
```

<localname> Elementにアスタリスクを指定すると、メッセージ内のヘッダーで複数の <headerprogram> Elementを一致させることができます。例えば、次のヘッダー・ブロックの場合を考えます。

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

は、次のすべての <headerprogram> Elementと一致します。

```
<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>
```

この場合、実行されるヘッダー・プログラムは、<headerprogram> エlementで指定されているヘッダー・プログラムの中で、ヘッダー・ブロックのElement名が最も正確に指定されているものです。この例では、HDRPROG3 です。

SOAP メッセージに複数のヘッダーが含まれる場合は、一致するヘッダーがあるたびにヘッダー処理プログラムが 1 回呼び出されますが、ヘッダーの処理順序は定義されていません。

<namespace> と <localname> は同じであるが、異なるヘッダー・プログラムを指定する <headerprogram> Elementを 2 つ以上記述した場合、1 つのヘッダー・プログラムだけが実行されますが、どのプログラムが実行されるかは定義されていません。

4. <mandatory> Element。XML ブール値 (true または false) が格納されています。代わりにそれぞれ 1 または 0 の値をコーディングできます。

true

サービス・プロバイダー・パイプラインでのサービス要求の処理中、およびサービス・リクエスター・パイプラインでのサービス応答の処理中に、SOAP メッセージに <namespace> および <localname> Elementと一致するヘッダーが存在しない場合でも、ヘッダー処理プログラムが 1 回以上呼び出されます。

- 一致するヘッダーがない場合、ヘッダー処理プログラムは 1 回呼び出されます。
- いずれかのヘッダーが一致すると、ヘッダー処理プログラムは、一致したヘッダーごとに 1 回呼び出されます。

サービス・リクエスター・パイプラインでのサービス要求の処理中、およびサービス・プロバイダー・パイプラインでのサービス応答の処理中に、CICS が作成する SOAP メッセージに最初にヘッダーが存在しない場合でも、ヘッダー処理プログラムが 1 回以上呼び出されます。メッセージにヘッダーを追加したい場合は、<mandatory>true</mandatory> または <mandatory>1</mandatory> を指定することにより、少なくとも 1 つのヘッダー処理プログラムが呼び出されるようにする必要があります。

false

SOAP メッセージに <namespace> および <localname> Elementと一致するヘッダーが 1 つ以上存在する場合にのみ、ヘッダー処理プログラムが呼び出されます。

- 一致するヘッダーがない場合、ヘッダー処理プログラムは呼び出されません。
- いずれかのヘッダーが一致すると、ヘッダー処理プログラムは、一致したヘッダーごとに 1 回呼び出されます。

例

```
<cics_soap_1.1_handler>
  <headerprogram>
    <program_name> ... </program_name>
    <namespace>...</namespace>
    <localname>...</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.1_handler>
```

<cics_soap_1.2_handler>エレメント

CICS 提供の SOAP 1.2 メッセージ・ハンドラー・プログラムの属性を定義します。

使用先

- サービス・リクエスター
- サービス・プロバイダー

格納元

<service_handler_list> エレメント

<terminal_handler> エレメント

内容

存在しないか 1 つ以上の <headerprogram> エレメント。各 <headerprogram> の内容は、以下のとおりです。

1. <program_name> エレメント。ヘッダー処理プログラムの名前が格納されています。
2. <namespace> エレメント。ヘッダー処理プログラムが処理する SOAP メッセージのヘッダー・ブロックを調べるときに、次の <localname> エレメントと組み合わせて使用します。<namespace> エレメントには、ヘッダー・ブロックのネーム・スペースの URI (Universal Resource Identifier) が格納されます。
3. <localname> エレメント。ヘッダー処理プログラムが処理する SOAP メッセージのヘッダー・ブロックを調べるときに、前の <namespace> エレメントと組み合わせて使用します。<localname> には、ヘッダー・ブロックのエレメント名が格納されます。

例えば、次のヘッダー・ブロックの場合を考えます。

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

- ネーム・スペース名は http://mynamespace です。
- エレメント名は myheaderblock です。

ヘッダー・プログラムをこのヘッダー・ブロックに一致させるには、<namespace> エレメントおよび <localname> エレメントを次のように記述します。

```
<namespace>http://mynamespace</namespace>  
<localname>myheaderblock</localname>
```

<localname> エレメントにアスタリスク (*) を記述すると、特定の文字ストリングで始まる名前を持つネーム・スペース内のすべてのヘッダー・ブロックを処理するように指定できます。例を次に示します。

```
<namespace>http://mynamespace</namespace>  
<localname>myhead*</localname>
```

<localname> エレメントにアスタリスクを指定すると、メッセージ内のヘッダーで複数の <headerprogram> エレメントを一致させることができます。例えば、次のヘッダー・ブロックの場合を考えます。

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

は、次のすべての <headerprogram> エlementと一致します。

```
<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>
```

この場合、実行されるヘッダー・プログラムは、<headerprogram> Elementで指定されているヘッダー・プログラムの中で、ヘッダー・ブロックのElement名が最も正確に指定されているものです。この例では、 HDRPROG3 です。

SOAP メッセージに複数のヘッダーが含まれる場合は、一致するヘッダーがあるたびにヘッダー処理プログラムが 1 回呼び出されますが、ヘッダーの処理順序は定義されていません。

<namespace> と <localname> は同じであるが、異なるヘッダー・プログラムを指定する <headerprogram> Elementを 2 つ以上記述した場合、1 つのヘッダー・プログラムだけが実行されますが、どのプログラムが実行されるかは定義されていません。

4. <mandatory> Element。XML ブール値 (true または false) が格納されています。代わりにそれぞれ 1 または 0 の値をコーディングできます。

true

サービス・プロバイダー・パイプラインでのサービス要求の処理中、およびサービス・リクエスター・パイプラインでのサービス応答の処理中に、SOAP メッセージに <namespace> および <localname> Elementと一致するヘッダーが存在しない場合でも、ヘッダー処理プログラムが 1 回以上呼び出されます。

- 一致するヘッダーがない場合、ヘッダー処理プログラムは 1 回呼び出されます。
- いずれかのヘッダーが一致すると、ヘッダー処理プログラムは、一致したヘッダーごとに 1 回呼び出されます。

サービス・リクエスター・パイプラインでのサービス要求の処理中、およびサービス・プロバイダー・パイプラインでのサービス応答の処理中に、CICS が作成する SOAP メッセージに最初にヘッダーが存在しない場合でも、ヘッダー処理プログラムが 1 回以上呼び出されます。メッセージにヘッダーを追加したい場合は、 <mandatory>true</mandatory> または <mandatory>1</mandatory> を指定することにより、少なくとも 1 つのヘッダー処理プログラムが呼び出されるようにする必要があります。

false

SOAP メッセージに <namespace> および <localname> エlementと一致するヘッダーが 1 つ以上存在する場合にのみ、ヘッダー処理プログラムが呼び出されます。

- 一致するヘッダーがない場合、ヘッダー処理プログラムは呼び出されません。
- いずれかのヘッダーが一致すると、ヘッダー処理プログラムは、一致したヘッダーごとに 1 回呼び出されます。

例

```
<cics_soap_1.2_handler>
  <headerprogram>
    <program_name> ... </program_name>
    <namespace>...</namespace>
    <localname>...</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler>
```

<default_http_transport_handler_list>Element

HTTP トランスポートが使用中の場合、デフォルトで呼び出されるメッセージ・ハンドラーを指定します。

サービス・プロバイダーの場合、このリストで指定されているメッセージ・ハンドラーが呼び出されるのは、<named_transport_entry> Elementに定義されているハンドラーのリストがあまり具体的でない場合のみです。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

- <transport> Element

内容

- 1 つ以上の <handler> Element。

例

```
<default_http_transport_handler_list>
  <handler>
    ...
  </handler>
  <handler>
    ...
  </handler>
</default_http_transport_handler_list>
```

<default_mq_transport_handler_list>Element

WebSphere MQ トランスポートが使用中の場合、デフォルトで呼び出されるメッセージ・ハンドラーを指定します。

サービス・プロバイダーの場合、このリストで指定されているメッセージ・ハンドラーが呼び出されるのは、<named_transport_entry> エlementに定義されているハンドラーのリストがあまり具体的でない場合のみです。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

- <transport> Element

内容

- 1 つ以上の <handler> Element。

例

```
<default_mq_transport_handler_list>
  <handler>
    ...
  </handler>
  <handler>
    ...
  </handler>
</default_mq_transport_handler_list>
```

<default_transport_handler_list>Element

いずれかのトランスポートが使用中の場合、デフォルトで呼び出されるメッセージ・ハンドラーを指定します。

サービス・プロバイダーでは、以下のいずれかのElementに定義されたハンドラーのリストがあまり具体的でない場合に、このリストに指定されたメッセージ・ハンドラーが呼び出されます。

```
  <default_http_transport_handler_list>
  <default_mq_transport_handler_list>
  <named_transport_entry>
```

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

- <transport> Element

内容

- 1 つ以上の <handler> Element。

例

```
<default_transport_handler_list>
  <handler>
    <program>HANDLER1</program>
    <handler_parameter_list/>
  </handler>
```



```

    <handler>
      <program>HANDLER2</program>
      <handler_parameter_list/>
    </handler>
  </default_transport_handler_list>

```

<handler>エレメント

メッセージ・ハンドラー・プログラムの属性を定義します。

CICS 提供のハンドラー・プログラムの中には、<handler> エレメントを使用しないものがあります。例えば、CICS 提供の SOAP メッセージ・ハンドラー・プログラムは、<cics_soap_1.1_handler> エレメントおよび <cics_soap_1.2_handler> エレメントを使用して定義されます。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```

  <default_transport_handler_list>
    <transport_handler_list>
      <service_handler_list>
        <terminal_handler>
          <default_http_transport_handler_list>
          <default_mq_transport_handler_list>

```

内容

1. <program> エレメント。ハンドラー・プログラムの名前が格納されます。
2. <handler_parameter_list> エレメント。コンテナ DFH-HANDLERPLIST のメッセージ・ハンドラーで使用可能になる XML エレメントが格納されます。

例

```

<handler>
  <program>MYPROG</program>
  <handler_parameter_list><output print="yes"/></handler_parameter_list>
</handler>

```

この例では、ハンドラー・プログラムは MYPROG です。ハンドラー・パラメーター・リストは、単一の <output> エレメントで構成されます。このパラメーター・リストの内容は、MYPROG に認識されます。

<service>エレメント

すべての要求に対して呼び出されるメッセージ・ハンドラーを指定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<provider_pipeline>  
<requester_pipeline>
```

内容

1. <service_handler_list> エlement
2. サービス・プロバイダーの場合に限り、<terminal_handler> Element

例

```
<service>  
  <service_handler_list>  
    ...  
  </service_handler_list>  
  <terminal_handler>  
    ...  
  </terminal_handler>  
</service>
```

<service_handler_list>Element

すべての要求に対して呼び出されるメッセージ・ハンドラーのリストを指定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

- <service> Element

内容

以下のElementのうち 1 つ以上のElement

```
<handler>  
<cics_soap_1.1_handler>  
<cics_soap_1.2_handler>  
<wsse_handler>
```

<cics_soap_1.1_handler> Elementと <cics_soap_1.2_handler> Elementは同じパイプラインに定義してはなりません。

サービス・プロバイダー・パイプラインが SOAP 1.1 メッセージと SOAP 1.2 メッセージの両方を処理することが予想される場合は、CICS 提供の SOAP 1.2 メッセージ・ハンドラーを使用してください。これは SOAP 1.1 と SOAP 1.2 の両方のメッセージをサポートします。

サービス・リクエスター・パイプラインでは SOAP 1.1 または SOAP 1.2 のいずれかのハンドラーを使用できますが、この場合 SOAP 1.2 ハンドラーは SOAP 1.1 メッセージをサポートしません。サービス・リクエスター・アプリケーションが DFHREQUEST コンテナ内の完全な SOAP エンベロープを送信する場合は、パイ

パイプラインに SOAP 1.1 または SOAP 1.2 ハンドラーを指定しないでください。これを行うと、アウトバウンド・メッセージで SOAP メッセージ・ヘッダーが重複するのを回避できます。

サービス・プロバイダーでは、<service_handler_list> エレメントの場合と同様に、汎用ハンドラーと SOAP ハンドラーを <terminal_handler> エレメントに指定できます。複数の SOAP ハンドラーを CICS パイプラインに指定すると、SOAP ヘッダー処理に対して特定の順序を実施できます。ほとんどの場合、これは必要ありません。

例

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

<service_parameter_list>エレメント

オプションのエレメントは、コンテナ DFH-SERVICEPLIST のパイプラインに存在するすべてのメッセージ・ハンドラーで使用可能になる XML エレメントが格納されます。

使用先

- サービス・リクエスター
- サービス・プロバイダー

内容

- WS-AT を使用している場合は、1 つの <registration_service_endpoint> エレメント。
- サービス・リクエスターで WS-AT を使用している場合、オプションの <new_tx_context_required/> エレメント
- オプションのユーザー定義タグ

例

```
<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>
```

関連資料

82 ページの『<requester_pipeline>エレメント』

サービス・リクエスターのパイプラインの構成を記述する XML 文書のルート・エレメント。

79 ページの『<provider_pipeline>エレメント』

Web サービス・プロバイダーの CICS パイプラインの 構成を記述する XML 文書のルート・エレメント。

<transport>エレメント

特定のトランスポートが使用中の場合にのみ呼び出されるハンドラーを指定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<provider_pipeline>
<requester_pipeline>
```

内容

サービス・プロバイダーの場合

1. オプションの <default_transport_handler_list> エレメント
2. オプションの <default_http_transport_handler_list> エレメント
3. オプションの <default_mq_transport_handler_list> エレメント
4. 存在しないか 1 つ以上の <named_transport_entry> エレメント

サービス・リクエスターの場合

1. オプションの <default_target> エレメント。 <default_target> には、サービス・リクエスター・アプリケーションによって URI が提供されない場合、ターゲット Web サービスの場所を特定するために CICS が使用する URI を指定します。ただし、多くの場合、ターゲットの URI はサービス・リクエスター・アプリケーションによって提供されるため、<default_target> に指定しても無視されます。例えば、CICS Web サービス・アシスタントを使用して配置されるサービス・プロバイダー・アプリケーションは、通常は Web サービス記述から URI を取得します。
2. オプションの <default_http_transport_handler_list> エレメント
3. オプションの <default_mq_transport_handler_list> エレメント
4. オプションの <default_transport_handler_list> エレメント

例

```
<transport>
  <default_transport_handler_list>
    ...
  </default_transport_handler_list>
</transport>
```

MTOM/XOP に合わせたパイプライン構成

バイナリー添付ファイルが含まれた MIME メッセージを Web サービス・リクエスター・アプリケーションおよび Web サービス・プロバイダー・アプリケーションで送受信できるようにするには、それに応じてパイプラインを構成する必要があります。

ます。これによって、MTOM ハンドラーが、ユーザーが定義した構成オプションを使用して、パイプラインで MIME メッセージを処理できるようになります。

<cics_mtom_handler>エレメント

XOP 文書およびバイナリー添付ファイルを含む MTOM MIME multipart/related メッセージのサポートを提供する、CICS 提供の MTOM ハンドラー・プログラムを使用可能にします。MTOM サポートは、パイプライン内に受信されるすべてのインバウンド・メッセージについて使用可能になりますが、アウトバウンド・メッセージについての MTOM サポートは、追加のオプションに従って条件付きで使用可能になります。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<provider_pipeline>
  <requester_pipeline>
```

プロバイダー・パイプライン構成ファイルでは、<cics_mtom_handler> エレメントを <transport> エレメントの前に定義してください。実行時に、トランスポート・ハンドラーを含む他のハンドラーが処理する前に、MTOM ハンドラー・プログラムがインバウンド MTOM メッセージをアンパックする必要があります。またこの後で、応答メッセージについての最終ハンドラーとして呼び出され、MTOM メッセージをパックして Web サービス・リクエスターに送信します。

リクエスター・パイプライン構成ファイルでは、<cics_mtom_handler> エレメントを <transport> エレメントの後に定義してください。実行時に、他のすべてのハンドラーが処理するまで、アウトバウンド要求メッセージは MTOM フォーマットに変換されません。またこの後で、インバウンド応答メッセージについて最初のハンドラーとして呼び出され、他のハンドラーが処理する前に MTOM メッセージをアンパックし、要求しているプログラムに戻します。

内容

<dfhmtom_configuration> エレメント

デフォルト・オプションは、<dfhmtom_configuration> エレメントに指定される構成オプションを使用して変更できます。デフォルト・オプションを変更しない場合は、空のエレメントを使用できます。

例

プロバイダー・モードのパイプラインでは、以下のように指定できます。

```
<provider_pipeline>
  <cics_mtom_handler/>
  <transport>
    ....
  </transport>
```

```

    <service>
    ....
    </service>
</provider_pipeline>

```

<dfhmtom_configuration>エレメント

XOP 文書およびバイナリー添付ファイルを含む MIME メッセージのサポートを提供する、CICS 提供の MTOM ハンドラー・プログラムの構成情報を指定します。MTOM についての構成を何も指定しないと、CICS はデフォルト値を想定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<cics_mtom_handler>
```

属性:

名前	説明
version	構成情報のバージョンを示す整数。有効な値は 1 のみです。

内容

1. オプションの <mtom_options> エレメント
2. オプションの <xop_options> エレメント
3. オプションの <mime_options> エレメント

例

```

<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>

```

<mtom_options>エレメント

アウトバウンド SOAP メッセージに MTOM を使用する場合に指定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<dfhmtom_configuration>
```

属性:

属性	説明
send_mtom	アウトバウンド SOAP メッセージを MIME メッセージに変換する場合に MTOM を使用するかどうかを指定します。 no アウトバウンド SOAP メッセージに MTOM を使用しません。 same サービス・プロバイダー・モードでは、リクエスターが MTOM を使用するときはいつでも、SOAP 応答メッセージに MTOM を使用します。これは、サービス・プロバイダー・パイプラインでのデフォルト値です。 サービス・リクエスター・モードでは、この値を指定することは、send_mtom="yes" を指定することと同じです。 yes すべてのアウトバウンド SOAP メッセージに MTOM を使用します。これは、サービス・リクエスター・パイプラインでのデフォルト値です。
send_when_no_xop	メッセージにバイナリー添付ファイルが存在しない場合でも MTOM メッセージを送信するかどうかを指定します。 no メッセージとともにバイナリー添付ファイルを送信する場合のみ、MTOM を使用します。 yes メッセージに送信するバイナリー添付ファイルが存在しない場合でも、すべてのアウトバウンド SOAP メッセージに MTOM を使用します。これはデフォルト値で、主として送信側が MTOM/XOP をサポートする受信側プログラムへの標識として使用されます。 この属性は、任意の send_mtom 属性値と組み合わせることができますが、send_mtom="no" を指定する場合は無効になります。

例

```
<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>
```

このプロバイダー・パイプラインの例では、メッセージとともにバイナリー添付ファイルを送信する必要があり、サービス・リクエスターが MTOM メッセージを送信した場合のみ、SOAP メッセージは MTOM メッセージに変換されます。

<xop_options>エレメント

XOP 処理を直接モードで行うか、または互換モードで行うかを指定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

<dfhmtom_configuration>

属性:

属性	説明
apphandler_ supports_xop	プロバイダー・モードでは、アプリケーション・ハンドラーが直接モードで XOP 文書进行处理できる場合に指定します。 no アプリケーション・ハンドラーは XOP 文書を直接处理できません。これは、<apphandler> エレメントで DFHPITP が指定されない場合のデフォルト値です。 互換モードは、MTOM 形式で受信あるいは送信されたインバウンド・メッセージまたはアウトバウンド・メッセージ进行处理するためにパイプラインで使用されます。 yes アプリケーション・ハンドラーは XOP 文書进行处理できます。これは、<apphandler> エレメントで DFHPITP が指定された場合のデフォルト値です。 直接モードは、MTOM 形式で受信あるいは送信されたインバウンド・メッセージまたはアウトバウンド・メッセージ进行处理するためにパイプラインで使用されます。これは、実行時に制約の対象となります。例えば、パイプライン構成ファイルで WS-Security 関連のエレメントを指定した場合、MTOM ハンドラーは、パイプラインでは XOP 文書の处理に直接モードではなく互換モードを使用することを決定します。 リクエスター・モードでは、サービス・リクエスター・アプリケーションが CICS Web サービス・サポートを使用して、直接モードで XOP 文書を作成および处理する場合に指定します。 no サービス・リクエスター・アプリケーションは、CICS Web サービス・サポートを使用しません。この値は、リクエスター・アプリケーションがパイプラインを実行するため DFHPITP にリンクしているために、XOP 文書を直接モードで作成および处理できない場合に指定します。 yes サービス・リクエスター・アプリケーションは、CICS Web サービス・サポートを使用します。この値は、リクエスター・アプリケーションが EXEC CICS INVOKE WEBSERVICE コマンドを使用する場合に指定します。

例

```
<provider_pipeline>
  <cics_mtom_handler>
    <dfhmtom_configuration version="1">
      <xop_options apphandler_supports_xop="no"/>
    </dfhmtom_configuration>
  </cics_mtom_handler>
  ...
</provider_pipeline>
```

このプロバイダー・パイプラインの例では、インバウンド MTOM メッセージとアウトバウンド応答メッセージは、互換モードを使用してパイプラインで処理されます。

<mime_options>エレメント

MIME コンテンツ ID 値の生成時に使用しなければならないドメイン名を指定します。この値は、バイナリー添付ファイルを識別するために使用されます。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<dfhmtom_configuration>
```

属性:

属性	説明
content_id_domain	使用する構文は <i>domain.name</i> です。 インターネット標準に準拠するためには、名前は有効なインターネット・ホスト名で、パイプラインがインストールされている CICS システムに対して一意でなければなりません。これは CICS によって検査されないことに注意してください。 このエレメントを省略すると、CICS は値 <i>cicsts</i> を使用します。

例

```
<provider_pipeline>
<dfhmtom_configuration version="1">
  <mime_options content_id_domain="example.org"/>
</dfhmtom_configuration>
...
</provider_pipeline>
```

この例では、バイナリー添付ファイルへの参照は、*cid:unique_value@example.org* を使用して作成されます。

WS-Security に合わせたパイプライン構成

Web サービス・リクエスター・アプリケーションおよび Web サービス・プロバイダー・アプリケーションが WS-Security プロトコルに参加するには、それに応じて

パイプラインを構成する必要があります。このためには、メッセージ・ハンドラー DFHWSSE を組み込んで、ハンドラーの構成情報を提供します。

例

以下は、WS-Security を使用するプロバイダー・パイプライン構成ファイルの形式の一例です。

```
<?xml version="1.0"?>
<provider_pipeline
  xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <service_handler_list>
      <wsse_handler>
        <dfhwsse_configuration version="1">
          <authentication trust="blind" mode="basic"/>
        </dfhwsse_configuration>
      </wsse_handler>
    </service_handler_list>
    <terminal_handler>
      <cics_soap_1.2_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

<wsse_handler>エレメント

WS-Security のサポートを提供する CICS 提供のメッセージ・ハンドラーで使われるパラメーターを指定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<service_handler_list>
```

内容

- <dfhwsse_configuration> エレメント。

<dfhwsse_configuration>エレメント

Web サービスの保護についてのサポートを提供する、セキュリティー・ハンドラー DFHWSSE1 の構成情報を指定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<wsse_handler>
```

属性:

名前	説明
version	構成情報のバージョンを示す整数。有効な値は 1 のみです。

内容

1. 以下のいずれかのエレメント

- オプションの `<authentication>` エレメント。
 - サービス・リクエスター・パイプラインでは、`<authentication>` エレメントが、アウトバウンド SOAP メッセージのセキュリティ・ヘッダーで使用する必要がある認証のタイプを指定します。
 - サービス・プロバイダー・パイプラインでは、このエレメントが、CICS がインバウンド SOAP メッセージでセキュリティ・トークンを使用して、処理が行われるユーザー ID を決定するかどうかを指定します。
- オプションの `<sts_authentication>` エレメント。

このエレメントの `action` 属性は、Security Token Service に送信する要求のタイプを指定します。要求が ID トークンの発行である場合、CICS は、ネストされたエレメント内の値を使用して、指定されたタイプの ID トークンを要求します。

2. `<sts_authentication>` エレメントを指定する場合は、`<sts_endpoint>` エレメントも指定する必要があります。

このエレメントが存在するとき、CICS は、`<endpoint>` エレメント内の URI を使用して、要求を Security Token Service に送信します。

3. オプションの、空の `<expect_signed_body/>` エレメント。

`<expect_signed_body/>` エレメントは、インバウンド・メッセージの `<body>` に署名が必要であることを示します。インバウンド・メッセージの本文が正しく署名されていない場合、CICS はセキュリティ障害でメッセージを拒否します。

4. オプションの、空の `<expect_encrypted_body/>` エレメント。

`<expect_encrypted_body/>` エレメントは、インバウンド・メッセージの `<body>` を暗号化する必要があることを示します。インバウンド・メッセージの本文が正しく暗号化されていない場合、CICS はセキュリティ障害でメッセージを拒否します。

5. オプションの `<sign_body>` エレメント。

このエレメントが存在する場合、CICS は、`<sign_body>` エレメント内に含まれる `<algorithm>` エレメントに指定されるアルゴリズムを使用して、アウトバウンド・メッセージの `<body>` に署名します。

6. オプションの `<encrypt_body>` エレメント。

このエレメントが存在する場合、CICS は、`<encrypt_body>` エレメント内に含まれ、`<algorithm>` エレメントに指定されるアルゴリズムを使用して、アウトバウンド・メッセージの `<body>` を暗号化します。

7. プロバイダー・パイプラインのみで、オプションの `<reject_signature/>` エレメント。

このエレメントが存在する場合、CICS は、メッセージの本文の一部またはすべてを署名する証明書がヘッダーに含まれているメッセージを拒否します。Web サービス・リクエスターに SOAP 障害が発行されます。

8. プロバイダー・パイプラインのみで、オプションの `<reject_encryption>` エレメント。

このエレメントが存在する場合、CICS は、部分的または完全に暗号化されているメッセージを拒否します。Web サービス・リクエスターに SOAP 障害が発行されます。

例

```
<dfwsse_configuration version="1">
  <sts_authentication action="issue">
    <auth_token_type>
      <namespace>http://example.org.tokens</namespace>
      <element>UsernameToken</element>
    </auth_token_type>
    <suppress/>
  </sts_authentication>
  <sts_endpoint>
    <endpoint>https://example.com/SecurityTokenService</endpoint>
  </sts_endpoint>
  <expect_signed_body/>
  <expect_encrypted_body/>
  <sign_body>
    <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
    <certificate_label>SIGCERT01</certificate_label>
  </sign_body>
  <encrypt_body>
    <algorithm>http://www.w3.org/2001/04/xmlenc#tripledes-cbc</algorithm>
    <certificate_label>ENCCERT02</certificate_label>
  </encrypt_body>
</dfwsse_configuration>
```

<authentication>エレメント

インバウンドおよびアウトバウンド SOAP メッセージのヘッダー内の、セキュリティー・トークンの使用について指定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<dfwsse_configuration>
```

属性:

属性	説明
trust	trust 属性と mode 属性と一緒に使用して、次のことを指定します。 - 宣言 ID が使用されるかどうか - SOAP メッセージで使用するセキュリティー・トークンの組み合わせ 宣言 ID によって、信頼できるユーザーがその ID に関連するクレデンシャルを持っていなくても、別の ID、つまり宣言 ID の下で作業を実行することを、信頼できるユーザーが宣言できます。 宣言 ID が使用されるとき、メッセージには trust トークン および ID トークン が含まれます。trust トークンは、宣言 ID に対する正しい許可が送信側にあるか検査するために使用され、ID トークンには、宣言 ID、つまり要求が実行されるユーザー ID が格納されます。
mode	宣言 ID を使用するには、サービス・プロバイダーがリクエスターを信頼してこの宣言を行う必要があります。CICS では、信頼関係は、セキュリティー・マネージャーの代理定義で確立されます。要求している ID には、宣言 ID の代わりに作業を開始するための、正しい権限が必要です。 これらの属性の許容できる組み合わせと、その意味について、表 2 および 102 ページの表 3 で説明します。

表 2. サービス・リクエスター・パイプラインにおける mode および trust 属性

trust	mode	意味
none	none	メッセージにクレデンシャルが追加されません
	basic	属性値の組み合わせが無効です
	signature	宣言 ID は使用されません。CICS は、メッセージに追加され、メッセージの本文の署名に使用される、単一の X.509 セキュリティー・トークンを使用します。証明書は <certificate_label> エLEMENTで識別され、アルゴリズムは <algorithm> ELEMENTで指定されます。
blind	none	属性値の組み合わせが無効です
	basic	宣言 ID は使用されません。CICS は ID トークンをメッセージに追加しますが、trust トークンは提供しません。ID トークンは、パスワードのないユーザー名です。ID トークンに配置されるユーザー ID は、DFHWS-USERID コンテナの内容です (デフォルトで、実行中のタスクのユーザー ID を含みます)。
	signature	属性値の組み合わせが無効です
basic	(任意)	属性値の組み合わせが無効です

表 2. サービス・リクエスター・パイプラインにおける **mode** および **trust** 属性 (続き)

trust	mode	意味
signature	none	属性値の組み合わせが無効です
	basic	宣言 ID が使用されます。CICS はメッセージに以下のトークンを追加します。 • trust トークンは、X.509 セキュリティー・トークンです。 • ID トークンは、パスワードのないユーザー名です。 ID トークンおよびメッセージの本文の署名に使用される証明書は、<certificate_label> によって指定されます。ID トークンに配置されるユーザー ID は、DFHWS-USERID コンテナの内容です (デフォルトで、実行中のタスクのユーザー ID を含みます)。
	signature	属性値の組み合わせが無効です

表 3. サービス・プロバイダー・パイプラインにおける **mode** および **trust** 属性

trust	mode	意味
none	none	インバウンド・メッセージはクレデンシャルを含む必要がないので、CICS はメッセージ内で検出されるクレデンシャルの抽出または検証を試みません。ただし、CICS は、署名されているすべてのエレメントについて、署名が正しいことを検査します。
	basic	インバウンド・メッセージに、パスワードのあるユーザー名セキュリティ・トークンが含まれる必要があります。CICS はユーザー名を DFHWS-USERID コンテナに入れます。
	signature	インバウンド・メッセージに、メッセージの本文の署名に使用された X.509 セキュリティー・トークンが含まれる必要があります。
blind	none	属性値の組み合わせが無効です
	basic	インバウンド・メッセージに、ID トークンが含まれる必要があります。ID トークンには、ユーザー ID と、オプションでパスワードが含まれます。CICS はユーザー ID を DFHWS-USERID コンテナに入れます。パスワードが含まれない場合は、CICS はユーザー ID を検証せずに使用します。パスワードが含まれる場合は、セキュリティ・ハンドラー DFHWSSE1 が検証します。
	signature	インバウンド・メッセージに、ID トークンが含まれる必要があります。ID トークンは、SOAP メッセージ・ヘッダー内の最初の X.509 証明書です。この証明書でメッセージに署名をしておく必要はありません。セキュリティ・ハンドラーが一致するユーザー ID を抽出し、DFHWS-USERID コンテナに配置します。

表 3. サービス・プロバイダー・パイプラインにおける **mode** および **trust** 属性 (続き)

trust	mode	意味
basic	none	属性値の組み合わせが無効です
	basic	インバウンド・メッセージは宣言 ID を使用する必要があります。 • trust トークンは、パスワードのあるユーザー名トークンです。 • ID トークンは、パスワードのない 2 番目のユーザー名トークンです。 CICS はこのユーザー名をコンテナ DFHWS-USERID に入れます。
	signature	インバウンド・メッセージは宣言 ID を使用する必要があります。 • trust トークンは、パスワードのあるユーザー名トークンです。 • ID トークンは、X.509 証明書です。 CICS は、証明書に関連するユーザー ID を、コンテナ DFHWS-USERID に入れます。
signature	none	属性値の組み合わせが無効です
	basic	インバウンド・メッセージは宣言 ID を使用する必要があります。 • trust トークンは X.509 証明書です • ID トークンは、パスワードのないユーザー名トークンです。 CICS はユーザー名をコンテナ DFHWS-USERID に入れます。 ID トークンおよび本文は、X.509 証明書で署名する必要があります。
	signature	インバウンド・メッセージは宣言 ID を使用する必要があります。 • trust トークンは X.509 証明書です • ID トークンは、2 番目の X.509 証明書です。 CICS は、この証明書に関連するユーザー ID を、コンテナ DFHWS-USERID に入れます。 ID トークンおよび本文は、最初の X.509 証明書 (trust トークン) で署名する必要があります。

注:

1. trust 属性値と mode 属性値の組み合わせは、PIPELINE がインストールされるときに検査されます。属性のコーディングが誤っていると、インストールは失敗します。

内容

1. オプションの、空の <suppress/> エレメント。

このエレメントがサービス・プロバイダー・パイプラインに指定される場合、ハンドラーは、作業が行われるユーザー ID を決定するメッセージ内のどのセキュリティ・トークンの使用も試みません。

このエレメントがサービス・リクエスター・パイプラインに指定される場合、ハンドラーは、アウトバウンド SOAP メッセージに、認証に必要な、どのセキュリティ・トークンの追加も試みません。

2. リクエスター・パイプラインでは、SOAP メッセージの本文の署名に使用されるアルゴリズムの URI を指定する、オプションの `<algorithm>` エレメント。trust 属性と mode 属性の値の組み合わせが、メッセージが署名されていることを示している場合は、このエレメントを指定する必要があります。このエレメントでは、SHA1 を使用する RSA アルゴリズムだけを指定できます。URI は `http://www.w3.org/2000/09/xmldsig#rsa-sha1` です。
3. RACF® にインストールされる X.509 デジタル証明書に関連したラベルを指定する、オプションの `<certificate_label>` エレメント。このエレメントがサービス・リクエスター・パイプラインに指定され、`<suppress>` エレメントが指定されない場合は、証明書が SOAP メッセージのセキュリティ・ヘッダーに追加されます。`<certificate_label>` エレメントを指定しない場合は、CICS が RACF 鍵リングでデフォルトの証明書を使用します。

このエレメントはサービス・プロバイダー・パイプラインでは無視されます。

例

```
<authentication trust="signature" mode="basic">
  <suppress/>
  <certificate_label>AUTHCERT03</certificate_label>
</authentication>
```

<sts_authentication>エレメント

認証のために Security Token Service (STS) を使用する必要があることを指定し、送信される要求のタイプを決定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<dfhwsse_configuration>
```


属性:

名前	説明
action	サービス・プロバイダー・パイプラインでメッセージを受信したときに CICS が STS に送信する要求のタイプを指定します。有効な値は次のとおりです。 issue STS は、SOAP メッセージの識別トークンを発行します。 validate STS は、提供された識別トークンを妥当性検査して、トークンが有効かどうかをセキュリティー・ハンドラーに戻します。 この属性を指定しない場合、CICS は、アクションは識別トークンを要求することだと想定します。 サービス・リクエスター・パイプラインでは、CICS は必ず STS によるトークンの発行を要求するため、この属性を指定することはできません。

内容

1. `<auth_token_type>` エlement。このElementは、サービス・リクエスター・パイプラインで `<sts_authentication>` Elementを指定する場合は必須で、サービス・プロバイダー・パイプラインではオプションです。詳しくは、`<auth_token_type>` を参照してください。
 - サービス・リクエスター・パイプラインでは、`<auth_token_type>` Elementは、CICS が DFHWS-USERID コンテナに含まれるユーザー ID を STS に送信したときに、STS が発行するトークンのタイプを示します。CICS が STS から受け取るトークンは、アウトバウンド・メッセージのヘッダーに置かれます。
 - サービス・プロバイダー・パイプラインでは、`<auth_token_type>` Elementは、CICS がメッセージ・ヘッダーから取得して、交換または妥当性検査のために STS に送信する識別トークンを判別するために使用されます。CICS は最初に、メッセージ・ヘッダーで指定されたタイプの識別トークンを使用します。このElementを指定しない場合、CICS は、メッセージ・ヘッダーで見つけた最初の識別トークンを使用します。CICS は、次のものは ID トークンと見なしません。
 - `wsu:Timestamp`
 - `xenc:ReferenceList`
 - `xenc:EncryptedKey`
 - `ds:Signature`
2. サービス・プロバイダー・パイプラインの場合に限り、オプションの空の `<suppress/>` Element。このElementが指定される場合、ハンドラーは、作業が行われるユーザー ID を決定するメッセージ内のどのセキュリティー・トークンの使用も試みません。`<suppress/>` Elementは、STS によって戻される ID トークンが含まれます。

例

次の例は、セキュリティー・ハンドラーが STS からトークンを要求するサービス・プロバイダー・パイプラインを示します。

```
<sts_authentication action="issue">
  <auth_token_type>
    <namespace>http://example.org.tokens</namespace>
    <element>UsernameToken</element>
  </auth_token_type>
  <suppress/>
</sts_authentication>
```

<auth_token_type>エレメント

必要な ID トークンのタイプを指定します。

このエレメントは、サービス・リクエスターで <sts_authentication> エレメントを指定するときは必須で、サービス・プロバイダーではオプションです。

- サービス・リクエスター・パイプラインでは、<auth_token_type> エレメントは、CICS が DFHWS-USERID コンテナに含まれるユーザー ID を STS に送信したときに、STS が発行するトークンのタイプを示します。CICS が STS から受け取るトークンは、アウトバウンド・メッセージのヘッダーに置かれます。
- サービス・プロバイダー・パイプラインでは、<auth_token_type> エレメントは、CICS がメッセージ・ヘッダーから取得して、交換または妥当性検査のために STS に送信する識別トークンを判別するために使用されます。CICS は最初に、メッセージ・ヘッダーで指定されたタイプの識別トークンを使用します。このエレメントを指定しない場合、CICS は、メッセージ・ヘッダーで見つけた最初の識別トークンを使用します。CICS は、次のものは ID トークンと見なしません。

- wsu:Timestamp
- xenc:ReferenceList
- xenc:EncryptedKey
- ds:Signature

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

<sts_authentication>

内容

- <namespace> エレメント。このエレメントには、検証または交換する必要があるトークン・タイプのネーム・スペースが含まれます。
- <element> エレメント。このエレメントには、検証または交換する必要があるトークン・タイプのローカル名が含まれます。

これらのエレメントの値は、トークンの QName を形成します。

例

```
<auth_token_type>
  <namespace>http://example.org.tokens</namespace>
  <element>UsernameToken</element>
</auth_token_type>
```

<sts_endpoint>エレメント

Security Token Service (STS) の場所を指定します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<dfwsse_configuration>
```

内容

- <endpoint> エレメント。このエレメントには、ネットワーク上の Security Token Service (STS) の場所を指し示す URI が含まれます。 STS への接続を安全に保つには、HTTP ではなく、SSL または TLS を使用することをお勧めします。

また、WebSphere MQ エンドポイントは、JMS フォーマットの URI を使用して指定することもできます。

例

この例では、エンドポイントは、指定した URI にある STS へのセキュア接続を使用するように構成されています。

```
<sts_endpoint>
  <endpoint>https://example.com/SecurityTokenService</endpoint>
</sts_endpoint>
```

<sign_body>エレメント

アウトバウンド SOAP メッセージの本文に署名するよう DFHWSSE に指示して、メッセージに署名する方法に関する情報を提供します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

```
<dfwsse_configuration>
```

内容

1. SOAP メッセージの本文に署名するために使用されるアルゴリズムを特定する URI を含む <algorithm> エレメント。

次のアルゴリズムを指定できます。

アルゴリズム	URI
Digital Signature Algorithm と Secure Hash Algorithm 1 (DSA と SHA1) インバウンド SOAP メッセージでのみサポートされます。	http://www.w3.org/2000/09/xmldsig#dsa-sha1
Rivest-Shamir-Adleman アルゴリズムと Secure Hash Algorithm 1 (RSA と SHA1)	http://www.w3.org/2000/09/xmldsig#rsa-sha1

2. RACF にインストールされたデジタル証明書に関連付けられたラベルを指定する `<certificate_label>` エlement。デジタル証明書は、メッセージへの署名に使用される鍵を提供します。

例

```
<sign_body>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>SIGCERT01</certificate_label>
</sign_body>
```

<encrypt_body>Element

アウトバウンド SOAP メッセージの本文を暗号化するよう DFHWSSE に指示して、メッセージを暗号化する方法に関する情報を提供します。

使用先

- サービス・プロバイダー
- サービス・リクエスター

格納元

`<dfhwsse_configuration>`

内容

1. SOAP メッセージの本文を暗号化するために使用される、アルゴリズムを特定する URI を含む `<algorithm>` Element。

次のアルゴリズムを指定できます。

アルゴリズム	URI
Triple Data Encryption Standard algorithm (Triple DES)	http://www.w3.org/2001/04/xmenc#tripledes-cbc
Advanced Encryption Standard (AES) アルゴリズム (鍵の長さは 128 ビット)	http://www.w3.org/2001/04/xmenc#aes128-cbc
Advanced Encryption Standard (AES) アルゴリズム (鍵の長さは 192 ビット)	http://www.w3.org/2001/04/xmenc#aes192-cbc

アルゴリズム	URI
Advanced Encryption Standard (AES) アルゴリズム (鍵の長さは 256 ビット)	http://www.w3.org/2001/04/xmlenc#aes256-cbc

2. RACF 内のデジタル証明書に関連付けられたラベルを指定する
 <certificate_label> エレメント。デジタル証明書は、メッセージの暗号化に使用される鍵を提供します。

例

```
<encrypt_body>
  <algorithm>http://www.w3.org/2001/04/xmlenc#aes256-cbc</algorithm>
  <certificate_label>ENCCERT02</certificate_label>
</encrypt_body>
```

メッセージ・ハンドラー

メッセージ・ハンドラーとは、入力時に Web サービスの要求を処理し、出力時に応答を処理するために使用する CICS プログラムのことです。メッセージ・ハンドラーは、メッセージ・ハンドラー同士の対話やシステムとの対話のために、チャンネルおよびコンテナを使用します。

メッセージ・ハンドラー・インターフェースを使用すると、メッセージ・ハンドラー・プログラム内部で以下のタスクを実行できます。

- XML 要求または応答の内容を、内容を変更せずに調べる
- XML 要求または応答の内容を変更する
- 端末以外のメッセージ・ハンドラーの場合は、XML 要求または応答をパイプライン内の次のメッセージ・ハンドラーに渡す
- 端末のメッセージ・ハンドラーの場合は、アプリケーション・プログラムを呼び出して、応答を生成する
- パイプラインの要求段階では、要求を吸収し、応答を生成することによって応答段階への移行を強制する
- ハンドル・エラー

ヒント: CICS 提供の SOAP 1.1 ハンドラーおよび SOAP 1.2 ハンドラーを使用して SOAP メッセージを処理することをお勧めします。これらのハンドラーを使用すると、SOAP メッセージ (SOAP ヘッダーおよび SOAP 本体) の主要なエレメントを直接処理できます。

メッセージ・ハンドラーとして使用されるすべてのプログラムは、同じインターフェースによって呼び出されます。これらのプログラムは、コンテナの数を保持するチャンネルによって呼び出されます。コンテナは次のように分類できます。

制御コンテナ

これらのコンテナは、パイプラインの運用に不可欠です。メッセージ・ハンドラーは、制御コンテナを使用して後続のハンドラーの処理順序を変更できます。

コンテキスト・コンテナ

状況によっては、メッセージ・ハンドラー・プログラムが呼び出されるコン

テキストについての情報がメッセージ・ハンドラー・プログラムに必要です。CICS は、プログラムに渡される一連のコンテキスト・コンテナにこの情報を提供します。

コンテキスト・コンテナの一部には、メッセージ・ハンドラーで変更できる情報が格納されます。例えば、サービス・プロバイダー・パイプラインで、ターゲット・アプリケーション・プログラムのユーザー ID やトランザクション ID を変更するには、該当するコンテキスト・コンテナの内容を変更します。

ユーザー・コンテナ

ここには、あるメッセージ・ハンドラーが別のメッセージ・ハンドラーに渡す必要がある情報が格納されます。ユーザー・コンテナの使用は、全面的にメッセージ・ハンドラーに委ねられます。

制約事項: ユーザー・コンテナでは、DFH で始まる名前を使用しないでください。

メッセージ・ハンドラー・プロトコル

パイプラインのメッセージ・ハンドラーは、要求メッセージと応答メッセージを処理します。メッセージ・ハンドラーの動作は、特定の状況でメッセージ・ハンドラーが可能な動作を記述する一連のプロトコルによって管理されます。

パイプライン内の各非端末メッセージ・ハンドラーは、2 度呼び出されます。

1. 最初は、要求 (サービス・プロバイダー・パイプラインではインバウンド要求、サービス・リクエスターではアウトバウンド要求) を処理するために呼び出されます。
2. 2 度目は、以下の 3 つのいずれかの理由で呼び出されます。
 - 応答 (サービス・プロバイダー・パイプラインではアウトバウンド応答、サービス・リクエスターではインバウンド応答) を処理するため。
 - パイプラインの他の場所でのエラーの後のリカバリーのため。
 - 応答がない場合に必要その後の処理を実行するため。

サービス・プロバイダー・パイプラインの端末メッセージ・ハンドラーは、要求の処理のため、一度呼び出されます。

メッセージ・ハンドラーがパイプラインに用意される理由はさまざまであり、各ハンドラーが実行する処理は大幅に異なる場合があります。特に、以下の場合が該当します。

- 一部のメッセージ・ハンドラーはメッセージの内容を変更せず、パイプラインの通常の処理シーケンスも変更しません。
- 一部のメッセージ・ハンドラーは、メッセージの内容は変更しますが、パイプラインの通常の処理シーケンスは変更しません。
- 一部のメッセージ・ハンドラーは、パイプラインの処理シーケンスを変更します。

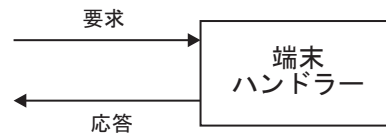
各ハンドラーは、実行できる処理を選択できます。選択の内容は、以下の条件に依存します。

- ハンドラーの呼び出し元はサービス・プロバイダーとサービス・リクエスターのどちらであるか
- サービス・プロバイダーの場合、ハンドラーは端末ハンドラーかどうか
- ハンドラーの呼び出し対象は要求メッセージと応答メッセージのどちらであるか

端末ハンドラーのプロトコル

通常の要求および応答

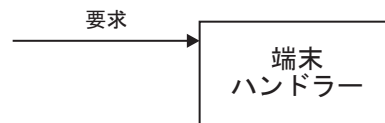
これは、端末ハンドラーの通常のプロトコルです。このハンドラーは、要求メッセージを対象として呼び出され、応答を作成します。



応答を作成するため、標準的な端末ハンドラーはターゲット・アプリケーション・プログラムにリンクしますが、これは必須ではありません。

通常の要求、応答なし

これは、端末ハンドラーのもう 1 つの一般プロトコルです。



このプロトコルは、通常、ターゲット・アプリケーションが要求に対して応答がないと判断した場合に検出されます (ただし、判断は端末ハンドラーで実行される場合もあります)。

非端末ハンドラーのプロトコル

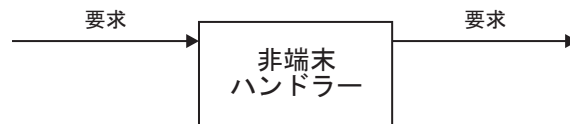
通常の要求および応答

これは、非端末ハンドラーの通常のプロトコルです。このハンドラーは、要求メッセージと応答メッセージの両方を対象として呼び出されます。どちらの場合も、ハンドラーはメッセージを処理し、パイプラインの次のハンドラーにメッセージを渡します。



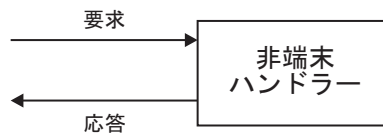
通常の要求、応答なし

これは、非端末ハンドラーのもう 1 つの一般的プロトコルです。このハンドラーは、要求メッセージを対象として呼び出され、メッセージの処理後、パイプラインの次のハンドラーに渡します。ターゲット・アプリケーション (または別のハンドラー) は、応答がないと判別します。このハンドラーが 2 度目に呼び出されたときに、処理する応答メッセージはありません。



ハンドラーが応答を作成する

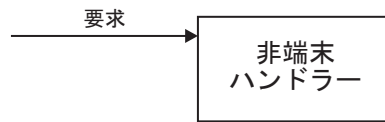
非端末ハンドラーは要求を次のハンドラーに渡さないため、このプロトコルは、通常、異常な状況で使用されます。その代わりに、このプロトコルでは応答が構成され、パイプラインに戻されます。



このプロトコルを使用できるのは、要求が何らかの意味で無効になり、要求がこれ以上処理されなくなるとハンドラーが判別した場合です。この状況では、ハンドラーが 2 回呼び出されることはありません。

ハンドラーが応答を抑止する

非端末ハンドラーは要求を次のハンドラーに渡さないため、このプロトコルは、通常、異常な状況で使用されるプロトコルとは別のプロトコルです。このプロトコルの場合、ハンドラーは要求に対する応答がないと判別します。



このプロトコルを使用できるのは、元の要求に対して応答が期待できなくなった場合と、要求が何らかの意味で無効になり、要求がこれ以上処理されない場合です。

独自のメッセージ・ハンドラーの提供

サービス・リクエスターとサービス・プロバイダーとの間でやり取りされるメッセージに対して特殊な処理を実行する際、この要望を満たすメッセージ・ハンドラーが CICS から提供されていない場合は、独自のメッセージ・ハンドラーを用意する必要があります。

このタスクについて

大半の状況では、CICS 提供のメッセージ・ハンドラーを使用することにより、必要なすべての処理を実行できます。例えば、SOAP メッセージを処理するには、CICS 提供の SOAP 1.1 メッセージ・ハンドラーおよび 1.2 メッセージ・ハンドラーを使用できます。ただし、Web サービス要求および応答に対して、独自の特殊な操作を実行することが必要になる場合があります。このためには、独自のメッセージ・ハンドラーを用意する必要があります。

1. メッセージ・ハンドラー・プログラムを作成します。メッセージ・ハンドラーとは、チャンネル・インターフェースを備えた CICS プログラムのことです。プログラムは、CICS がサポートしている任意の言語で記述でき、プログラム内部の DPL サブセットには任意の CICS コマンドを使用できます。
2. プログラムをコンパイルして、リンク・エディットします。メッセージ・ハンドラー・プログラムは通常、属性 TASKDATALOC(ANY) で定義されるトランザクション CPIH の下で実行されます。したがって、プログラムをリンク・エディットする際は、AMODE(31) オプションを指定する必要があります。
3. 通常の方法で CICS システムにプログラムをインストールします。
4. パイプライン構成ファイルでプログラムを定義します。メッセージ・ハンドラーを定義する場合は、<handler> エレメントを使用します。<handler> エレメントの内部には、プログラムの名前を格納した <program> エレメントを記述します。

端末以外のメッセージ・ハンドラーでのメッセージの処理

標準的な端末以外のメッセージ・ハンドラーは、メッセージを処理してから、パイプラインに存在する次のメッセージ・ハンドラーに制御を渡します。

このタスクについて

端末以外のメッセージ・ハンドラーの場合は、要求または応答を、その内容を変更するかまたは変更せずに処理し、次のメッセージ・ハンドラーに渡すことができます。

注: Web サービスでは、通常、XML を含む SOAP メッセージが使用されますが、メッセージ・ハンドラーは、その他のメッセージ・フォーマットの場合と同様に機能します。

1. コンテナ DFHFUNCTION の内容を使用して、このメッセージ・ハンドラーに渡されたメッセージが要求か応答かを判別します。

DFHFUNCTION	要求または応答	メッセージ・ハンドラーのタイプ	インバウンドまたはアウトバウンド
RECEIVE-REQUEST	要求	端末以外	インバウンド
SEND-RESPONSE	応答	端末以外	アウトバウンド
SEND-REQUEST	要求	端末以外	アウトバウンド
RECEIVE-RESPONSE	応答	端末以外	インバウンド

ヒント:

- DFHFUNCTION に PROCESS-REQUEST が格納されている場合、メッセージ・ハンドラーは端末メッセージ・ハンドラーであり、以下の手順は適用されません。
 - DFHFUNCTION に HANDLER-ERROR が格納されている場合、ハンドラーはエラー処理のために呼び出され、以下の手順は適用されません。
2. 適切なコンテナから要求または応答を取り出します。
 - メッセージが要求である場合、このメッセージはコンテナ DFHREQUEST に格納されてプログラムに渡されます。コンテナ DFHRESPONSE も存在しますが、長さはゼロです。
 - メッセージが応答である場合、このメッセージはコンテナ DFHRESPONSE に格納されてプログラムに渡されます。
 3. 必要なメッセージの処理を実行します。メッセージ・ハンドラーの目的に応じて、次のいずれかを実行できます。
 - 内容を変更せずにメッセージを調べ、パイプラインに存在する次のメッセージ・ハンドラーに渡します。
 - 要求を変更し、パイプラインに存在する次のメッセージ・ハンドラーに渡します。
 - メッセージが要求の場合は、パイプラインに存在する以降のメッセージ・ハンドラーをバイパスして、代わりに応答メッセージを作成できます。

注: これは、どのメッセージ・ハンドラーが次に呼び出されるかを判別する情報をメッセージ・ハンドラーが戻すコンテナの内容です。

メッセージ・ハンドラーが渡されたコンテナをまったく変更しない場合、エラーになります。

メッセージ・ハンドラー・プログラムが以下のいずれかを戻すと、エラーとなります。

- 空の DFHRESPONSE コンテナ。
- 空ではない DFHREQUEST コンテナおよび空ではない DFHRESPONSE コンテナ。
- アウトバウンド要求での空の DFHREQUEST コンテナ。

パイプラインに存在する次のメッセージ・ハンドラーへのメッセージの引き渡し

標準的な端末以外のメッセージ・ハンドラーの場合は、要求または応答を、その内容を変更するかまたは変更せずに処理し、次のメッセージ・ハンドラーに渡します。

1. メッセージを (変更するか、未変更のまま) 適切なコンテナにあるパイプラインに戻します。
 - メッセージが要求で、その内容を変更した場合は、そのメッセージをコンテナ DFHREQUEST に戻します。
 - メッセージが応答で、その内容を変更した場合は、そのメッセージをコンテナ DFHRESPONSE に書き込みます。
 - メッセージを変更しなかった場合、メッセージはすでに適切なコンテナに格納されています。
2. メッセージが要求の場合は、コンテナ DFHRESPONSE を削除します。 要求を処理するためにメッセージ・ハンドラーが呼び出されると、コンテナ DFHREQUEST および DFHRESPONSE はプログラムに渡されます。 DFHRESPONSE の長さはゼロです。ただし、DFHREQUEST と DFHRESPONSE の両方を戻すのは誤りです。

タスクの結果

メッセージは、パイプラインに存在する次のメッセージ・ハンドラーに渡されます。

パイプラインの応答段階への強制的な移行

要求の処理中には、パイプラインに存在する次のメッセージ・ハンドラーに要求を渡すのではなく、応答を速やかに生成するタイミングがあります。

1. コンテナ DFHREQUEST を削除します。
2. 応答を作成して、コンテナ DFHRESPONSE に書き込みます。

タスクの結果

応答は、パイプラインの応答段階で次のメッセージ・ハンドラーに渡されます。

応答の抑止

状況によっては、要求を受けるのみで応答を返信しないようにしたいことがあります。

1. コンテナ DFHREQUEST を削除します。
2. コンテナ DFHRESPONSE を削除します。

サービス・リクエスター・パイプラインでの片方向メッセージの処理

サービス・リクエスター・パイプラインがサービス・プロバイダーに要求を送信する場合、通常であれば、要求の送信後に応答があり、応答が到着すると、パイプライン内のメッセージ・ハンドラーが再び呼び出されるという期待があります。一部の Web サービスでは応答が送信されないため、2 回目にメッセージ・ハンドラーを呼び出す前に CICS が応答を待機しないことを示すために、特別な対策を講じる必要があります。

このタスクについて

このためには、コンテナ DFHNORESPONSE が要求段階のパイプライン処理の最後に存在することを確認します。通常、この処理はアプリケーション・レベルのコードで実行されます。これは、応答が期待されるかどうかの認識はアプリケーションにとどまるためです。

- CICS Web サービス・アシスタントによって配置されたアプリケーションの場合、CICS コードがコンテナを作成します。
- Web サービス・アシスタントによって配置されなかったアプリケーションは、通常、アプリケーションを呼び出す前にコンテナを作成します。

メッセージ・ハンドラーでコンテナ DFHNORESPONSE を作成または破棄する場合は、こうすることでサービス・リクエスターとサービス・プロバイダー間のメッセージ・プロトコルを妨害しないことを確認する必要があります。

端末メッセージ・ハンドラーでのメッセージの処理

標準的な端末ハンドラーは、要求を処理し、アプリケーション・プログラムを呼び出して、応答を生成します。

このタスクについて

注: Web サービスでは、通常、XML を含む SOAP メッセージが使用されますが、メッセージ・ハンドラーは、その他のメッセージ・フォーマットの場合と同様に機能します。

端末メッセージ・ハンドラーでは、要求を処理できます。また、必要に応じて応答を生成し、パイプラインをたどって戻すこともできます。標準的な端末ハンドラーでは、要求がアプリケーション・プログラムへの入力として使用され、アプリケーション・プログラムの応答を使用して応答が作成されます。

1. コンテナ DFHFUNCTION の内容を使用して、このハンドラーに渡されたメッセージが要求であることと、このハンドラーは端末ノードとして呼び出されていることを確認します。

DFHFUNCTION	要求または応答	ハンドラーのタイプ	インバウンドまたはアウトバウンド
PROCESS-REQUEST	要求	端末	インバウンド

ヒント:

- DFHFUNCTION にその他の値が格納されている場合、このハンドラーは端末ハンドラーではなく、これらのステップは適用されません。
- 2. コンテナ DFHREQUEST から要求を取り出します。 コンテナ DFHRESPONSE も存在しますが、長さはゼロです。
- 3. 必要なメッセージの処理を実行します。 通常、端末ハンドラーはアプリケーション・プログラムを呼び出します。
- 4. 応答を作成して、コンテナ DFHRESPONSE に書き込みます。 応答がない場合は、コンテナ DFHRESPONSE を削除する必要があります。

タスクの結果

応答は、パイプラインの応答段階で次のハンドラーに渡されます。ハンドラーは、機能 SEND-RESPONSE のために呼び出されます。応答がない場合は、次のハンドラーが機能 NO-RESPONSE のために呼び出されます。

エラーの処理

メッセージ・ハンドラーは、パイプラインで発生する可能性のあるエラーを処理する目的で設計する必要があります。

このタスクについて

メッセージ・ハンドラー・プログラムでエラーが発生すると、このプログラムはエラー処理のためにもう一度呼び出されます。パイプラインの応答段階では、エラー処理が必ず発生します。要求段階でエラーが発生すると、要求段階の後続のハンドラーはバイパスされます。

したがって大半の場合は、発生する可能性があるすべてのエラーを処理するためにハンドラー・プログラムを記述する必要があります。

1. コンテナ DFHFUNCTION に、エラー処理のためにメッセージ・ハンドラーが呼び出されたことを示す HANDLER-ERROR が格納されていることを確認します。

ヒント:

- DFHFUNCTION に他の値が格納されている場合は、このメッセージ・ハンドラーがエラー処理のために呼び出されることはなく、これらのステップは適用されません。
- 2. エラー情報を分析し、適切な応答を作成することにより、メッセージ・ハンドラーがエラー状態から復旧できるかどうかを判断します。

コンテナ DFHERROR には、以下のエラーに関する情報が保持されます。このコンテナについて詳しくは、126 ページの『DFHERROR コンテナ』を参照してください。

コンテナ DFHRESPONSE も存在しますが、長さはゼロです。

3. リカバリー処理を実行します。
 - メッセージ・ハンドラーが回復できる場合は、応答を作成して、コンテナ DFHRESPONSE に応答を戻します。

- メッセージ・ハンドラーは回復できるが、応答は必要ない場合は、コンテナ DFHRESPONSE を削除し、代わりにコンテナ DFHNORESPONSE に戻ります。
- メッセージ・ハンドラーが回復できない場合は、コンテナ DFHRESPONSE を未変更状態 (つまり、長さゼロ) に戻します。

タスクの結果

メッセージ・ハンドラーがエラーから回復できる場合は、パイプライン処理は正常に続行されます。回復できない場合は、CICS は、エラーに関する情報が含まれた SOAP 障害を生成します。トランザクションが異常終了する場合は、障害に異常終了コードが含まれます。

メッセージ・ハンドラー・インターフェース

CICS パイプラインは、多数のコンテナを内蔵するチャンネルを使用して、メッセージ・ハンドラーにリンクします。コンテナには、オプションのコンテナもあれば、すべてのメッセージ・ハンドラーが必要とするコンテナもあります。また、一部のメッセージ・ハンドラーが使用し、それ以外のハンドラーは使用しないコンテナもあります。

ハンドラーが呼び出される前に、一部またはすべてのコンテナには、ハンドラーがその作業を実行するために使用できる情報が取り込まれます。後続の処理は、ハンドラーによって戻されたコンテナによって決定し、このコンテナはパイプラインのその後のハンドラーに渡されます。

SOAP メッセージ・ハンドラー

SOAP メッセージ・ハンドラーは、パイプラインに組み込んで SOAP 1.1 メッセージおよび SOAP 1.2 メッセージを処理できる、CICS 提供のメッセージ・ハンドラーです。SOAP メッセージ・ハンドラーは、サービス・リクエスター・パイプラインまたはサービス・プロバイダー・パイプラインで使用できます。

入力では、SOAP メッセージ・ハンドラーがインバウンド SOAP メッセージを解析し、アプリケーション・プログラムによる使用に備えて SOAP <Body> エlement を抽出します。出力では、アプリケーションによって提供される <Body> Element を使用して、ハンドラーが完全な SOAP メッセージを作成します。

メッセージに SOAP ヘッダーを使用すると、SOAP ハンドラーは、インバウンド・メッセージでヘッダーを処理し、アウトバウンド・メッセージでヘッダーを追加できる、ユーザー作成のヘッダー処理プログラムを呼び出すことができます。

SOAP メッセージ・ハンドラーおよびすべてのヘッダー処理プログラムは、<cics_soap_1.1_handler> Element および <cics_soap_1.2_handler> Element、およびそのサブElementを使用して、パイプライン構成ファイルで指定されます。

通常、1 つのパイプラインに必要な SOAP ハンドラーは 1 つだけです。ただし、状況によっては、複数の SOAP ハンドラーが必要です。例えば、複数の SOAP ハンドラーを定義すれば、SOAP ヘッダーを特定の順序で処理できるようになります。

ヘッダー処理プログラム

ヘッダー処理プログラムとは、SOAP ヘッダー・ブロックを処理するために、CICS 提供の SOAP 1.1 および SOAP 1.2 メッセージ・ハンドラーにリンクされているユーザー作成 CICS プログラムのことです。

ヘッダー処理プログラムは、CICS がサポートしている任意の言語で記述でき、DPL サブセットに任意の CICS コマンドを使用できます。ヘッダー処理プログラムは、他の CICS プログラムとリンクできます。

ヘッダー処理プログラムには、チャンネル・インターフェースがあります。コンテナーには、ヘッダー処理プログラムによって検査または変更できる以下の情報が保持されます。

プログラムの呼び出し対象となる SOAP ヘッダー・ブロック

SOAP メッセージの本文

このインターフェースと、ヘッダー処理プログラムが使用できるコンテナーについては、121 ページの『ヘッダー処理プログラム・インターフェース』で説明します。

別のコンテナーには、ヘッダー・プログラムの呼び出し元の環境について以下のような情報が保持されます。

ヘッダー・プログラムの呼び出しに使用されたトランザクション ID

プログラムの呼び出し元がサービス・プロバイダーと要求側パイプラインのいずれであるか

処理対象のメッセージは要求と応答のいずれであるか

ヘッダー処理プログラムは、通常トランザクション CPIH の下で実行されます。CPIH は属性 TASKDATALOC(ANY) で定義されます。したがって、プログラムをリンク・エディットする際は、AMODE(31) オプションを指定する必要があります。

SOAP 要求に対するヘッダー処理プログラムの呼び出し方法

パイプライン構成内の `<cics_soap_1.1_handler>` エレメントおよび `<cics_soap_1.2_handler>` エレメントには、0 または 1 つ以上の `<headerprogram>` エレメントが格納されており、このエレメントのそれぞれには、以下の子が格納されています。

`<program_name>`

`<namespace>`

`<localname>`

`<mandatory>`

パイプラインがインバウンド SOAP メッセージ (サービス・プロバイダーでは要求、サービス・リクエスターでは応答) を処理している場合、`<program_name>` エレメントで指定されたヘッダー・プログラムが呼び出されるかどうかは、以下の内容によって決まります。

- `<namespace>`、`<localname>`、および `<mandatory>` エレメントの内容

- SOAP ヘッダー自体のルート・エレメントの特定の属性の値 (SOAP 1.1 の場合は **actor** 属性、SOAP 1.2 の場合は **role** 属性)

以下の規則では、ヘッダー・プログラムが特定の場合に呼び出されるかどうかが判別されます。

パイプライン構成ファイル内の **<mandatory>** エレメント

エレメントに **true** (つまり 1) が含まれる場合は、その他の規則によって処理のために選択される SOAP メッセージのヘッダーが存在しない場合でも、ヘッダー処理プログラムが 1 回以上呼び出されます。

- 選択されたヘッダー・ブロックがない場合、ヘッダー処理プログラムは 1 回呼び出されます。
- その他の規則によっていずれかのヘッダー・ブロックが選択されると、ヘッダー処理プログラムは、選択されたヘッダーごとに 1 回呼び出されます。

SOAP ヘッダー・ブロック内の属性

SOAP 1.1 の場合、ヘッダー・ブロックは、**actor** 属性が存在しない場合、または `http://schemas.xmlsoap.org/soap/actor/next` の値が存在する場合にのみ、処理について適格です。

SOAP 1.2 の場合、ヘッダー・ブロックは、**role** 属性が存在しない場合、または以下のいずれかの値が存在する場合にのみ、処理について適格です。

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

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

処理について適格なヘッダー・ブロックは、次の規則によって選択されるまで処理されません。

パイプライン構成ファイル内の **<namespace> エレメントおよび **<localname>** エレメント** 前の規則に基づく処理について適格なヘッダー・ブロックは、次の場合にのみ処理のために選択されます。

- ヘッダー・ブロックのルート・エレメントの名前が、パイプライン構成ファイルの **<localname>** エレメントと一致する場合
- かつ ルート・エレメントのネーム・スペースが、パイプライン構成ファイルの **<namespace>** エレメントと一致する場合

例えば、次のヘッダー・ブロックの場合を考えます。

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

パイプライン構成ファイルに以下のコードを記述すると、他の規則に従って、処理のためにヘッダー・ブロックが選択されます。

```
<namespace>http://mynamespace</namespace>
<localname>myheaderblock</localname>
```

<localname> に ***** を記述すると、ネーム・スペース内のすべてのヘッダー・ブロックを処理することを指定できます。したがって、次のコードを記述すると、同じヘッダー・ブロックを選択できます。

```
<namespace>http://mynamespace</namespace>
<localname>*</localname>
```


SOAP メッセージに複数のヘッダーが含まれる場合は、一致するヘッダーがあるたびにヘッダー処理プログラムが 1 回呼び出されますが、ヘッダーの処理順序は定義されていません。

CICS 提供の SOAP メッセージ・ハンドラーは、SOAP メッセージを受信すると、その内部に存在するヘッダー・ブロックに基づいて呼び出されるヘッダー処理プログラムを選択します。従って、ヘッダー処理プログラムは、SOAP メッセージ・ハンドラー内にあるメッセージに追加されるヘッダー・ブロックの結果として呼び出されることはありません。パイプライン内で新規のヘッダー（または任意の変更済みヘッダー）を処理する場合は、パイプライン内に別の SOAP メッセージ・ハンドラーを定義する必要があります。

アウトバウンド・メッセージの場合（サービス・リクエスターでは要求、サービス・プロバイダーでは応答）、CICS 提供の SOAP メッセージ・ハンドラーはヘッダーを含まない SOAP メッセージを作成します。メッセージに 1 つ以上のヘッダーを追加するためには、ヘッダー・ハンドラー・プログラムを記述してヘッダーを追加する必要があります。このヘッダー・ハンドラーを確実に呼び出すようにするには、パイプライン構成ファイルでヘッダー・ハンドラーを定義し、`<mandatory>true</mandatory>` を指定する必要があります。

パイプラインの要求段階でヘッダー・ハンドラーが呼び出される場合、応答段階で出されるメッセージに、対応するヘッダーが含まれていない場合でも、応答段階でこのヘッダー・ハンドラーが再度呼び出されます。

ヘッダー処理プログラム・インターフェース

CICS 提供の SOAP 1.1 および SOAP 1.2 メッセージ・ハンドラーは、チャンネル DFHHHC-V1 を使用してヘッダー処理プログラムにリンクします。チャンネル上で渡されるコンテナには、いくつかのヘッダー処理プログラム・インターフェースに固有のコンテナと、パイプライン内のすべてのヘッダー処理プログラムおよびメッセージ・ハンドラーでアクセス可能な一連のコンテキスト・コンテナ およびユーザー・コンテナ があります。

コンテナ DFHHEADER は、ヘッダー処理プログラム・インターフェースに固有のコンテナです。その他のコンテナは、パイプライン内の他の場所で使用することができますが、ヘッダー処理プログラムでは特定の使用方法があります。このカテゴリーのコンテナには、次のものがあります。

DFHWS-XMLNS

DFHWS-BODY

DFHXMLSS-PARSE

コンテナ DFHHEADER

ヘッダー処理プログラムが呼び出された場合、DFHHEADER には、ヘッダー処理プログラムを駆動する単一のヘッダー・ブロックが格納されています。ヘッダー・プログラムを、パイプライン構成ファイルで `<mandatory>true</mandatory>` または `<mandatory>1</mandatory>` と共に指定すると、SOAP メッセージ内に一致するヘッダー・ブロックがない場合でも、このヘッダー・プログラムが呼び出されます。この場合、コンテナ DFHHEADER の長さはゼロになります。ヘッダー・ブロック

を持たない SOAP メッセージにヘッダー・ブロックを追加するためにヘッダー処理プログラムを呼び出す場合に、このようになります。

CICS が作成する SOAP メッセージには最初はヘッダーはありません。メッセージにヘッダーを追加したい場合は、`<mandatory>true</mandatory>` または `<mandatory>1</mandatory>` を指定することにより、少なくとも 1 つのヘッダー処理プログラムが呼び出されるようにする必要があります。

ヘッダー・プログラムが戻った場合、コンテナ DFHHEADER には、以下に示すようにヘッダー・ブロックがゼロまたは 1 つ以上格納されている必要があります。このヘッダー・ブロックは、元のヘッダー・ブロックの代わりに CICS が SOAP メッセージに挿入したものです。

- 元のヘッダー・ブロックを未変更のまま戻すことができます。
- ヘッダー・ブロックの内容を変更できます。
- 元のヘッダー・ブロックに 1 つ以上の新規ヘッダー・ブロックを追加できます。
- 元のヘッダー・ブロックを、1 つ以上の異なるブロックと置換できます。
- ヘッダー・ブロックを完全に削除できます。

コンテナ DFHWS-XMLNS

ヘッダー処理プログラムが呼び出された場合、DFHWS-XMLNS には、SOAP エンベロープ内で宣言された XML ネーム・スペースに関する情報が格納されています。ヘッダー・プログラムは、以下のことを目的としてこの情報を使用できます。

- ヘッダー・ブロック内で検出した修飾名を解決する
- 新規または変更したヘッダー・ブロックで修飾名を構成する

ネーム・スペース情報は、ネーム・スペースを宣言するための標準の XML 表記を使用している、ネーム・スペース宣言のリストで構成されます。DFHWS-XMLNS のネーム・スペース宣言は、スペースで区切られます。例を次に示します。

```
xmlns:na='http://abc.example.org/schema' xmlns:nx='http://xyz.example.org/schema'
```

ネーム・スペース宣言を DFHWS-XMLNS の内容に追加すれば、ネーム・スペース宣言を SOAP エンベロープにさらに追加できます。ただし、有効範囲が SOAP ヘッダー・ブロックまたは SOAP 本体であるネーム・スペースは、それぞれヘッダー・ブロックまたは本体で宣言するのが最適です。ヘッダー処理プログラムでは、コンテナ DFHWS-XMLNS からネーム・スペース宣言を削除しないことをお勧めします。このプログラムでは表示されない XML エlement がネーム・スペース宣言に依存する場合があるからです。

コンテナ DFHWS-BODY

SOAP エンベロープの本体部分を格納します。ヘッダー処理プログラムは、内容を変更する場合があります。

ヘッダー処理プログラムが呼び出された場合、DFHWS-BODY には、SOAP `<Body>` Element が格納されています。

ヘッダー・プログラムが戻った場合、コンテナ DFHWS-BODY には、以下に示すように有効な SOAP <Body> が再度格納されている必要があります。この SOAP 本体は、元の SOAP 本体の代わりに CICS が SOAP メッセージ内に挿入したものです。

- 元の本体を未変更のまま戻すことができます。
- 本体の内容を変更できます。

各 SOAP メッセージには <Body> エレメントが格納されている必要があるため、SOAP 本体を完全に削除することはできません。

コンテナ DFHXMLSS-PARSE

ヘッダー・プログラムが呼び出されるとき、DFHXMLSS-PARSE にはそのヘッダーの XML システム・サービス (XMLSS) レコードが含まれます。

コンテキスト・コンテナおよびユーザー・コンテナ

ここで説明したコンテナと同様に、インターフェースは、チャンネル DFHHHC-V1 上で 制御コンテナ、コンテキスト・コンテナ、およびユーザー・コンテナを渡します。

これらのコンテナについて詳しくは、125 ページの『パイプラインで使用されるコンテナ』を参照してください。

SOAP ハンドラー・インターフェース

SOAP ハンドラーには、ユーザー作成プログラムを接続する 2 つのインターフェースがあります。ヘッダー処理プログラム・インターフェースは、SOAP ハンドラーとヘッダー処理プログラム間で情報を受け渡します。アプリケーション・インターフェースは、SOAP ハンドラーとターゲット・アプリケーション間で情報を受け渡します。

アプリケーション・インターフェース

アプリケーション・インターフェースは、ターゲット・アプリケーションがチャンネル・インターフェースで呼び出される際に、SOAP ハンドラーとターゲット・アプリケーション・プログラムの間で受け渡されるチャンネルです。ターゲットが COMMAREA インターフェースで呼び出される場合、チャンネルはターゲット・アプリケーション・プログラムでは使用できません。

アプリケーション・インターフェースが使用するチャンネル (DFHAHC-V1) は、以下のコンテナを受け渡します。

DFHWS-XMLNS

ネーム・スペースの接頭部をネーム・スペースにマップする名前と値のペアのリストを格納します。

- 入力時には、このリストに有効範囲にある SOAP エンベロープ内のネーム・スペースが格納されています。
- 出力時には、このリストにエンベロープ・タグ内にあると見なされるネーム・スペース・データが格納されています。

DFHWS-BODY

SOAP エンベロープの本体部分を格納します。通常、アプリケーションは内容を変更します。アプリケーションが内容を変更しない場合は、アプリケーション・ハンドラー・プログラムがこのコンテナの内容を更新する必要があります。端末ハンドラーに戻る前に同じ内容をコンテナに戻す場合でも、同様に更新する必要があります。

DFHNORESPONSE

サービス・リクエスター・パイプラインの要求段階では、サービス・プロバイダーが応答を戻すことを期待できないことを示します。コンテナ

DFHNORESPONSE の内容は定義されていません。サービス・プロバイダーが応答を戻すことが見込まれるかどうかを認識する必要のあるメッセージ・ハンドラーは、コンテナが存在するかどうかのみを判別する必要があります。

- コンテナ DFHNORESPONSE が存在する場合は、応答は見込まれません。
- コンテナ DFHNORESPONSE が不在の場合は、応答が見込まれます。

チャンネルは、呼び出し側メッセージ・ハンドラーに渡されたすべてのコンテキスト・コンテナも同様に渡します。ヘッダー処理プログラムはチャンネルにコンテナを追加することができます。追加されたコンテナはユーザー・コンテナとして、パイプライン内の次のハンドラーに渡されます。

端末ハンドラーでのインバウンド要求の動的ルーティング

サービス・プロバイダー・パイプラインの端末ハンドラーが CICS 提供の SOAP メッセージ・ハンドラーの 1 つであるとき、コンテナ DFHWS-APPHANDLER に指定されるターゲット・アプリケーションのハンドラー・プログラムが、動的ルーティングに適格である場合があります。アプリケーション・ハンドラー・プログラムより前のパイプライン処理はすべて、常に SOAP メッセージを受け取った CICS 領域でローカルに実行されます。

ターゲット・アプリケーションのハンドラー・プログラムを実行するトランザクションは、以下のいずれかが真である場合に、ルーティングに適格です。

- パイプラインがメッセージを処理するトランザクションが、DYNAMIC または REMOTE として定義される。このトランザクションは、インバウンド SOAP メッセージからの URI のマップに使用される URIMAP に定義されます。
- パイプラインのプログラムが、コンテナ DFHWS-USERID の内容を初期値から変更した。
- パイプラインのプログラムが、コンテナ DFHWS-TRANID の内容を初期値から変更した。
- WS-AT SOAP ヘッダーがインバウンド SOAP メッセージ内にある。

上記のすべてのシナリオで、パイプラインの処理中にタスク切り替えが起こります。2 番目のタスクは、DFHWS-TRANID コンテナに指定されたトランザクションの下で実行します。このタスク切り替えによって動的ルーティングが起こる機会が生じますが、CICS 領域同士の接続に MRO が使用される場合に限りです。さらに、ルーティング先の CICS 領域が、チャンネルおよびコンテナをサポートする必要があります。

DFHWS-TRANID に指定されたトランザクションの TRANSACTION 定義が以下の一連の属性のいずれかを指定する場合にのみ、ルーティングが起こります。

DYNAMIC(YES)

トランザクションは、ルーティング・プログラムが **DSRTPGM** システム初期設定パラメーターに指定された、分散ルーティング・モデルを使用してルーティングされます。

DYNAMIC(NO) REMOTESYSTEM(sysid)

トランザクションは、*sysid* で識別されるシステムにルーティングされます。

詳しくは、「*CICS Customization Guide*」を参照してください。

CICS Web サービス・アシスタントを使用して配置されたアプリケーションでは、CICS がユーザー・プログラムにリンクする際に、要求を動的にルーティングする 2 番目の機会があります。この時点で、要求は、ルーティング・プログラムが **DTRPGM** システム初期設定パラメーターに指定された、動的ルーティング・モデルを使用してルーティングされます。このケースでは、プログラムの特性によってルーティングが適格であると判断されます。プログラムにリンクする際にチャンネルおよびコンテナを使用する場合は、V3.1 以上の CICS 領域にのみ要求を動的にルーティングできます。COMMAREA を使用する場合は、この制限は適用されません。

詳しくは、「*CICS Customization Guide*」を参照してください。

パイプラインで使用されるコンテナ

一般に、パイプラインは、いくつかのメッセージ・ハンドラー・プログラムから構成されます。CICS 提供の SOAP メッセージ・ハンドラーが使用される場合は、いくつかのヘッダー処理プログラムから構成されます。CICS は、コンテナを使用してこれらのプログラムとの間で情報を受け渡します。各プログラムは、パイプライン内の他のプログラムとの通信にもコンテナを使用します。

CICS のパイプラインは、いくつかのコンテナから成るチャンネルを使用して、メッセージ・ハンドラーや、ヘッダー処理プログラムとリンクします。コンテナには、オプションのコンテナもあれば、すべてのメッセージ・ハンドラーが必要とするコンテナもあります。また、一部のメッセージ・ハンドラーが使用し、それ以外のハンドラーは使用しないコンテナもあります。

ハンドラーが呼び出される前に、一部またはすべてのコンテナには、ハンドラーがその作業を実行するために使用できる情報が取り込まれます。後続の処理は、ハンドラーによって戻されたコンテナによって決定し、このコンテナはパイプラインのその後のハンドラーに渡されます。

次のようにコンテナを分類することができます。

制御コンテナ

このコンテナは、パイプラインの運用に不可欠です。ハンドラーは制御コンテナを使用してハンドラーの処理順序を変更できます。制御コンテナの名前は CICS によって定義されます。名前が DFH という文字で始まります。

コンテキスト・コンテナ

このコンテナは、ハンドラーが呼び出された環境に関する情報が格納されます。CICS はこれらのコンテナに情報を書き込んでから最初のメッセージ・ハンドラーを呼び出しますが、場合によっては、ハンドラーが内容の変更やコンテナの削除を自由に実行できます。コンテキスト・コンテナの変更が、ハンドラーの呼び出し順序に直接影響することはありません。コンテキスト・コンテナの名前は CICS によって定義されます。名前が DFH という文字で始まります。

ヘッダー処理プログラムのコンテナ

このコンテナには、CICS 提供の SOAP メッセージ・ハンドラーから呼び出されたヘッダー処理プログラムが使用する情報が格納されます。

セキュリティ・コンテナ

このコンテナは、Trust クライアント・インターフェースとセキュリティ・メッセージ・ハンドラーが、Security Token Service (STS) を使用してセキュリティ・トークンを処理するために使用する情報が含まれます。セキュリティ・コンテナの名前は CICS によって定義されます。名前は DFH という文字で始まります。

生成されたコンテナ

このコンテナは、処理のためにアプリケーション・プログラムとの間で受け渡しされる、変数配列や長ストリングなどの SOAP メッセージからのデータが含まれます。CICS は、パイプライン処理中にこれらのコンテナを自動的に作成し、名前は DFH という文字で始まります。

ユーザー・コンテナ

このコンテナは、あるメッセージ・ハンドラーが別のメッセージ・ハンドラーに渡す必要がある情報が格納されます。ユーザー・コンテナの使用は、全面的にメッセージ・ハンドラーに委ねられます。これらのコンテナには独自の名前を選択することができますが、DFH で始まる名前は使用できません。

制御コンテナ

制御コンテナは、パイプラインの操作に不可欠です。ハンドラーは制御コンテナを使用してハンドラーの処理順序を変更できます。

DFHERROR コンテナ

DFHERROR は、パイプラインのエラーに関する情報を他のメッセージ・ハンドラーに伝達する、DATATYPE(BIT) のコンテナです。

表 4. DFHERROR コンテナの構造：構造内の全フィールドに文字データが含まれます。

フィールド名	長さ (バイト)	内容
PIISNEB-MAJOR-VERSION	1	『1』
PIISNEB-MINOR-VERSION	1	『1』
PIISNEB-ERROR-TYPE	1	エラーのタイプを示す数値。 値については 127 ページの表 5 で説明します。

表 4. DFHERROR コンテナの構造 (続き): 構造内の全フィールドに文字データが含まれます。

フィールド名	長さ (バイト)	内容
PIISNEB-ERROR-MODE	1	P プロバイダー・パイプラインで起こったエラー R リクエスター・パイプラインで起こったエラー T Trust クライアントで起こったエラー
PIISNEB-ABCODE	4	エラーがトランザクションの異常終了に関連する場合の異常終了コード。
PIISNEB-ERROR-CONTAINER1	16	エラーがコンテナに関連する場合のコンテナの名前。
PIISNEB-ERROR-CONTAINER2	16	エラーが複数のコンテナに関連する場合の、2 番目のコンテナの名前。
PIISNEB-ERROR-NODE	8	エラーが発生したハンドラー・プログラムの名前。

表 5. PIISNEB-ERROR-TYPE フィールドの値

PIISNEB-ERROR-TYPE の値	意味
1	ハンドラー・プログラムは失敗しました。異常終了コードはフィールド PIISNEB-ABCODE にあります。
2	ハンドラーに必要なコンテナが空でした。コンテナの名前はフィールド PIISNEB-ERROR-CONTAINER1 にあります。
3	ハンドラーに必要なコンテナが欠落していました。コンテナの名前はフィールド PIISNEB-ERROR-CONTAINER1 にあります。
4	1 つのコンテナしか予想されていないときに、ハンドラーに 2 つのコンテナが渡されました。コンテナの名前はフィールド PIISNEB-ERROR-CONTAINER1 および PIISNEB-ERROR-CONTAINER2 にあります。
5	ターゲット・プログラムへのリンクに失敗しました。ターゲット・プログラムが失敗した場合、異常終了コードはコンテナ PIISNEB-ABCODE にあります。

表 5. PIISNEB-ERROR-TYPE フィールドの値 (続き)

PIISNEB-ERROR-TYPE の値	意味
6	基礎トランスポートのエラーのために、パイプライン・マネージャーがリモート・サーバーとの通信に失敗しました。
7	DFHWS-STSACTION コンテナにエラーがあります。欠落または破損しているか、間違った値が含まれています。
8	DFHPIRT はパイプラインの開始に失敗しました。
9	DFHPIRT はコンテナにメッセージを書き込もうとして失敗しました。
10	DFHPIRT はコンテナからメッセージを取得しようとして失敗しました。
11	未処理エラーが発生しました。

コンテナの構造の COBOL 宣言は、次のとおりです。

```
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).
```

コンテナと対応する言語コピーブックは次のとおりです。

表 6.

言語	コピーブック
COBOL	DFHPIUCO
PL/I	DFHPIUCL
C および C++	dfhpiuch.h
アセンブラー	DFHPIUCD

DFHFUNCTION コンテナ

DFHFUNCTION は、パイプライン内のどこでプログラムが呼び出されるかを示す 16 文字のストリングを格納する、DATATYPE(CHAR) のコンテナです。

このストリングには、次のいずれかの値が含まれます。右端の文字位置は、ブランク文字で埋め込まれます。

RECEIVE-REQUEST

このハンドラーはサービス・プロバイダー・パイプラインの端末以外のハンドラーで、インバウンド要求メッセージを処理するときに呼び出されます。ハンドラーへの入力時は、このメッセージは制御コンテナ DFHREQUEST に格納されています。

SEND-RESPONSE

このハンドラーはサービス・プロバイダー・パイプラインの端末以外のハンドラーで、アウトバウンド応答メッセージを処理するときに呼び出されます。ハンドラーへの入力時は、このメッセージは制御コンテナ DFHRESPONSE に格納されています。

SEND-REQUEST

このハンドラーは、要求を送信しているパイプラインによって呼び出されます。つまり、アウトバウンド・メッセージを処理しているサービス・リクエスターに存在します。

RECEIVE-RESPONSE

このハンドラーは、応答を受信しているパイプラインによって呼び出されます。つまり、インバウンド・メッセージを処理しているサービス・リクエスターに存在します。

PROCESS-REQUEST

このハンドラーは、サービス・プロバイダー・パイプラインの端末ハンドラーとして呼び出されます。

NO-RESPONSE

このハンドラーは、処理の対象となる応答が存在しない場合、要求の処理後に呼び出されます。

HANDLER-ERROR

このハンドラーは、エラーが検出されたために呼び出されます。

要求を処理し応答を戻すサービス・プロバイダー・パイプラインでは、発生する DFHFUNCTON の値は RECEIVE-REQUEST、PROCESS-REQUEST、および SEND-RESPONSE です。図 23 には、ハンドラーが呼び出される順序と、各ハンドラーに渡される DFHFUNCTON の値が示されています。

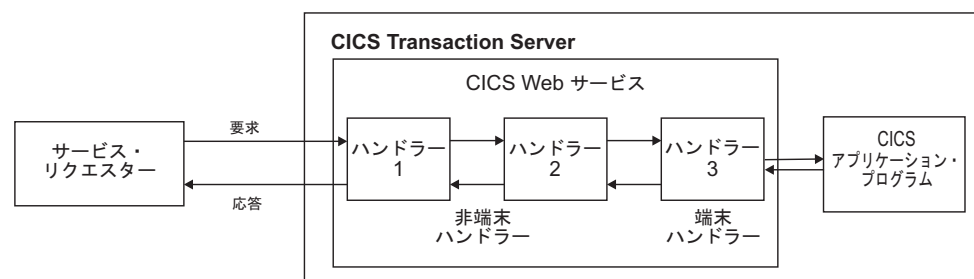


図 23. サービス・プロバイダー・パイプラインでのハンドラーの順序

順序	ハンドラー	DFHFUNCTON
1	ハンドラー 1	RECEIVE-REQUEST
2	ハンドラー 2	RECEIVE-REQUEST
3	ハンドラー 3	PROCESS-REQUEST
4	ハンドラー 2	SEND-RESPONSE
5	ハンドラー 1	SEND-RESPONSE

要求を送信し応答を受信するサービス・リクエスター・パイプラインでは、発生する DFHFUNCTION の値は SEND-REQUEST および RECEIVE-RESPONSE です。図 24 には、ハンドラーが呼び出される順序と、各ハンドラーに渡される DFHFUNCTION の値が示されています。

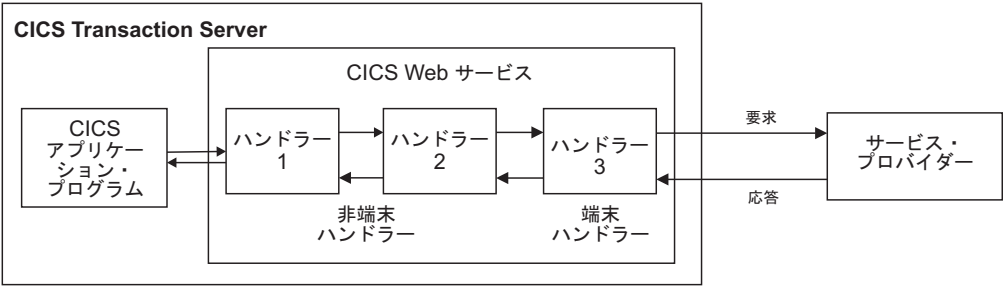


図 24. サービス・リクエスター・パイプラインでのハンドラーの順序

順序	ハンドラー	DFHFUNCTION
1	ハンドラー 1	SEND-REQUEST
2	ハンドラー 2	SEND-REQUEST
3	ハンドラー 3	SEND-REQUEST
4	ハンドラー 3	RECEIVE-RESPONSE
5	ハンドラー 2	RECEIVE-RESPONSE
6	ハンドラー 1	RECEIVE-RESPONSE

特定のメッセージ・ハンドラーで検出できる DFHFUNCTION の値は、パイプラインがプロバイダーであるかリクエスターであるか、パイプラインの要求段階であるか応答段階であるか、およびハンドラーが端末ハンドラーであるか端末以外のハンドラーであるかによって異なります。次の表に、それぞれの値が発生する場合をまとめます。

DFHFUNCTION の値	プロバイダー・パイプラインであるか、リクエスター・パイプラインであるか	パイプラインの段階	端末ハンドラーであるか、端末以外のハンドラーであるか
RECEIVE-REQUEST	プロバイダー	要求段階	端末以外
SEND-RESPONSE	プロバイダー	応答段階	端末以外
SEND-REQUEST	リクエスター	要求段階	端末以外
RECEIVE-RESPONSE	リクエスター	応答段階	端末以外
PROCESS-REQUEST	プロバイダー	要求段階	端末
NO-RESPONSE	両方	応答段階	端末以外
HANDLER-ERROR	両方	両方	両方

DFHHTTPSTATUS コンテナー

DFHHTTPSTATUS は、サービス・プロバイダー・パイプラインの応答段階で生成されるメッセージの HTTP 状況コードと状況テキストを指定するために使われる、DATATYPE(CHAR) のコンテナーです。

DFHHTTPSTATUS コンテナの内容は、以下のような構造の、HTTP 応答メッセージの初期状況表示行と同じでなければなりません。

```
HTTP/1.1 nnn tttttttt
```

HTTP/1.1

HTTP のバージョンとリリース。

nnn 戻される HTTP 状況コード (3 桁の 10 進数)。

tttttttt

状況コード **nnn** に関連した、人が読んで理解できる状況テキスト。

内容は、例えば次のストリングのようになります。

```
HTTP/1.1 412 Precondition Failed
```

パイプラインが WebSphere MQ トランスポートを使用する場合、DFHHTTPSTATUS コンテナは無視されます。

DFHMEDIATYPE コンテナ

DFHMEDIATYPE は、サービス・プロバイダー・パイプラインの応答段階で生成されるメッセージのメディア・タイプを指定するために使われる、DATATYPE(CHAR) のコンテナです。

DFHMEDIATYPE コンテナの内容は、スラッシュ文字で区切られたタイプおよびサブタイプである必要があります。以下の 2 つのストリングは、DFHMEDIATYPE コンテナの内容として正しい例を示しています。

```
text/plain
```

```
image/svg+xml
```

パイプラインが WebSphere MQ トランスポートを使用する場合、DFHMEDIATYPE コンテナは無視されます。

DFHNORESPONSE コンテナ

DFHNORESPONSE は、サービス・リクエスター・パイプラインの要求段階で、サービス・プロバイダーが応答を戻すことを期待できないことを示す、DATATYPE(CHAR) のコンテナです。

DFHNORESPONSE コンテナの内容は定義されていません。サービス・プロバイダーが応答を戻すことが見込まれるかどうかを認識する必要のあるメッセージ・ハンドラーは、コンテナが存在するかどうかのみを判別する必要があります。

- コンテナ DFHNORESPONSE が存在する場合は、応答は見込まれません。
- コンテナ DFHNORESPONSE が不在の場合は、応答が見込まれます。

この情報は、サービス・プロバイダーと組み合わせて使用するプロトコルに基づいて、最初はサービス・リクエスターのアプリケーションから伝達されます。したがって、メッセージ・ハンドラーに存在するこのコンテナを削除すること (または、存在しない場合に作成すること) はお勧めできません。エンドポイント間のプロトコルを乱す可能性があるためです。

サービス・リクエスター・パイプラインの要求段階以外では、このコンテナの使用は定義されていません。

DFHREQUEST コンテナ

DFHREQUEST は、パイプラインの要求段階で処理される要求メッセージを格納する DATATYPE(CHAR) のコンテナです。

CICS 提供 SOAP メッセージ・ハンドラーの 1 つがパイプラインで構成されている場合は、コンテナ DFHREQUEST が更新されて SOAP エンベロープに SOAP メッセージ・ヘッダーが組み込まれます。メッセージが CICS 提供の SOAP メッセージ・ハンドラーによって作成され、その後変更されていない場合、DFHREQUEST には完全な SOAP エンベロープが格納され、そのすべての内容が UTF-8 コード・ページに入ります。

DFHREQUEST コンテナは、メッセージ・ハンドラーが呼び出されるときに要求に存在し、DFHFUNCTION コンテナには RECEIVE-REQUEST または SEND-REQUEST が格納されます。

この状態の通常のプロトコルでは、DFHREQUEST を同じ内容または変更した内容でパイプラインに戻します。パイプライン要求段階の処理が正常に続行され、パイプライン内の次のメッセージ・ハンドラー・プログラム (存在する場合) の処理が行われます。

これに代わる方法として、メッセージ・ハンドラーでコンテナ DFHREQUEST を削除し、DFHRESPONSE コンテナに応答を書き込むことができます。このようにして、通常の順序の逆に処理が実行され、パイプラインの応答段階の処理が続行されます。

DFHRESPONSE コンテナ

DFHRESPONSE は、パイプラインの応答段階で処理される応答メッセージを格納する DATATYPE(CHAR) のコンテナです。メッセージが CICS 提供の SOAP メッセージ・ハンドラーによって作成され、その後変更されていない場合、DFHRESPONSE には完全な SOAP エンベロープとその UTF-8 コード・ページのすべての内容が格納されます。

DFHRESPONSE コンテナは、メッセージ・ハンドラーが呼び出されるときに存在し、DFHFUNCTION コンテナには SEND-RESPONSE または RECEIVE-RESPONSE が格納されます。

この状態の通常のプロトコルでは、DFHRESPONSE を同じ内容または変更した内容でパイプラインに戻します。パイプラインの処理は正常に続行され、パイプライン内の次のメッセージ・ハンドラー・プログラム (存在する場合) の処理が行われます。

DFHFUNCTION コンテナに RECEIVE-REQUEST、SEND-REQUEST、PROCESS-REQUEST、または HANDLER-ERROR が格納されているとき、DFHRESPONSE コンテナも存在しますが、長さはゼロです。

コンテナがパイプライン・プロトコルを制御する方法

DFHFUNCTION、DFHREQUEST、および DFHRESPONSE コンテナの内容が一緒にパイプライン・プロトコルを制御します。

パイプラインの実行の 2 つの段階 (要求段階と応答段階) で、DFHFUNCTION の値が、各メッセージ・ハンドラーに渡される制御コンテナを決定します。

DFHFUNCTION	コンテキスト	DFHREQUEST	DFHRESPONSE
RECEIVE-REQUEST	サービス・プロバイダー、要求段階	存在 (長さ > 0)	存在 (長さ = 0)
SEND-RESPONSE	サービス・プロバイダー、応答段階	不在	存在 (長さ > 0)
SEND-REQUEST	サービス・リクエスター、要求段階	存在 (長さ > 0)	存在 (長さ = 0)
RECEIVE-RESPONSE	サービス・リクエスター、応答段階	不在	存在 (長さ > 0)
PROCESS-REQUEST	サービス・プロバイダー、端末ハンドラー	存在 (長さ > 0)	存在 (長さ = 0)
HANDLER-ERROR	サービス・リクエスターまたはサービス・プロバイダー、どちらかの段階	不在	存在 (長さ = 0)
NO-RESPONSE	サービス・リクエスターまたはサービス・プロバイダー、応答段階	不在	不在

その後の処理は、メッセージ・ハンドラーがパイプラインに戻すコンテナによって決定されます。

要求段階中

- メッセージ・ハンドラーは DFHREQUEST コンテナに戻すことができます。処理は次のハンドラーを使用して要求段階で続行します。コンテナ内のデータの長さをゼロにしてはなりません。
- メッセージ・ハンドラーは DFHRESPONSE コンテナに戻すことができます。処理は応答段階に切り替わり、DFHFUNCTION をサービス・プロバイダーで SEND-RESPONSE に、サービス・リクエスターで RECEIVE-RESPONSE に設定して、同じハンドラーが呼び出されます。コンテナ内のデータの長さをゼロにしてはなりません。
- メッセージ・ハンドラーはコンテナに戻すことができません。処理は応答段階に切り替わり、DFHFUNCTION を NO-RESPONSE に設定して、同じハンドラーが呼び出されます。

端末ハンドラーで (サービス・プロバイダーのみ)

- メッセージ・ハンドラーは DFHRESPONSE コンテナに戻すことができます。処理は応答段階に切り替わり、DFHFUNCTION の新しい値

(SEND-RESPONSE) で、前のハンドラーが呼び出されます。コンテナ内のデータの長さをゼロにしてはなりません。

- メッセージ・ハンドラーはコンテナを戻すことができません。処理は応答段階に切り替わり、DFHFUNCTION の新しい値 (NO-RESPONSE) で、前のハンドラーが呼び出されます。

応答段階中

- メッセージ・ハンドラーは DFHRESPONSE コンテナを戻すことができます。処理は応答段階で続行し、次のハンドラーが呼び出されます。コンテナ内のデータの長さをゼロにしてはなりません。
- メッセージ・ハンドラーはコンテナを戻すことができません。処理は応答段階で続行し、DFHFUNCTION の新しい値 (NO-RESPONSE) で、シーケンスの次のハンドラーが呼び出されます。

重要: 要求段階で、メッセージ・ハンドラーは DFHREQUEST または DFHRESPONSE を戻すことができますが、両方を戻すことはできません。メッセージ・ハンドラーが呼び出されるときに両方のコンテナが存在しているので、どちらかを削除する必要があります。

次の表に、DFHFUNCTION のすべての値と、各メッセージ・ハンドラーによって戻される DFHREQUEST と DFHRESPONSE のすべての組み合わせについて、パイプラインによってとられるアクションを示します。

DFHFUNCTION	コンテキスト	DFHREQUEST	DFHRESPONSE	アクション
RECEIVE-REQUEST	サービス・プロバイダー、要求段階	存在 (長さ > 0)	存在	(エラー)
			不在	RECEIVE-REQUEST 関数で次のハンドラーを呼び出す
		存在 (長さ = 0)	適用外	(エラー)
		不在	存在 (長さ > 0)	応答段階に切り替え、SEND-RESPONSE 関数で同じハンドラーを呼び出す
			存在 (長さ = 0)	(エラー)
			不在	NO-RESPONSE 関数で同じハンドラーを呼び出す
SEND-RESPONSE	サービス・プロバイダー、応答段階	適用外	存在 (長さ > 0)	SEND-RESPONSE 関数で前のハンドラーを呼び出す
			存在 (長さ = 0)	(エラー)
			不在	NO-RESPONSE 関数で同じハンドラーを呼び出す
SEND-REQUEST	サービス・リクエスター、要求段階	存在 (長さ > 0)	存在 (長さ ≥ 0)	(エラー)
			不在	SEND-REQUEST 関数で次のハンドラーを呼び出す
		存在 (長さ = 0)	適用外	(エラー)
		不在	存在 (長さ > 0)	応答段階に切り替え、RECEIVE-RESPONSE 関数で前のハンドラーを呼び出す
			存在 (長さ = 0)	(エラー)
			不在	NO-RESPONSE 関数で同じハンドラーを呼び出す

DFHFUNCTION	コンテキスト	DFHREQUEST	DFHRESPONSE	アクション
RECEIVE-RESPONSE	サービス・リクエスター、応答段階	適用外	存在 (長さ > 0)	RECEIVE-RESPONSE 関数で前のハンドラーを呼び出す
			存在 (長さ = 0)	(エラー)
			不在	NO-RESPONSE 関数で同じハンドラーを呼び出す
PROCESS-REQUEST	サービス・プロバイダー、端末ハンドラー	適用外	存在 (長さ > 0)	RECEIVE-RESPONSE 関数で前のハンドラーを呼び出す
			存在 (長さ = 0)	(エラー)
			不在	NO-RESPONSE 関数で同じハンドラーを呼び出す
HANDLER-ERROR	サービス・リクエスターまたはサービス・プロバイダー、どちらかの段階	適用外	存在 (長さ > 0)	SEND-RESPONSE 関数または RECEIVE-RESPONSE 関数で前のハンドラーを呼び出す
			存在 (長さ = 0)	(エラー)
			不在	NO-RESPONSE 関数で同じハンドラーを呼び出す

コンテキスト・コンテナー

状況によっては、ユーザー作成のメッセージ・ハンドラー・プログラム、およびヘッダー処理プログラムが呼び出されるコンテキストについての情報がこれらのプログラムに必要です。CICS は、プログラムに渡される一連のコンテキスト・コンテナー にこの情報を提供します。

CICS は、各コンテキスト・コンテナーの内容を初期化しますが、場合によっては、メッセージ・ハンドラー・プログラムやヘッダー処理プログラムの内容を変更できます。例えば、端末ハンドラーが CICS 提供の SOAP ハンドラーの 1 つであるサービス・プロバイダー・パイプラインで、ターゲット・アプリケーション・プログラムのユーザー ID やトランザクション ID を変更するには、該当するコンテキスト・コンテナーの内容を変更します。

コンテナーに格納される情報の一部は、サービス・プロバイダーにのみ、またはサービス・リクエスターにのみ適用されるため、すべてのコンテキスト・コンテナーが両方で利用できるわけではありません。

DFH-HANDLERPLIST コンテナー

DFH-HANDLERPLIST は、パイプライン構成ファイルの適切な

<handler_parameter_list> エレメントの内容で初期化される DATATYPE(CHAR) のコンテナーです。

パイプライン構成ファイルのハンドラー・パラメーター・リストを指定していなかった場合、コンテナーは空になります。つまり、コンテナーの長さはゼロです。

このコンテナーの内容を変更することはできません。

DFH-SERVICEPLIST コンテナー

DFH-SERVICEPLIST は、パイプライン構成ファイルの <service_parameter_list> エレメントの内容を格納する DATATYPE(CHAR) のコンテナーです。

パイプライン構成ファイルにサービス・パラメーター・リストを指定していない場合、147 ページの『DFHWS-STURI コンテナ』 138 ページの『DFHWS-URI コンテナ』 コンテナは空になります (つまり、コンテナの長さはゼロです)。

このコンテナの内容を変更することはできません。

DFHWS-APPHANDLER コンテナ

DFHWS-APPHANDLER は、サービス・プロバイダー・パイプラインで、パイプライン構成ファイルの <apphandler> エレメントの内容で初期化される DATATYPE(CHAR) のコンテナです。

パイプラインの端末ハンドラーの場合、CICS 提供の SOAP ハンドラーは、このコンテナからターゲット・アプリケーション・プログラムの名前を取得します。

メッセージ・ハンドラーまたはヘッダー処理プログラムでこのコンテナの内容を変更することができます。

CICS は、サービス・リクエスター・パイプラインではこのコンテナを提供しません。

DFHWS-DATA コンテナ

DFHWS-DATA は、CICS Web サービス・アシスタントで配置されるサービス・リクエスター・アプリケーション (およびオプションでサービス・プロバイダー・アプリケーション) で使用される、DATATYPE(BIT) のコンテナです。このコンテナは、SOAP 要求と相互にマップする最上位のデータ構造を格納します。

サービス・リクエスター・アプリケーションでは、サービス・リクエスター・プログラムが EXEC CICS INVOKE SERVICE コマンドを出すときに、DFHWS-DATA コンテナが存在している必要があります。このコマンドが出されると、CICS は、コンテナにあるデータ構造を SOAP 要求に変換します。SOAP 応答が受信されると、CICS はこれを、同じコンテナにあるアプリケーションに戻される別のデータ構造に変換します。

サービス・プロバイダー・アプリケーションでは、DFHLS2WS または DFHWS2LS バッチ・ジョブで **CONTID** パラメーターを指定しないと、DFHWS-DATA コンテナがデフォルトで使用されます。CICS は、SOAP 要求メッセージを、DFHWS-DATA コンテナ内にあるアプリケーションに渡されるデータ構造に変換します。次に応答が同じコンテナに保管され、CICS がデータ構造を SOAP 応答メッセージに変換します。

DFHWS-MEP コンテナ

DFHWS-MEP は、インバウンドまたはアウトバウンドの SOAP メッセージのメッセージ交換パターン (MEP) についての代表値を格納する、DATATYPE(BIT) のコンテナです。この値の長さは 1 バイトです。

CICS は、サービス・リクエスターとサービス・プロバイダーの両方について、4 つのメッセージ交換パターンをサポートします。メッセージ交換パターンは Web サービスについて WSDL 2.0 文書で定義され、CICS がプロバイダーとして応答すべきかどうか、および CICS が外部プロバイダーからの応答を期待すべきかど

うかを決定します。リクエスター・モードでは、CICS が応答を待機する時間は PIPELINE リソースを使用して構成されます。

CICS Web サービス・アシスタントを使ってアプリケーションを配置した場合、CICS によってこのコンテナのデータが設定されます。

- ・ サービス・プロバイダー・パイプラインでは、このコンテナは、端末ハンドラーからインバウンド・メッセージを受け取る時に、DFHPITP アプリケーション・ハンドラーによってデータを設定されます。
- ・ サービス・リクエスター・パイプラインでは、このコンテナは、アプリケーションが INVOKE SERVICE コマンドを使用するときにデータを設定されます。

アプリケーションが DFHPIRT チャンネルを使用してパイプラインを開始する場合、アプリケーションがこのコンテナのデータを設定します。コンテナが存在しないかコンテナに値がない場合、チャンネルに DFHNORESPONSE コンテナが存在するかどうかに応じて、CICS は要求が In-Out または In-Only MEP のいずれかを使用していると想定します。

表 7. コンテナ DFHWS-MEP に表示される値

値	MEP	URI
1	In-Only	http://www.w3.org/ns/wsdl/in-only
2	In-Out	http://www.w3.org/ns/wsdl/in-out
4	Robust-In-Only	http://www.w3.org/ns/wsdl/robust-in-only
8	In-Optional-Out	http://www.w3.org/ns/wsdl/in-opt-out

DFHWS-OPERATION コンテナ

DFHWS-OPERATION は、CICS Web サービス・アシスタントで配置されるサービス・プロバイダー・アプリケーションで通常使用される、DATATYPE(CHAR) のコンテナです。SOAP 要求で指定される操作の名前を格納します。

サービス・プロバイダーでは、アプリケーションが呼び出される操作の名前をこのコンテナが指定します。CICS 提供の SOAP メッセージ・ハンドラーがターゲット・アプリケーション・プログラムに制御を渡すときに、このコンテナにデータが設定され、ターゲット・プログラムがチャンネル・インターフェースを使用して呼び出されるときのみ、このコンテナが表示されます。

サービス・リクエスター・パイプラインでは、このコンテナは、EXEC CICS INVOKE SERVICE コマンドの OPERATION オプションに指定される名前を格納します。このコンテナは、このコマンドを出すアプリケーションでは使用できません。

DFHWS-PIPELINE コンテナ

DFHWS-PIPELINE は、プログラムが実行されている PIPELINE の名前を格納する、DATATYPE(CHAR) のコンテナです。

このコンテナの内容を変更することはできません。

DFHWS-RESPWAIT コンテナ

DFHWS-RESPWAIT は DATATYPE(BIT) のコンテナであり、アウトバウンド Web サービス要求メッセージに適用されるタイムアウト (秒) を表す符号なしフルワード 2 進数がこれに入ります。

このコンテナの初期値は PIPELINE リソースの RESPWAIT 属性に基づきますが、パイプライン処理中にこの値を適切に変更することができます。

このコンテナはリクエスター・モードのパイプラインでのみ使用されます。

DFHWS-SOAPLEVEL コンテナ

DFHWS-SOAPLEVEL は、処理するメッセージで使用される SOAP のレベルについての情報を格納する、DATATYPE(BIT) のコンテナです。

このコンテナには、Web サービス要求または応答で使用される SOAP のレベルを示す、バイナリー・フルワードが格納されます。

- 1 要求または応答は SOAP 1.1 メッセージです。
- 2 要求または応答は SOAP 1.2 メッセージです。
- 10 要求または応答は SOAP メッセージではありません。

このコンテナの内容を変更することはできません。

DFHWS-TRANID コンテナ

DFHWS-TRANID は、パイプラインが稼働中のタスクのトランザクション ID を使用して初期化される、DATATYPE(CHAR) のコンテナです。

端末ハンドラーが CICS 提供の SOAP ハンドラーの 1 つであるサービス・プロバイダー・パイプラインでこのコンテナの内容を変更した場合 (かつ変更のタイミングが、ターゲット・アプリケーション・プログラムに制御が渡される前である場合)、ターゲット・アプリケーションは、新規のトランザクション ID を使用して新規のタスク内で実行されます。

DFHWS-URI コンテナ

DFHWS-URI は、サービスの URI を格納する、DATATYPE(CHAR) のコンテナです。

サービス・プロバイダー・パイプラインでは、CICS が着信メッセージから相対 URI を抽出し、これを DFHWS-URI コンテナに入れます。

例えば、Web サービスの URI が `http://example.com/location/address` または `jms://queue?destination=INPUT.QUEUE&targetService=/location/address` の場合、相対 URI は `/location/address` です。

リクエスター・パイプラインで Web サービス・アドレッシングを使用する場合、このコンテナは以下の順序で作成および更新されます。

1. INVOKE SERVICE コマンドが実行されると、DFHWS-URI コンテナが作成され、それが WSDL サービス・エンドポイント・アドレスの値で開始されます。アドレッシング・コンテキストが WSACONTEXT BUILD API コマンドを使用

して作成された場合は、INVOKE SERVICE コマンドに **URI** パラメーターまたは **URIMAP** パラメーターを指定しないでください。

2. Web サービス・アドレッシング・ハンドラー (DFHWSADH) の実行時に、アドレッシング・コンテキスト内に、匿名ではない URI が設定された <wsa:To> EPR が存在する場合は、その <wsa:To> EPR の値によって DFHWS-URI コンテナ内の URI が上書きされます。

SOAP メッセージが、DFHWS-URI の URI によって定義されたサービスに送られます。

サービス・リクエスター・パイプラインでは、CICS は DFHWS-URI コンテナ内に、INVOKE SERVICE コマンドで指定されている URI を入れます。これが存在しない場合には Web サービス・バインディングの URI を入れます。パイプラインでメッセージ・ハンドラーを使用することにより、この URI を指定変更することができます。

サービスは外部サービスとして HTTP、HTTPS、JMS、または WMQ URI を使用できます。また、サービスでは、次に示すように別の CICS アプリケーションによって提供されるサービス用の CICS URI を使用することもできます。

URI	照会ストリング	説明
cics://PROGRAM/program	?options	指定されたプログラムにリンクするために CICS トランスポート・ハンドラーは EXEC CICS LINK PROGRAM コマンドを使用し、現在のチャンネルとコンテナを渡します。アプリケーション・データに対するデータ変換は行われません。
cics://SERVICE/service	?targetServiceUri=targetServiceUri &options	プロバイダー・パイプラインを介して要求を実行するために、CICS トランスポート・ハンドラーは (targetServiceUri と表される) サービスのパスを使って URIMAP リソースをマッチングします。 この URI タイプを使用する場合には、targetServiceUri パラメーターの値を指定する必要があります。
cics://PIPELINE/pipeline	?targetServiceUri=targetServiceUri	CICS トランスポート・ハンドラーは別のサービス・リクエスター・パイプラインを開始します。

parameter=value という形式 (各パラメーターをアンパーサンドで区切る) を使用して、それぞれのタイプの CICS URI にパラメーターを追加することができます。CICS URI には次のような規則が適用されます。

- 照会ストリング内の最初のパラメーターには、接頭部として疑問符 (?) が必要です。URI の中で、この場所より前に疑問符 (?) を使うことはできません。
- パラメーター値の中にアンパーサンドを含めるには、文字をエスケープする必要があります。詳しくは、下記の例のセクションを参照してください。

- *program* および *pipeline* に小文字の値が含まれている場合、CICS はそれらを大文字に変更します。

リクエスト・パイプラインの終わりで CICS がどのように要求を処理するかは、照会ストリングのパラメーターによって次のように決定されます。

maxCommareaLength=value

ターゲット・アプリケーション・プログラムに必要な COMMAREA の最大サイズ (バイト) を指定します。この値は 32 763 を超えることができません。このパラメーターが照会ストリングに存在する場合、CICS は COMMAREA を使用して指定されたプログラムにリンクします。このパラメーターが照会ストリングに存在しない場合、CICS はチャンネルを使用して指定されたプログラムにリンクします。

このパラメーターでは大/小文字の区別がなく、cics://PROGRAM URI でのみ有効です。

newTask=yes|no

トランスポート・ハンドラーが新しいタスクとして要求を実行するかどうかを指定します。

このパラメーターは、大/小文字が区別されません。cics://PROGRAM/testapp?newTask=yes と cics://PROGRAM/testapp?NEWTASK=Yes は同じです。

targetServiceUri=uri

呼び出されるサービスのパスを指定します。SERVICE 宛先タイプでは、トランスポート・ハンドラーはサービス・プロバイダー・パイプラインを開始するために host=localhost という値を使って URIMAP リソースを見つけます。PIPELINE 宛先タイプでは、トランスポート・ハンドラーは別のリクエスト・パイプラインを開始するためにこの値を使用します。

このパラメーターは、大/小文字が区別されます。

transid=char(4)

どのトランザクションで要求が実行されるかを指定します。トランスポート・ハンドラーは、指定されたトランザクション ID を使って要求ストリームを開始します。

このパラメーターは、大/小文字が区別されます。

userid=char(8)

どのユーザー ID で要求が実行されるかを指定します。トランスポート・ハンドラーは、指定されたユーザー ID を使って要求ストリームを開始します。

このパラメーターは、大/小文字が区別されません。

宛先タイプ	URI におけるパラメーター	
PROGRAM	userid	オプション
	transid	オプション
	maxCommareaLength	オプション
	newTask	オプション。userid または transid を指定する場合、これを yes にするか、または指定しないでください。
	targetServiceUri	サポートされていない

宛先タイプ	URI におけるパラメーター	
SERVICE	userid	オプション
	transid	オプション
	maxCommareaLength	サポートされていない
	newTask	オプション。 userid または transid を指定する場合、これを yes にするか、または指定しないでください。
	targetServiceUri	必須
PIPELINE	userid	サポートされていない
	transid	サポートされていない
	maxCommareaLength	サポートされていない
	newTask	サポートされていない
	targetServiceUri	必須

CICS URI の例

この最初の例では、パイプラインの終わりに達するまでに、以下のような URI が DFHWS-URI コンテナに入ります。

```
cics://PROGRAM/testapp?newTask=yes&userid=user1
```

トランスポート・ハンドラーは testapp という CICS プログラムにリンクして、チャネルとコンテナを渡します。データ変換は行われなため、ターゲット・プログラムは現在のチャネル上のコンテナの内容を処理する必要があります。CICS は新しい作業単位、別のユーザー ID user1 のもとでプログラムにリンクします。

この 2 番目の例では、パイプラインの終わりに達するまでに、以下のような URI が DFHWS-URI コンテナに入ります。

```
cics://SERVICE/getStockQuote?targetServiceUri=/stock/getQuote&newTask=yes&userid=user2
```

トランスポート・ハンドラーは値 /stock/getQuote を使って DFHWS-URI コンテナ内の URI を置換し、URI を解決するために **targetServiceUri** パラメーター内のパスを使って URIMAP を検出し、新しいタスクおよび異なるユーザー ID のもとでプロバイダー・パイプラインを開始します。

この 3 番目の例では、パイプラインの終わりに達するまでに、以下のような URI が DFHWS-URI コンテナに入ります。

```
cics://PIPELINE/reqpipeA?targetServiceUri=cics://PROGRAM/testapp?newTask=yes%26userid=user1
```

トランスポート・ハンドラーは値 cics://PROGRAM/testapp?newTask=yes&userid=user1 を使って DFHWS-URI コンテナ内の URI を置換し、reqpipeA というリクエスター・パイプラインを開始して現在のチャネルとコンテナを渡します。文字 %26 はアンパーサンドをエスケープするため、トランスポート・ハンドラーは URI 全体を DFHWS-URI コンテナに入れます。

関連概念

258 ページの『リクエスター・パイプライン処理を制御するためのオプション』サービス・リクエスター・パイプラインにおいて、メッセージ・ハンドラーは URI を変更することによって Web サービス要求が送られる場所を決定できま

す。CICS はさまざまな URI 形式をサポートするため、パイプラインで Web サービス要求が処理される方法をかなり柔軟に制御することができます。

関連タスク

261 ページの『URI を使用したリクエスター・パイプライン処理の制御』
サービス・リクエスター・パイプラインにおいて、メッセージ・ハンドラーは URI を変更することにより Web サービス要求をどこに送るかを決定できます。URI 形式を変更すると、別のリクエスター・パイプラインを開始したり、ネットワークを介して要求を送信せずにサービス・プロバイダー・パイプラインを開始するなどの最適化を実行できます。

DFHWS-USERID コンテナ

DFHWS-USERID は、パイプラインが稼働中のタスクのユーザー ID を使用して初期化される、DATATYPE(CHAR) のコンテナです。

端末ハンドラーが CICS 提供の SOAP ハンドラーの 1 つであるサービス・プロバイダー・パイプラインでこのコンテナの内容を変更した場合 (かつ変更のタイミングが、ターゲット・アプリケーション・プログラムに制御が渡される前である場合)、ターゲット・アプリケーションは、新規のユーザー ID と関連した新規のタスクで実行されます。コンテナ DFHWS-TRANID の内容を変更しない限り、新しいタスクのトランザクション ID は、パイプラインが実行されているタスクと同じになります。

DFHWS-WEBSERVICE コンテナ

DFHWS-WEBSERVICE は、サービス・プロバイダー・パイプラインでのみ使用される、DATATYPE(CHAR) のコンテナです。これは、ターゲット・アプリケーションが Web サービス・アシスタントを使用して配置されているときに、実行環境を指定する Web サービスの名前を格納します。

CICS は、サービス・リクエスター・パイプラインではこのコンテナを提供しません。

DFHWS-CID-DOMAIN コンテナ

DFHWS-CID-DOMAIN は DATATYPE(CHAR) のコンテナです。バイナリー添付ファイルを参照するための content-ID 値の生成に使用されるドメイン・ネームを格納します。

ドメイン名の値はデフォルトでは `cicsts` です。パイプライン構成ファイルに `<mime_options>` エレメントを指定して、この値を指定変更できます。

このコンテナの内容を変更することはできません。

DFHWS-MTOM-IN コンテナ

DFHWS-MTOM-IN は、パイプライン構成ファイルの `<cics_mtom_handler>` エレメントに指定されたオプションについての情報と、パイプラインに受信されたメッセージ・フォーマットについての情報を格納する、DATATYPE(BIT) のコンテナです。

このコンテナは、パイプラインのインバウンド MTOM メッセージを処理するための情報を格納します。インバウンド・メッセージは、Web サービス・リクエスターからの要求メッセージか、Web サービス・プロバイダーからの応答メッセージである可能性があります。

パイプライン構成ファイルに <cics_mtom_handler> エレメントを指定しない場合や、MTOM メッセージの代わりに SOAP メッセージが受信される場合は、このコンテナは作成されません。

Web サービス・セキュリティがパイプラインで構成されるか、Web サービスについて検証のスイッチが入ると、このコンテナが作成されるときに、DFHWS-MTOM-IN の XOP_MODE フィールドの内容が CICS によって指定変更されることがあります。例えば、MTOM メッセージの内容を直接モードで処理するためにパイプラインを構成してから、Web サービスについて検証のスイッチを入れると、CICS は、パイプライン構成ファイル内に定義された値を指定変更して、互換モードで実行するように XOP 処理を設定します。CICS が指定変更する理由は、パイプライン内での XOP 文書とバイナリー添付ファイルの処理サポートにおける制限のためです。

このコンテナの内容を変更することはできません。

表 8. DFHWS-MTOM-IN コンテナの構造

フィールド名	長さ (バイト)	内容
MTOM_STATUS	4	CICS が受信したメッセージが MTOM フォーマットであることを示す値「1」が含まれます。
MTOMNOXOP_STATUS	4	以下のいずれかの値が含まれます。 0 MTOM メッセージにバイナリー添付ファイルが含まれます。 1 MTOM メッセージにバイナリー添付ファイルが含まれません。
XOP_MODE	4	以下のいずれかの値が含まれます。 0 XOP 処理は行われません。 1 XOP 処理は互換モードで行われます。 2 XOP 処理は直接モードで行われます。

DFHWS-MTOM-OUT コンテナ

DFHWS-MTOM-OUT は、パイプライン構成ファイルの <cics_mtom_handler> エレメントに指定されたオプションについての情報を格納する、DATATYPE(BIT) のコンテナです。

このコンテナは、パイプライン内のアウトバウンド MTOM メッセージが Web サービス・リクエスターについての応答メッセージであるか Web サービス・プロバイダーについての要求メッセージであるかにかかわらず、これを処理するための情報を格納します。

パイプライン構成ファイルに <cics_mtom_handler> エlementを指定しない場合や、パイプライン構成ファイルの <mtom_options> Elementに属性 send_mtom="no" がある場合は、このコンテナは作成されません。

プロバイダー・モードでは、このコンテナは DFHWS-MTOM-IN コンテナと同時に作成されます。パイプライン構成ファイルの <mtom_options> Elementに属性 send_mtom="same" がある場合は、Web サービス・リクエスターにとって MTOM と SOAP どちらの応答メッセージが望ましいのかを示す MTOM_STATUS フィールドが設定されます。

Web サービス・セキュリティーがパイプラインで構成されるか、Web サービスについて検証のスイッチが入ると、このコンテナが作成されるときに、DFHWS-MTOM-OUT の XOP_MODE フィールドが CICS によって変更されることがあります。例えば、XOP 文書と任意のバイナリー添付ファイルを直接モードで処理するためにパイプラインを構成してから、Web サービスについて検証のスイッチを入れると、CICS は、パイプライン構成ファイル内に定義された値を指定変更して、コンテナを作成するときに互換モードで実行するように XOP 処理を設定します。CICS が指定変更する理由は、パイプライン内での XOP 文書とバイナリー添付ファイルの処理サポートにおける制限のためです。

このコンテナの内容を変更することはできません。

表 9. DFHWS-MTOM-OUT コンテナの構造

フィールド名	長さ (バイト)	内容
MTOM_STATUS	4	MTOM が使用可能かどうかを示します。 0 MTOM は使用可能ではありません。アウトバウンド・メッセージは SOAP フォーマットで送信されます。 1 MTOM は使用可能です。アウトバウンド・メッセージは MTOM フォーマットで送信されます。
MTOMNOXOP_STATUS	4	バイナリー添付ファイルがない場合に MTOM を使用するかどうかを示します。 0 バイナリー添付ファイルがない場合に MTOM メッセージを送信しません。 1 バイナリー添付ファイルがない場合に MTOM メッセージを送信します。
XOP_MODE	4	行われる XOP 処理を示します。 0 XOP 処理は行われません。 1 XOP 処理は互換モードで行われます。 2 XOP 処理は直接モードで行われます。

DFHWS-WSDL-CTX コンテナ

DFHWS-WSDL-CTX は、CICS Web サービス・アシスタントと共に配置されるサービス・プロバイダーまたはサービス・リクエスター・アプリケーションで使用される、DATATYPE(CHAR) のコンテナです。これには、モニターに使用できる WSDL コンテキスト情報が保持されます。

DFHWS-WSDL-CTX は、WSDL 文書に関する次のようなコンテキスト情報を保持します。

- アプリケーションが呼び出される操作の名前と名前空間。
- 既知の場合、使用される WSDL 1.1 ポートまたは WSDL 2.0 エンドポイントの名前と名前空間。

これらの値はスペース文字で区切られます。DFHWS-WSDL-CTX は、ランタイム・レベル 2.1 以上の CICS によってのみデータが設定されます。

CICS Web サービス・アシスタントを使ってアプリケーションを配置した場合、CICS によってこのコンテナのデータが設定されます。

- サービス・プロバイダー・パイプラインでは、このコンテナは、端末ハンドラーからインバウンド・メッセージを受け取る際に、DFHPITP アプリケーション・ハンドラーによってデータを設定されます。
- サービス・リクエスター・パイプラインでは、このコンテナは、アプリケーションが INVOKE SERVICE コマンドを使用するときにデータを設定されます。

アプリケーションが DFHPIRT プログラムを使ってパイプラインを開始する場合、アプリケーションは必要に応じて DFHWS-WSDL-CTX コンテナのデータを設定します。

DFHWS-XOP-IN コンテナ

DFHWS-XOP-IN は DATATYPE(BIT) のコンテナです。インバウンド MIME メッセージからアンパックされ、XOP 処理を使用してコンテナに配置された、バイナリー添付ファイルに対する参照のリストを格納します。

DFHWS-XOP-IN コンテナの各添付レコードは、次のような項目から成ります。

- バイナリー添付ファイルに関する MIME ヘッダーを格納するコンテナの 16 バイトの名前
- バイナリー添付ファイルを格納するコンテナの 16 バイトの名前
- 符号付きハーフワード・バイナリー・フォーマットの、2 バイト長の content-ID
- ASCII 文字ストリングとして保管された、< および > 区切り文字を含む content-ID

このコンテナの内容を変更することはできません。

DFHWS-XOP-OUT コンテナ

DFHWS-XOP-OUT は DATATYPE(BIT) のコンテナです。バイナリー添付ファイルを格納するコンテナに対する参照のリストが含まれます。バイナリー添付ファイルは、MTOM ハンドラー・プログラムによって、アウトバウンド MIME メッセージにパックされます。

DFHWS-XOP-OUT コンテナの各添付レコードは、次のような項目から成ります。

- バイナリー添付ファイルに関する MIME ヘッダーを格納するコンテナの 16 バイトの名前
- バイナリー添付ファイルを格納するコンテナの 16 バイトの名前
- 符号付きハーフワード・バイナリー・フォーマットの、2 バイト長の content-ID

- ASCII 文字ストリングとして保管された、< および > 区切り文字を含む content-ID

このコンテナの内容を変更することはできません。

セキュリティー・コンテナ

セキュリティー・コンテナは、Tivoli Federated Identity Manager などの Security Token Service (STS) との間で ID トークンを送受信するために、DFHWSTC-V1 チャンネル上で使用されます。このインターフェースは *Trust* クライアント・インターフェースと呼ばれ、Web サービス・リクエスターおよびプロバイダーのパイプラインで使用できます。

DFHWS-IDTOKEN コンテナ

DFHWS-IDTOKEN は DATATYPE(CHAR) のコンテナです。これには、メッセージの ID トークンを発行するために Security Token Service (STS) によって検証または使用されるトークンが入ります。

トークンは XML 形式でなくてはなりません。

このコンテナは、Trust クライアント・インターフェースのチャンネル DFHWSTC-V1 でのみ使用してください。

DFHWS-RESTOKEN コンテナ

DFHWS-RESTOKEN は DATATYPE(CHAR) のコンテナです。Security Token Service (STS) からの応答を格納します。

応答は、DFHWS-STSACTION コンテナで STS から要求されたアクションに応じて異なります。

- アクションが発行である場合、このコンテナは、DFHWS-IDTOKEN コンテナに送信されたものに対して STS が交換したトークンを格納します。
- アクションが検証である場合、このコンテナは、DFHWS-IDTOKEN コンテナに送信されたセキュリティー・トークンが有効か無効かを示す URI を保持します。戻される可能性のある URI は次のとおりです。

URI	説明
http://schemas.xmlsoap.org/ws/2005/02/trust/status/valid	セキュリティー・トークンが有効です。
http://schemas.xmlsoap.org/ws/2005/02/trust/status/invalid	セキュリティー・トークンが無効です。

このコンテナは、Trust クライアント・インターフェースを使用する際に、チャンネル DFHWSTC-V1 に戻されます。

DFHWS-SERVICEURI コンテナ

DFHWS-SERVICEURI は DATATYPE(CHAR) のコンテナです。Security Token Service (STS) が AppliesTo の有効範囲として使用する URI を格納します。

AppliesTo スコープは、セキュリティー・トークンに関連付ける Web サービスを決定するために使用されます。

このコンテナーは、Trust クライアント・インターフェースのチャネル DFHWSTC-V1 でのみ使用してください。

DFHWS-STSACTION コンテナー

DFHWS-STSACTION は DATATYPE(CHAR) のコンテナーです。セキュリティー・トークンを検証または発行するために、Security Token Service (STS) がとる必要があるアクションの URI を格納します。

このコンテナーで指定できる URI 値は次のとおりです。

URI	説明
http://schemas.xmlsoap.org/ws/2005/02/trust/Issue	STS は、DFHWS-IDTOKEN コンテナーに送信されるものと交換にトークンを発行します。
http://schemas.xmlsoap.org/ws/2005/02/trust/Validate	STS は、DFHWS-IDTOKEN コンテナーに送信されるトークンを検証します。

このコンテナーは、Trust クライアント・インターフェースのチャネル DFHWSTC-V1 でのみ使用してください。

DFHWS-STSFault コンテナー

DFHWS-STSFault は DATATYPE(CHAR) のコンテナーです。Security Token Service (STS) によって戻されたエラーを格納します。

エラーが発生すると、STS が SOAP 障害を出します。SOAP 障害の内容がこのコンテナーに戻されます。

このコンテナーは、Trust クライアント・インターフェースを使用する際に、チャネル DFHWSTC-V1 に戻されます。

DFHWS-STReason コンテナー

DFHWS-STReason は DATATYPE(CHAR) のコンテナーです。このエレメントが Security Token Service (STS) からの応答メッセージに存在する場合は、<wst:Reason> エレメントの内容を格納します。

<wst:Reason> エレメントには、CICS によって STS に送信された検証要求の状況に関連する情報を提供する、オプションのストリングが含まれます。セキュリティー・トークンが無効な場合、STS によって提供されたこのエレメント内の情報を調べると、トークンが無効である理由を判別できる可能性があります。

詳しくは、<http://www.ibm.com/developerworks/library/specification/ws-trust/> に公開されている「Web Services Trust Language」仕様を参照してください。

DFHWS-STURI コンテナー

DFHWS-STURI は DATATYPE(CHAR) のコンテナーです。SOAP メッセージについて ID トークンを検証または発行するために使用する、Security Token Service (STS) の絶対 URI を格納します。

URI の形式は `http://www.example.com:8080/TrustServer/SecurityTokenService` です。セキュリティ要件に応じて、HTTP または HTTPS を使用できます。

このコンテナは、Trust クライアント・インターフェースのチャネル DFHWSTC-V1 でのみ使用してください。

DFHWS-TOKENTYPE コンテナ

DFHWS-TOKENTYPE は DATATYPE(CHAR) のコンテナです。Security Token Service (STS) が SOAP メッセージについて ID トークンとして発行する、要求されたトークン・タイプの URI を格納します。

STS でサポートされている限り、有効などのトークン・タイプでも指定できます。

このコンテナは、Trust クライアント・インターフェースのチャネル DFHWSTC-V1 でのみ使用してください。

CICS によって生成されるコンテナ

CICS は、変数配列や長ストリングなどのデータを保管するためのコンテナを生成します。これらのコンテナはパイプライン処理中に作成され、アプリケーション・プログラムとの間の入出力として使用されます。これらのコンテナの接頭部は DFH です。

これらのコンテナの命名規則は、コンテナ名を要求内で固有にするためにそれらを数値接尾部と組み合わせて作成した CICS モジュールを使用することです。パイプライン処理中に作成されるコンテナ名は次のとおりです。

DFHPICC-*nnnnnnnn*

ストリングと変数配列を保管するために使用されるコンテナ。アプリケーションに渡されます。このコンテナはバイナリー・データを含めることもできます。

DFHPH-*nnnnnnnn*

パイプラインが MTOM メッセージ・ハンドラーで使用可能であり、直接モードで実行中の場合に作成される、アウトバウンド接続コンテナ。これらのコンテナは、アプリケーション・プログラムによってバイナリー・データがコンテナではなくフィールドに提供されている場合に作成されます。

DFHPIMM-*nnnnnnnn*

MIME メッセージの処理中に作成されるインバウンド接続コンテナ。これらのコンテナは、MTOM メッセージ・ハンドラーがパイプラインで使用可能な場合に、CICS によって生成されます。直接モード処理が有効な場合、これらのコンテナはアプリケーションに直接移動する場合があります。

DFHPIXO-*nnnnnnnn*

パイプラインが MTOM メッセージ・ハンドラーで使用可能であり、互換モードで実行中の場合に作成される、アウトバウンド接続コンテナ。

番号が付いたコンテナ名は、Web サービス要求ごとに 1 から始まります (例えば、DFHPICC-00000001)。ただし、アプリケーション・プログラムが INVOKE SERVICE コマンドを使って同じチャネルで複数の Web サービス要求を開始する場合、1 つの応答のためにアプリケーションに戻されたコンテナが、それ以降の要

求でも引き続き存在する可能性があります。このような状況では、CICS は、コンテナがすでに存在しているかどうかを確認し、生成されるコンテナの数を増やして命名の競合を回避します。

ユーザー・コンテナ

このコンテナは、あるメッセージ・ハンドラーが別のメッセージ・ハンドラーに渡す必要がある情報が格納されます。ユーザー・コンテナの使用は、全面的にメッセージ・ハンドラーに委ねられます。これらのコンテナには独自の名前を選択することができますが、DFH で始まる名前は使用できません。

第 8 章 Web サービスの作成

既存の CICS アプリケーションを Web サービスとして公開し、新規の CICS アプリケーションを作成して Web サービス・プロバイダーまたはリクエスターとして機能させることができます。

始める前に

Web サービスの作成を始める前に、以下の作業を行ってください。

1. Web サービスをサポートするよう CICS システムを構成します (55 ページの『第 6 章 Web サービスに応じた CICS システムの構成』を参照)。
2. Web サービスの配置をサポートするために必要なインフラストラクチャーを作成します (67 ページの『第 7 章 Web サービス・インフラストラクチャーの作成』を参照)。
3. Web サービス・アシスタントを使用するかどうか決定します (47 ページの『Web サービス使用の計画立案』を参照)。

このタスクについて

付属のユーティリティである CICS Web サービス・アシスタントは、新しい Web サービス・プロバイダーまたはサービス・リクエスター・アプリケーションに必要な成果物を作成したり、既存のアプリケーションを Web サービス・プロバイダーとして使用可能にする作業を支援します。

CICS Web サービス・アシスタントでは、単純な言語構造から WSDL 文書を作成したり、既存の WSDL 文書から言語構造を作成したりすることができ、COBOL、C/C++、および PL/I をサポートします。また、SOAP メッセージからコンテナおよび COMMAREA への自動ランタイム変換、さらに、この逆の変換を可能にするために使用される情報も生成します。この情報は、パイプラインの処理中に CICS Web サービス・サポートで使用されます。

下記のようにして Web サービスを作成した後、正しく機能することを検証します。

1. 以下の 3 通りのいずれかの方法で Web サービスを作成します。
 - Web サービス・アシスタントを使用して、Web サービス記述または言語構造を作成して、CICS に配置します。PIPELINE SCAN を実行して、必要な CICS リソースを自動的に作成できます。
 - Rational Developer for System z または Java™ API を使用して、Web サービス記述または言語構造を作成して、CICS に配置します。この方法を使用すれば、PIPELINE SCAN コマンドを使用して、必要な CICS リソースを自動的に作成することもできます。
 - データ変換を含む、インバウンド・メッセージとアウトバウンド・メッセージの XML を処理し、正しいコンテナをパイプラインに取り込むためのアプリケーション・プログラムを作成または変更します。必要な CICS リソースを手動で作成する必要があります。

2. Web サービスを呼び出して、意図したように機能するかをテストします。 Web サービス・アシスタントを使用して Web サービスを配置する場合は、SET WEBSERVICE コマンドを使用して検証をオンにすることができます。この検証によって、データが正しく変換されていることがチェックされます。

次のタスク

これらのステップの詳細については、次のトピックで説明します。

CICS Web サービス・アシスタント

CICS Web サービス・アシスタントとは、1 組のバッチ・ユーティリティーで、既存の CICS アプリケーションを Web サービスに変換するのに役立ちます。また、これを使用すると、CICS アプリケーションが、外部のプロバイダーによって提供された Web サービスを使用できるようになります。このアシスタントは、サービス・プロバイダーやサービス・リクエスターが使用するための CICS アプリケーションの迅速な配置をサポートしており、プログラミングの労力が最小限で済みます。

CICS の Web サービス・アシスタントを使用する場合は、インバウンド・メッセージを解析し、アウトバウンド・メッセージを作成するための独自のコードを記述する必要はありません。CICS は、SOAP メッセージ本文とアプリケーション・プログラムのデータ構造間でデータをマップするからです。

このアシスタントでは、単純な言語構造から WSDL 文書を作成したり、既存の WSDL 文書から言語構造を作成したりすることができ、COBOL、C/C++、および PL/I をサポートします。また、SOAP メッセージからコンテナおよび COMMAREA への自動ランタイム変換、さらに、この逆の変換を可能にするために使用される情報も生成します。

CICS Web サービス・アシスタントは、以下の 2 つのユーティリティー・プログラムで構成されています。

DFHLS2WS

言語構造を基にして Web サービス・バインディング・ファイルを生成します。このユーティリティーは、Web サービス記述も生成します。

DFHWS2LS

Web サービス記述を基にして Web サービス・バインディング・ファイルを生成します。このユーティリティーは、アプリケーション・プログラムで利用できる言語構造も生成します。

hlq.XDFHINST ライブラリー内に両方のプログラムを実行する JCL プロシージャがあります。

Web サービス・アシスタントのユーティリティー・プログラムとデータ・マッピングの詳細については、以下のトピックを参照してください。

DFHLS2WS: 高水準言語から WSDL への変換

DFHLS2WS プロシージャは高水準言語データ構造から Web サービス記述および Web サービス・バインディング・ファイルを生成します。CICS アプリケーション・プログラムをサービス・プロバイダーとして公開する場合に、DFHLS2WS を使用することができます。

DFHLS2WS のジョブ制御ステートメント、そのシンボリック・パラメーター、入力パラメーターとそれらの説明、およびサンプル・ジョブは、このプロシージャを使用する助けとなります。

DFHLS2WS のジョブ制御ステートメント

JOB ジョブを開始します。

EXEC プロシージャ名 (DFHLS2WS) を指定します。

INPUT.SYSUT1 DD

入力を指定します。入力パラメーターは、通常、入力ストリーム内に指定します。ただし、データ・セットや区分データ・セットのメンバーに定義することもできます。

シンボリック・パラメーター

以下のシンボリック・パラメーターは、DFHLS2WS で定義されます。

JAVADIR=*path*

DFHLS2WS によって使用される Java ディレクトリーの名前を指定します。このパラメーターの値は /usr/lpp/ に追加され、これによって /usr/lpp/*path* という完全なパス名が生成されます。

通常、このパラメーターは指定しません。デフォルト値は、**JAVADIR** パラメーターの CICS インストール・ジョブ (DFHISTAR) に指定された値です。

PATHPREFIX=*prefix*

他のパラメーターで使用される z/OS UNIX ディレクトリー・パスを拡張するオプションの接頭部を指定します。デフォルトは空ストリングです。

通常、このパラメーターは指定しません。デフォルト値は、**JAVADIR** パラメーターの CICS インストール・ジョブ (DFHISTAR) に指定された値です。

SERVICE=*value*

このパラメーターは IBM サポートに指示された場合にのみ使用します。

TMPDIR=*tmpdir*

DFHLS2WS が一時ワークスペースとして使用する z/OS UNIX のディレクトリーの場所を指定します。このジョブを実行するユーザー ID には、このディレクトリーに対する読み取り権限および書き込み権限が必要です。

デフォルト値は /tmp です。

TMPFILE=*tmpprefix*

一時ワークスペース・ファイルの名前を作成するために DFHLS2WS が使用する接頭部を指定します。

デフォルト値は LS2WS です。

USSDIR=path

UNIX システム・サービスのファイル・システム内の CICS TS ディレクトリーの名前を指定します。このパラメーターの値は /usr/lpp/cicsts/ に追加され、これによって /usr/lpp/cicsts/path という完全なパス名が生成されます。

通常、このパラメーターは指定しません。デフォルト値は、**USSDIR** パラメーターの CICS インストール・ジョブ (DFHISTAR) に指定された値です。

一時ワークスペース

DFHLS2WS は、実行時に、次の 3 つの一時ファイルを作成します。

tmpdir/tmpprefix.in
tmpdir/tmpprefix.out
tmpdir/tmpprefix.err

ここで、

tmpdir は、**TMPDIR** パラメーターに指定されている値です。

tmpprefix は、**TMPFILE** パラメーターに指定されている値です。

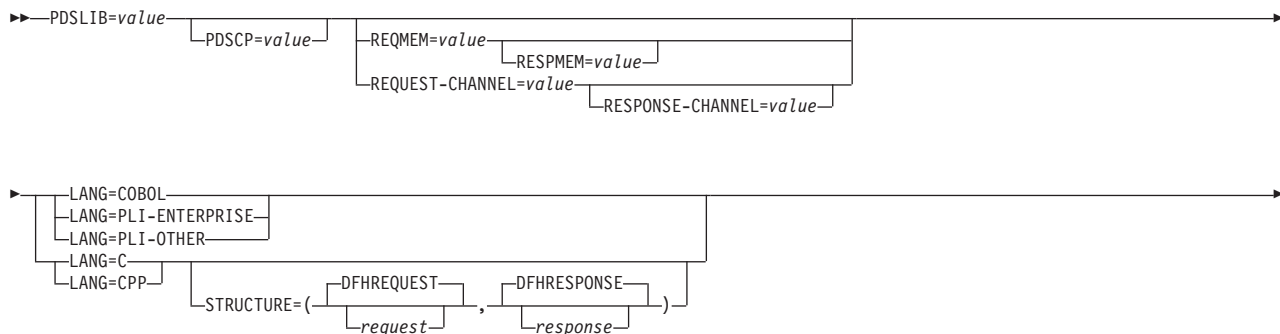
TMPDIR および **TMPFILE** が指定されていない場合、ファイルのデフォルトの名前は、次のとおりです。

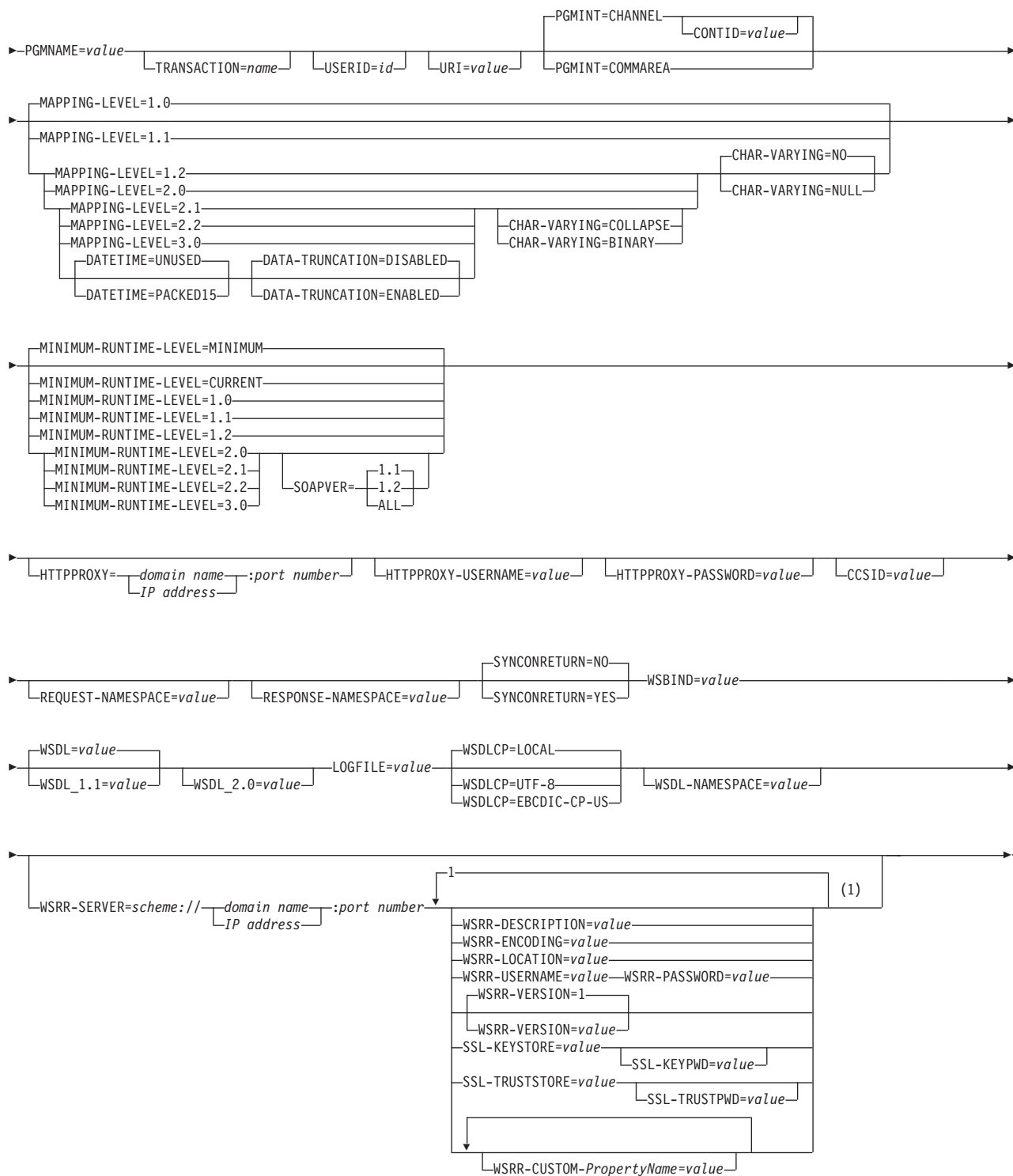
/tmp/LS2WS.in
/tmp/LS2WS.out
/tmp/LS2WS.err

重要: DFHLS2WS は、生成された z/OS UNIX ファイル名へのアクセスをロックしません。したがって、DFHLS2WS の複数のインスタンスが同時に動作して、同じ一時ワークスペース・ファイルを使用する場合には、あるジョブがワークスペース・ファイルを使っているときに別のジョブがそれを上書きするのを防ぐことはできず、予測不能な障害が発生します。

したがって、この状況を回避する命名規則および操作手順を考案することをお勧めします。例えば、システム・シンボリック・パラメーター **SYSUID** を使用すると、個々のユーザーに対して一意のワークスペース・ファイルを生成できます。これらの一時ファイルはジョブの終わりの前に削除されます。

DFHLS2WS のパラメーターの入力





注:

- 1 **WSRR-SERVER** パラメーターの設定時に指定できるそれぞれの **WSRR** パラメーターは、一度だけ指定可能です。ただし、この規則の例外として、**WSRR-CUSTOM** パラメーターは最大 255 回まで指定可能です。

パラメーターの使用法

- 入力パラメーターの指定順序は自由です。
- 各パラメーターは改行後に記述を始める必要があります。
- パラメーターおよび使用する場合は継続文字は、72 列を超えてはなりません。列 73 から 80 はブランクにする必要があります。
- パラメーターが長すぎて 1 行に収まらない場合は、行の末尾にアスタリスク文字 (*) を使用して、そのパラメーターが次の行に続くことを示します。スペースを含むアスタリスクより前の文字はすべてパラメーターの一部とみなされます。例を次に示します。

```
WSBIND=wsbinddir*  
/app1
```

このコードは、次のコードと同じ意味になります。

```
WSBIND=wsbinddir/app1
```

- 行の先頭の文字の位置に # という文字がある場合、この文字はコメント文字を表します。この行は無視されます。

パラメーターの記述

CCSID=value

アプリケーション・データ構造に文字データをエンコードするために、実行時に使用される CCSID を指定します。このパラメーターの値は、**LOCALCCSID** システム初期設定パラメーターの値を指定変更します。*value* は、Java および z/OS 変換サービスでサポートされる EBCDIC CCSID である必要があります。このパラメーターを指定しない場合、アプリケーション・データ構造は、システム初期設定パラメーターで指定された CCSID を使用してエンコードされます。

このパラメーターは任意のマッピング・レベルで使用できます。ただし、生成されたファイルを CICS TS 3.1 領域に配置するには、APAR PK23547 を適用して、Web サービス・バインディング・ファイルをインストールするために、コードの最小実行時レベルを達成する必要があります。

CHAR-VARYING=NO|NULL|COLLAPSE|BINARY

マッピング・レベルが 1.2 以上のとき、言語構造内の文字フィールドがマップされる方法を指定します。COBOL の文字フィールドは X タイプの Picture 節、例えば PIC(X) 10 で、C/C++ の文字フィールドは文字配列です。このパラメーターは、Enterprise およびその他の PL/I 言語構造に適用されません。以下のオプションを選択できます。

NO 文字フィールドは <xsd:string> にマップされ、固定長フィールドとして処理されます。データの最大長はフィールドの長さと同じです。NO は、マッピング・レベル 2.0 以前での、COBOL および PL/I における **CHAR-VARYING** パラメーターのデフォルト値です。

NULL 文字フィールドは <xsd:string> にマップされ、ヌル終了ストリングとして処理されます。CICS は、SOAP メッセージから変換する際に、終了ヌル文字を追加します。文字ストリングの最大長は、言語構造に示される長さより 1 文字少ないものとして計算されます。NULL は、C/C++ における **CHAR-VARYING** パラメーターのデフォルト値です。

COLLAPSE

文字フィールドは <xsd:string> にマップされます。フィールドの末尾の空白は SOAP メッセージに含まれません。マッピング・レベル 2.1 以降の COBOL および PL/I では、**CHAR-VARYING** パラメーターのデフォルト値は COLLAPSE です。

BINARY

文字フィールドは <xsd:base64binary> にマップされ、固定長フィールドとして処理されます。**CHAR-VARYING** パラメーターで BINARY 値が使用できるのは、マッピング・レベル 2.1 以降のみです。

CONTID=value

サービス・プロバイダーで、SOAP メッセージを表示するために使用される最上位のデータ構造を格納するコンテナの名前を指定します。

DATA-TRUNCATION=DISABLED|ENABLED

固定長フィールド構造で可変長データを許容するかどうかを指定します。

DISABLED

データが、CICS が予期する固定長未満である場合、CICS は切り捨てられたデータをリジェクトし、エラー・メッセージを発行します。

ENABLED

データが、CICS が予期する固定長未満である場合、CICS は切り捨てられたデータを許容し、欠落データをヌル値として処理します。

DATETIME=UNUSED|PACKED15

高水準言語構造の dateTime フィールドをタイム・スタンプとしてマップするかどうかを、次のように指定します。

PACKED15

すべての dateTime フィールドがタイム・スタンプとしてマップされます。

UNUSED

どの dateTime フィールドも、タイム・スタンプとしてマップされません。

このパラメーターは、マッピング・レベル 3.0 で設定できます。

HTTPPROXY={domain name:port number}|{IP address:port number}

WSDL に、インターネット上にある他の WSDL ファイルへの参照が含まれており、DFHLS2WS を実行しているシステムがプロキシ・サーバーを使用してインターネットにアクセスする場合は、そのプロキシ・サーバーのドメイン名、IP アドレス、およびポート番号を指定します。例を次に示します。

HTTPPROXY=proxy.example.com:8080

その他の場合、このパラメーターは必要ありません。

HTTPPROXY-PASSWORD=value

HTTP プロキシ・パスワードを指定します。DFHLS2WS を実行するシステムが HTTP プロキシ・サーバーを使ってインターネットにアクセスする場合、HTTP プロキシ・サーバーが基本認証を使用するならば、

HTTPPROXY-USERNAME と共にこのパスワードを使用する必要があります。

HTTPPROXY も指定した場合にのみ、このパラメーターを使用できます。

HTTPPROXY-USERNAME=value

HTTP プロキシ・ユーザー名を指定します。 DFHLS2WS を実行するシステムが HTTP プロキシ・サーバーを使ってインターネットにアクセスする場合、HTTP プロキシ・サーバーが基本認証を使用するならば、

HTTPPROXY-PASSWORD と共にこのユーザー名を使用する必要があります。

HTTPPROXY も指定した場合にのみ、このパラメーターを使用できます。

LANG=COBOL

高水準言語構造のプログラム言語が COBOL であることを指定します。

LANG=PLI-ENTERPRISE

高水準言語構造のプログラム言語が Enterprise PL/I であることを指定します。

LANG=PLI-OTHER

高水準言語構造のプログラム言語が Enterprise PL/I 以外の PL/I の水準であることを指定します。

LANG=C

高水準言語構造のプログラム言語が C であることを指定します。

LANG=CPP

高水準言語構造のプログラム言語が C++ であることを指定します。

LOGFILE=value

DFHLS2WS がアクティビティー・ログとトレース情報を書き込むファイルの完全修飾 z/OS UNIX 名です。 DFHLS2WS は、このファイルが存在しない場合、ディレクトリー構造は作成しませんがファイルを作成します。

通常はこのファイルを使用しませんが、DFHLS2WS に問題が発生した場合、このファイルの提出を IBM のサービス組織から依頼される場合があります。

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0}

Web サービス・バインディング・ファイルおよび Web サービス記述を生成する際に、DFHLS2WS が使用するマッピングのレベルを指定します。以下のオプションを選択できます。

- 1.0** このマッピング・レベルはデフォルトです。Web サービス・バインディング・ファイルは CICS TS 3.1 マッピング・レベルを使用して生成されることを示します。
- 1.1** このマッピングは、この特定のレベルでバインディング・ファイルを再生成するために使用します。
- 1.2** このマッピング・レベルでは、**CHAR-VARYING** パラメーターを使用して、実行時に文字配列が処理される方法を制御します。 **VARYING** および **VARYINGZ** 配列も PL/I でサポートされます。
- 2.0** このマッピング・レベルを CICS TS 3.2 領域 (またはそれ以上) で使用すると、言語構造と Web サービス・バインディング・ファイルの間のマッピング機能強化を利用できます。
- 2.1** このマッピング・レベルは、APAR PK59794 が適用済みの CICS TS 3.2 領域、または CICS TS 3.2 より上位の領域で使用します。このマッピング・レベルでは、**CHAR-VARYING** パラメーターの新しい値 **COLLAPSE** および **BINARY** を利用できます。 COBOL の **FILLER** フィールドと PL/I の * フィールドはこのマッピング・レベルでは体系的

に無視されます。生成される WSDL 文書にはこれらフィールドが示されず、相応のギャップが実行時にデータ構造に残されます。

2.2 このマッピング・レベルを APAR PK69738 適用済みの CICS TS 3.2 領域、または CICS TS 3.2 より上位の領域で使用すると、DFHWS2LS 使用時のマッピング機能強化を利用できます。

3.0 このマッピング・レベルは CICS TS 4.1 領域と共に使用します。このマッピング・レベルでは、**REQUEST-CHANNEL** および **RESPONSE-CHANNEL** パラメーターを設定することにより、インターフェースで多数のコンテナを使用するアプリケーションから Web サービスを作成できます。また、**DATETIME** パラメーターを設定することにより、dateTime フィールドを XML タイム・スタンプにマップできます。

MINIMUM-RUNTIME-LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|CURRENT}

Web サービス・バインディング・ファイルを配置できる、最低限の CICS 実行時環境を指定します。指定してある他のパラメーターと一致しないレベルを選択すると、エラー・メッセージを受け取ります。以下のオプションを選択できます。

MINIMUM

指定したパラメーターを前提として、最低限可能な CICS 実行時レベルが自動的に割り振られます。

1.0 生成された Web サービス・バインディング・ファイルが、APAR PK15904 および PK23547 を適用していない CICS TS 3.1 領域に正常に配置されます。一部のパラメーターは、このランタイム・レベルでは使用できません。

1.1 生成された Web サービス・バインディング・ファイルが、少なくとも APAR PK15904 を適用した CICS TS 3.1 領域に正常に配置されます。MAPPING-LEVEL パラメーターには、マッピング・レベル 1.1 以下を使用できます。一部のパラメーターは、このランタイム・レベルでは使用できません。

1.2 生成された Web サービス・バインディング・ファイルが、APAR PK15904 および PK23547 の両方を適用している CICS TS 3.1 領域に正常に配置されます。MAPPING-LEVEL パラメーターには、マッピング・レベル 1.2 以下を使用できます。一部のパラメーターは、このランタイム・レベルでは使用できません。

2.0 生成された Web サービス・バインディング・ファイルが、CICS TS 3.2 領域またはそれ以上で正常に配置されます。MAPPING-LEVEL パラメーターには、マッピング・レベル 2.0 以下を使用できます。一部のパラメーターは、このランタイム・レベルでは使用できません。

2.1 生成された Web サービス・バインディング・ファイルが、APAR PK59794 適用済みの CICS TS 3.2 領域、または CICS TS 3.2 より上位の領域に正常に配置されます。MAPPING-LEVEL パラメーターにはマッピング・レベル 2.1 以下を使用できます。このレベルでは任意のオプション・パラメーターを使用できます。

2.2 生成された Web サービス・バインディング・ファイルが、APAR

PK69738 適用済みの CICS TS 3.2 領域、または CICS TS 3.2 より上位の領域に正常に配置されます。この実行時レベルでは、**MAPPING-LEVEL** パラメーターにマッピング・レベル 2.2 以下を使用できます。このレベルでは任意のオプション・パラメーターを使用できます。

- 3.0** 生成された Web サービス・バインディング・ファイルが、CICS TS 4.1 領域またはそれ以上で正常に配置されます。この実行時レベルでは、**MAPPING-LEVEL** パラメーターにマッピング・レベル 3.0 以下を使用できます。このレベルでは任意のオプション・パラメーターを使用できます。

CURRENT

生成された Web サービス・バインディング・ファイルは、Web サービス・バインディング・ファイルの生成に使用するものと同じ実行時レベルで、CICS 領域に正常に配置されます。

PDSLIB=value

処理の対象となる高水準言語データ構造が格納されている区分データ・セットの名前を指定します。要求および応答に使用されるデータ・セット・メンバーは、それぞれ、**REQMEM** パラメーターおよび **RESPMEM** パラメーターで指定されます。

制約事項: 区分データ・セット内のレコードは、80 バイトの固定長にする必要があります。

PDSCP=value

REQMEM および **RESPMEM** パラメーターに指定される区分データ・セット・メンバーで使用されるコード・ページを指定します。ここで、*value* は CCSID 番号または Java コード・ページ番号です。このパラメーターを指定しない場合は、z/OS UNIX システム・サービスのコード・ページが使用されます。例えば、**PDSCP=037** と指定できます。

PGMINT=CHANNEL|COMMAREA

サービス・プロバイダーの場合は、CICS がターゲット・アプリケーション・プログラムにデータを渡す方法を次のように指定します。

CHANNEL

CICS は、チャネル・インターフェースを使用して、データをターゲット・アプリケーション・プログラムに渡します。

チャネルには、入力と出力の両方に使用される、1 つのコンテナのみが含まれます。**CONTID** パラメーターを使用してコンテナの名前を指定します。デフォルトの名前は DFHWS-DATA です。

COMMAREA

CICS は、通信域を使用して、データをターゲット・アプリケーション・プログラムに渡します。

PGMNAME=value

Web サービスとして公開されるターゲット・アプリケーション・プログラムの、CICS PROGRAM リソースの名前を指定します。これは、CICS Web サービス・サポートのリンク先となるプログラムです。

REQMEM=value

Web サービス要求の高水準言語データ構造が格納されている区分データ・セット・メンバーの名前を指定します。サービス・プロバイダーの場合、Web サービス要求は、アプリケーション・プログラムの入力になります。

REQUEST-CHANNEL=value

チャンネル記述文書の名前と場所を指定します。Web サービス・プロバイダー・アプリケーションは、Web サービス・リクエスターから SOAP メッセージを受け取る際に、チャンネル記述に記述されているコンテナをインターフェースで使うことができます。チャンネル記述は、CICS 提供のチャンネル・スキーマに準拠する必要がある XML 文書です。

このパラメーターはマッピング・レベル 3.0 のみで使用できます。

REQUEST-NAMESPACE=value

生成される Web サービス記述に、要求メッセージについての XML スキーマのネーム・スペースを指定します。このパラメーターを指定しない場合は、CICS が自動的にネーム・スペースを生成します。

RESPMEM=value

Web サービス応答の高水準言語データ構造が格納されている区分データ・セット・メンバーの名前を指定します。サービス・プロバイダーの場合、Web サービス応答は、アプリケーション・プログラムの出力になります。

応答が存在しない場合（つまり片方向のメッセージの場合）は、このパラメーターを省略します。

RESPONSE-CHANNEL=value

チャンネル記述文書の名前と場所を指定します。Web サービス・プロバイダー・アプリケーションは、Web サービス・リクエスターに SOAP 応答メッセージを送る際に、チャンネル記述に記述されているコンテナをインターフェースで使うことができます。チャンネル記述は、CICS 提供のチャンネル・スキーマに準拠する必要がある XML 文書です。

このパラメーターはマッピング・レベル 3.0 のみで使用できます。

RESPONSE-NAMESPACE=value

生成される Web サービス記述に、応答メッセージについての XML スキーマのネーム・スペースを指定します。このパラメーターを指定しない場合は、CICS が自動的にネーム・スペースを生成します。

SOAPVER=1.1|1.2|ALL

生成された Web サービス記述で使用する SOAP レベルを指定します。このパラメーターは **MINIMUM-RUNTIME-LEVEL** が 2.0 以上に設定されているときにのみ使用可能です。

1.1 Web サービス記述のバインディングとして SOAP 1.1 プロトコルを使用します。

1.2 Web サービス記述のバインディングとして SOAP 1.2 プロトコルを使用します。

ALL Web サービス記述のバインディングとして SOAP 1.1 または 1.2 プロトコルの両方を使用できます。

このパラメーターに値を指定しない場合は、作成したい WSDL のバージョンに応じてデフォルト値が異なります。

- WSDL 1.1 だけが必要な場合は、SOAP 1.1 バインディングが使用されます。
- WSDL 2.0 だけが必要な場合は、SOAP 1.2 バインディングが使用されます。
- WSDL 1.1 と WSDL 2.0 の両方が必要な場合は、各 Web サービス記述について、SOAP 1.1 と 1.2 両方のバインディングが使用されます。

SSL-KEYSTORE=*value*

このオプション・パラメーターは、鍵ストア・ファイルの完全修飾された場所を指定します。

Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用してネットワークを介して IBM WebSphere Service Registry and Repository (WSRR) と通信する場合には、このパラメーターを使用してください。

SSL-KEYPWD=*value*

このオプション・パラメーターは、鍵ストアのパスワードを指定します。

Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用してネットワークを介して IBM WebSphere Service Registry and Repository (WSRR) と通信する場合には、このパラメーターを使用してください。

SSL-TRUSTSTORE=*value*

このオプション・パラメーターは、トラストストア・ファイルの完全修飾された場所を指定します。

Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用してネットワークを介して IBM WebSphere Service Registry and Repository (WSRR) と通信する場合には、このパラメーターを使用してください。

SSL-TRUSTPWD=*value*

このオプション・パラメーターは、トラストストアのパスワードを指定します。

Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用してネットワークを介して IBM WebSphere Service Registry and Repository (WSRR) と通信する場合には、このパラメーターを使用してください。

STRUCTURE=*(request,response)*

C と C++ の場合にのみ、**REQMEM** および **RESPMEM** パラメーターに指定された区分データ・セットのメンバーに格納されている高水準言語データ構造の名前を指定します。

request

REQMEM パラメーターを指定する場合に、要求を格納する高水準言語データ構造の名前を指定します。デフォルト値は **DFHREQUEST** です。

区分データ・セット・メンバーは、指定した名前を持つ高水準言語データ構造や名前を指定しない場合は **DFHREQUEST** という名前の構造を格納している必要があります。

response

RESPMEM パラメーターを指定する場合に、応答を格納する高水準言語データ構造の名前を指定します。デフォルト値は **DFHRESPONSE** です。

値を指定する場合、区分データ・セット・メンバーが、指定した名前を持つ高水準言語データ構造や名前を指定しない場合は DFHRESPONSE という名前の構造を格納している必要があります。

SYNCONRETURN=NO|YES

リモート Web サービスが同期点を発行できるかどうかを指定します。

NO リモート Web サービスは同期点を発行できません。これはデフォルトの値です。リモート Web サービスが同期点を発行すると、ADPL 異常終了になり失敗します。

YES リモート Web サービスは同期点を発行できます。YES を選択した場合、リモート Web サービスから制御が戻されたときにリモート・タスクが別個の作業単位としてコミットされます。リモート Web サービスがリカバリー可能リソースを更新して戻した後に障害が発生した場合、そのリソースの更新内容をバックアウトすることはできません。

TRANSACTION=name

サービス・プロバイダーで、このパラメーターは、応答を組み立てるためにパイプラインを開始できる 1 から 4 文字の別名トランザクションの名前を指定します。このパラメーターの値は、URIMAP リソースが PIPELINE スキャン・コマンドを使用して自動的に作成される際に、URIMAP リソースの TRANSACTION 属性を定義するために使用されます。

許容文字:

A - Z、a - z、0 - 9、\$、@、#、_、<、>

URI=value

このパラメーターはクライアントが Web サービスのアクセスに使用することになる相対または絶対 URI を指定します。CICS は、DFHLS2WS によって作成された Web サービス・バインディング・ファイルから URIMAP リソースを生成するときに指定された値を使用します。このパラメーターは URIMAP 定義が適用される URI のパスのコンポーネントを指定します。

USERID=id

サービス・プロバイダーで、このパラメーターは、任意の Web クライアントによって使用できる 1 から 8 文字のユーザー ID を指定します。アプリケーション生成の応答または Web サービスでは、別名トランザクションがこのユーザー ID の下に付加されます。このパラメーターの値は、URIMAP リソースが PIPELINE スキャン・コマンドを使用して自動的に作成される際に、URIMAP リソースの USERID 属性を定義するために使用されます。

許容文字:

A - Z、a - z、0 - 9、\$、@、#

WSBIND=value

Web サービス・バインディング・ファイルの完全修飾 z/OS UNIX 名です。DFHLS2WS は、このファイルが存在しない場合、ディレクトリ構造は作成しませんがファイルを作成します。ファイル拡張子は .wsbind です。

WSDL=value

Web サービス記述を書き込むファイルの完全修飾 z/OS UNIX 名です。Web サービス記述は WSDL 1.1 仕様に準拠します。DFHLS2WS は、このファイルが存在しない場合、ディレクトリー構造は作成しませんがファイルを作成します。ファイル拡張子は .wsdl です。

WSDL_1.1=value

Web サービス記述を書き込むファイルの完全修飾 z/OS UNIX 名です。Web サービス記述は WSDL 1.1 仕様に準拠します。DFHLS2WS は、このファイルが存在しない場合、ディレクトリー構造は作成しませんがファイルを作成します。ファイル拡張子は .wsdl です。このパラメーターと **WSDL** パラメーターの結果は同じです。どちらか 1 つのみを指定できます。

WSDL_2.0=value

Web サービス記述を書き込むファイルの完全修飾 z/OS UNIX 名です。Web サービス記述は WSDL 2.0 仕様に準拠します。DFHLS2WS は、このファイルが存在しない場合、ディレクトリー構造は作成しませんがファイルを作成します。ファイル拡張子は .wsdl です。このパラメーターは、**WSDL** または **WSDL_1.1** パラメーターと使用できます。このパラメーターは **MINIMUM-RUNTIME-LEVEL** が 2.0 以上に設定されているときにのみ使用可能です。

WSDLCP=LOCAL|UTF-8|EBCDIC-CP-US

WSDL 文書を生成するために使用するコード・ページを指定します。

LOCAL

WSDL 文書をローカル・コード・ページを使用して生成し、WSDL 文書にはエンコード・タグが生成されないように指定します。

UTF-8 WSDL 文書を UTF-8 コード・ページを使用して生成するように指定します。エンコード・タグが WSDL 文書に生成されます。このオプションを指定する場合、異なるプラットフォーム間で WSDL 文書をコピーするときに正しいエンコードが保持されることを確認する必要があります。

EBCDIC-CP-US

この値は、WSDL 文書を US EBCDIC コード・ページを使用して生成するように指定します。エンコード・タグが WSDL 文書に生成されます。

WSDL-NAMESPACE=value

生成される WSDL 文書で使われる CICS の名前空間を指定します。

このパラメーターを指定しない場合は、CICS が自動的にネーム・スペースを生成します。

WSRR-CUSTOM-PropertyName=value

このオプション・パラメーターを使用して、カスタマイズされたメタデータを WSRR の WSDL 文書に追加できます。WSDL 文書に **WSRR-CUSTOM-PropertyName=value** という組が追加されて、WSSR-CUSTOM 接頭部なしで WSRR に表示されます。

最大で 255 個のカスタマイズされた *PropertyName=value* という組を指定できます。重複する *PropertyName=value* の組、およびブランクの組を指定しないでください。

このパラメーターは、**WSRR-SERVER** パラメーターが指定された場合にのみ使用します。

WSRR-DESCRIPTION=value

このオプション・パラメーターを使用すると、公開される WSDL 文書を記述するメタデータを指定できます。

このパラメーターは、**WSRR-SERVER** パラメーターが指定された場合にのみ使用します。

WSRR-ENCODING=value

このオプション・パラメーターを使用して、WSDL 文書の文字セット・エンコードを指定します。**WSRR-ENCODING** パラメーターが指定されない場合、WSRR は WSDL 文書で指定された値を使用します。

このパラメーターは、**WSRR-SERVER** パラメーターが指定された場合にのみ使用します。

WSRR-LOCATION=value

このオプション・パラメーターを使用して、WSDL 文書の場所を識別する URI を指定します。このパラメーターを指定しない場合、**WSDL** パラメーターで指定されているファイル名がデフォルト URI として使われます。例えば **WSDL** パラメーターの値が *wsrr/example.wsdl* である場合、**WSRR-LOCATION** パラメーターのデフォルト値は *example.wsdl* になります。

このパラメーターは、**WSRR-SERVER** パラメーターが指定された場合にのみ使用します。

WSRR-PASSWORD=value

WSRR へのアクセスにパスワードの入力が必要な場合は、このオプション・パラメーターを使用します。

WSRR-USERNAME パラメーターを指定する場合には、このパラメーターも指定する必要があります。

このパラメーターは、**WSRR-SERVER** パラメーターが指定された場合にのみ使用します。

WSRR-SERVER={domain name:port number}/{IP address:port number}

このパラメーターを使用して、IBM WebSphere Service Registry and Repository (WSRR) サーバーの場所を指定します。このパラメーターを指定した場合、WSRR パラメーター検証が使用されます。

WSRR-USERNAME=value

WSRR へのアクセスでユーザー名を指定する必要がある場合は、このオプション・パラメーターを使用します。WSRR はこのユーザー名を使って所有者プロパティを設定します。

このパラメーターは、**WSRR-SERVER** パラメーターが指定された場合にのみ使用します。

WSRR-VERSION=1|value

このパラメーターを使用して、WSRR の WSDL 文書のバージョン・プロパティを設定します。

このパラメーターは、**WSRR-SERVER** パラメーターが指定された場合にのみ使用します。

その他の情報

- DFHLS2SC を実行するユーザー ID は、UNIX システム・サービスを使用するように構成されていなければなりません。このユーザー ID には、CICS z/OS UNIX ファイル構造および PDS ライブラリーに対する読み取り権限と **LOGFILE**、**WSBIND**、および **WSDL** パラメーターに指定されたディレクトリーに対する書き込み権限が必要です。
- Java を実行するため、このユーザー ID には十分な大きさのストレージを割り振る必要があります。

例

```
//LS2WS JOB 'accounting information',name,MSGCLASS=A
// SET QT='''
//JAVAPROG EXEC DFHLS2WS,
// TMPFILE=&QT.&SYSUID.&QT
//INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
REQMEM=DFH0XCP4
RESPMEM=DFH0XCP4
LANG=COBOL
LOGFILE=/u/exampleapp/wsbinding/example.log
MINIMUM-RUNTIME-LEVEL=2.1
MAPPING-LEVEL=2.1
CHAR-VARYING=COLLAPSE
PGMNAME=DFH0XCMN
URI=http://myserver.example.org:8080/exampleApp/example
PGMINT=COMMAREA
SOAPVER=ALL
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbinding/example.wsbinding
WSDL=/u/exampleapp/wsd1/example.wsd1
WSDL_2.0=/u/exampleapp/wsd1/example_20.wsd1
WSDLCP=LOCAL
WSDL-NAMESPACE=http://mywsdlnamespace
/*
```

DFHWS2LS: WSDL から高水準言語への変換

DFHWS2LS プロシージャは Web サービス記述から高水準言語データ構造および Web サービス・バインディング・ファイルを生成します。CICS アプリケーション・プログラムをサービス・プロバイダーとして公開する場合、またはサービス・リクエスターを作成する場合に、DFHWS2LS を使用することができます。

DFHWS2LS のジョブ制御ステートメント、そのシンボリック・パラメーターと入力パラメーター、それらについての説明、およびサンプル・ジョブを、このプロシージャの使用に役立てることができます。

DFHWS2LS のジョブ制御ステートメント

JOB ジョブを開始します。

EXEC プロシージャ名 (DFHWS2LS) を指定します。

INPUT.SYSUT1 DD

入力を指定します。入力パラメーターは、通常、入力ストリーム内に指定します。ただし、データ・セットや区分データ・セットのメンバーに定義することもできます。

シンボリック・パラメーター

以下のシンボリック・パラメーターは、DFHWS2LS で定義されます。

JAVADIR=*path*

DFHWS2LS によって使用される Java ディレクトリーの名前を指定します。このパラメーターの値は /usr/lpp/ に追加され、これによって /usr/lpp/*path* という完全なパス名が生成されます。

通常、このパラメーターは指定しません。デフォルト値は、**JAVADIR** パラメーターの CICS インストール・ジョブ (DFHISTAR) に指定された値です。

PATHPREFIX=*prefix*

他のパラメーターで使用される z/OS UNIX ディレクトリー・パスを拡張するオプションの接頭部を指定します。デフォルトは空ストリングです。

通常、このパラメーターは指定しません。デフォルト値は、**JAVADIR** パラメーターの CICS インストール・ジョブ (DFHISTAR) に指定された値です。

TMPDIR=*tmpdir*

DFHWS2LS が一時ワークスペースとして使用する z/OS UNIX のディレクトリーの場所を指定します。このジョブを実行するユーザー ID には、このディレクトリーに対する読み取り権限および書き込み権限が必要です。

デフォルト値は /tmp です。

TMPPFILE=*tmpprefix*

一時ワークスペース・ファイルの名前を作成するために DFHWS2LS が使用する接頭部を指定します。

デフォルト値は WS2LS です。

USSDIR=*path*

UNIX システム・サービスのファイル・システム内の CICS TS ディレクトリーの名前を指定します。このパラメーターの値は /usr/lpp/cicsts/ に追加され、これによって /usr/lpp/cicsts/*path* という完全なパス名が生成されます。

通常、このパラメーターは指定しません。デフォルト値は、**USSDIR** パラメーターの CICS インストール・ジョブ (DFHISTAR) に指定された値です。

SERVICE=*value*

このパラメーターは IBM サポートに指示された場合にのみ使用します。

一時ワークスペース

DFHWS2LS は、実行時に、次の 3 つの一時ファイルを作成します。

tmpdir/*tmpprefix*.in

tmpdir/*tmpprefix*.out

tmpdir/*tmpprefix*.err

ここで、

tmpdir は、**TMPDIR** パラメーターに指定されている値です。

tmpprefix は、**TMPFILE** パラメーターに指定されている値です。

TMPDIR および **TMPFILE** が指定されていない場合、ファイルのデフォルトの名前は、次のとおりです。

/tmp/WS2LS.in

/tmp/WS2LS.out

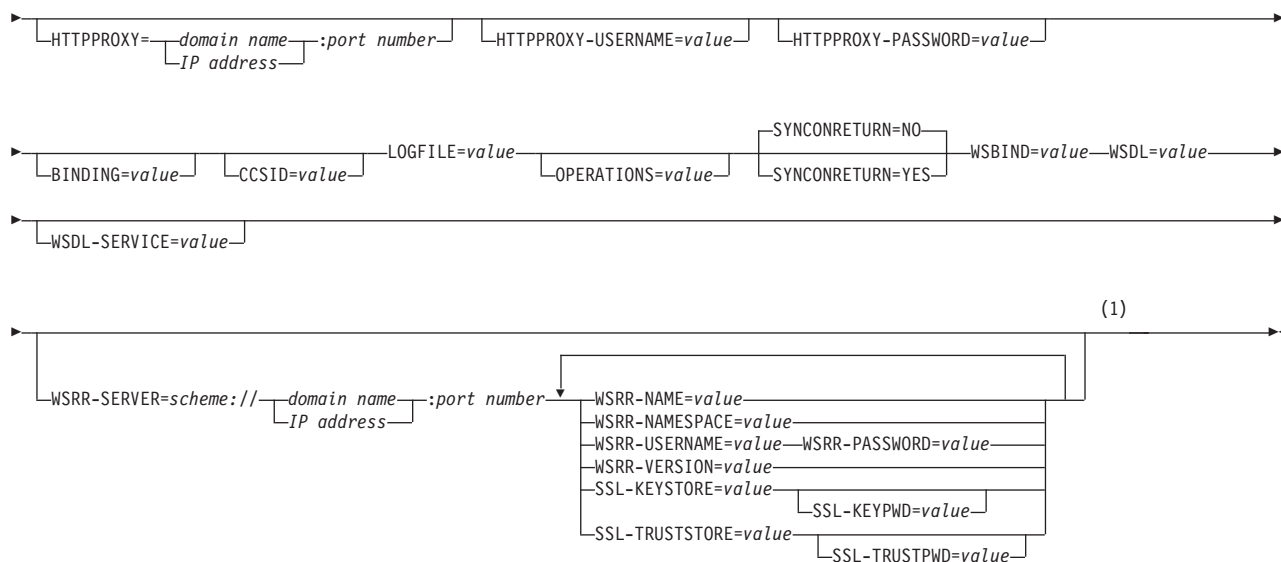
/tmp/WS2LS.err

重要: DFHWS2LS は、生成された z/OS UNIX ファイル名へのアクセスをロックしません。したがって、DFHWS2LS の複数のインスタンスが同時に動作して、同じ一時ワークスペース・ファイルを使用する場合には、あるジョブがワークスペース・ファイルを使っているときに別のジョブがそれを上書きするのを防ぐことはできず、予測不能な障害が発生します。

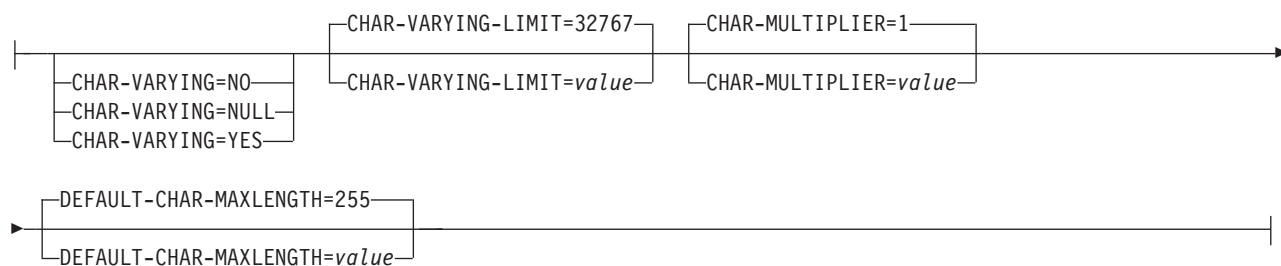
したがって、この状況を回避する命名規則および操作手順を考案することをお勧めします。例えば、システム・シンボリック・パラメーター **SYSUID** を使用すると、個々のユーザーに対して一意のワークスペース・ファイルを生成できます。これらの一時ファイルはジョブの終わりの前に削除されます。

DFHWS2LS のパラメーターの入力





拡張データ・マッピング (マッピング・レベル 1.2 以上):



拡張データ・マッピング (マッピング・レベル 2.1 以上):



拡張データ・マッピング (マッピング・レベル 2.2 以上):



拡張データ・マッピング (実行時レベル 2.1 以上):



拡張データ・マッピング (実行時レベル 3.0 以上):



注:

- 1 **WSRR-SERVER** パラメーターの設定時に指定できるそれぞれの WSRR パラメーターは、一度だけ指定可能です。

パラメーターの使用法

- 入力パラメーターの指定順序は自由です。
- 各パラメーターは改行後に記述を始める必要があります。
- パラメーターおよび使用する場合は継続文字は、72 列を超えてはなりません。列 73 から 80 はブランクにする必要があります。
- パラメーターが長すぎて 1 行に収まらない場合は、行の末尾にアスタリスク文字 (*) を使用して、そのパラメーターが次の行に続くことを示します。スペースを含むアスタリスクより前の文字はすべてパラメーターの一部とみなされます。例を次に示します。

```
WSBIND=wsbinddir*  
/app1
```

このコードは、次のコードと同じ意味になります。

```
WSBIND=wsbinddir/app1
```

- 行の先頭の文字の位置に # という文字がある場合、この文字はコメント文字を表します。この行は無視されます。

パラメーターの記述

BINDING=value

Web サービス記述に複数の <wsdl:Binding> エlementが格納されている場合は、このパラメーターを使用して、言語構造および Web サービス・バインディング・ファイルを生成するためにどのElementを使用するかを指定します。

Web サービス記述の <wsdl:Binding> Elementに使用される **name** 属性の値を指定します。

CCSID=value

アプリケーション・データ構造に文字データをエンコードするために、実行時に使用される CCSID を指定します。このパラメーターの値は、**LOCALCCSID** システム初期設定パラメーターの値を指定変更します。*value* は、Java および z/OS 変換サービスでサポートされる EBCDIC CCSID である必要があります。このパラメーターを指定しない場合、アプリケーション・データ構造は、システム初期設定パラメーターで指定された CCSID を使用してエンコードされます。

このパラメーターは任意のマッピング・レベルで使用できます。ただし、生成されたファイルを CICS TS 3.1 領域に配置するには、APAR PK23547 を適用して、Web サービス・バインディング・ファイルをインストールするために、コードの最小実行時レベルを達成する必要があります。

CHAR-MULTIPLIER=1|value

マッピング・レベルが 1.2 以上のとき、各文字に許可されるバイト数を指定します。このパラメーターの *value* は、1 から 2 147 483 647 の範囲の正整数です。非数字に基づくマッピングはすべて、この乗数の対象です。バイナリー、数値、ゾーンおよびパック 10 進数フィールドは、この乗数の対象ではありません。

このパラメーターが役に立つのは、例えば、実行時にすべての 2 バイト文字の周囲に潜在的なシフトアウト文字とシフトイン文字のスペースを許可するために、乗数 3 を選択できる DBCS 文字の使用を計画している場合などです。

CHAR-VARYING=NO|NULL|YES

マッピング・レベルが 1.2 以上のとき、可変長文字データがマップされる方法を指定します。可変長バイナリー・データ・タイプは、常にコンテナまたは可変構造のいずれかにマップされます。このパラメーターを指定しない場合は、指定される言語に応じてデフォルトのマッピングが異なります。以下のオプションを選択できます。

NO 可変長文字データは固定長ストリングとしてマップされます。

NULL 可変長文字データはヌル終了ストリングにマップされます。

YES 可変長文字データは PL/I では CHAR VARYING データ・タイプにマップされます。COBOL、C および C++ 言語では、可変長文字データは、2 つの関連エレメント (データ長およびデータ) から構成される同等の表現にマップされます。

CHAR-VARYING-LIMIT=32767|value

マッピング・レベルが 1.2 以上のとき、言語構造にマップされるバイナリー・データおよび可変長文字データの最大サイズを指定します。文字データまたはバイナリー・データが、このパラメーターで指定される値より大きい場合は、コンテナにマップされ、コンテナ名が生成された言語構造で使用されます。値の範囲は 0 からデフォルトの 32 767 バイトまでです。

CONTID=value

サービス・プロバイダーで、SOAP メッセージを表示するために使用される最上位のデータ構造を格納するコンテナの名前を指定します。

DATA-TRUNCATION=DISABLED|ENABLED

固定長フィールド構造で可変長データを許容するかどうかを指定します。

DISABLED

データが、CICS が予期する固定長未満である場合、CICS は切り捨てられたデータをリジェクトし、エラー・メッセージを発行します。

ENABLED

データが、CICS が予期する固定長未満である場合、CICS は切り捨てられたデータを許容し、欠落データをヌル値として処理します。

DATETIME=PACKED15|STRING

<xsd:dateTime> エレメントを言語構造にマップする方法を指定します。

PACKED15

デフォルトでは、すべての <xsd:dateTime> エレメントがタイム・スタンプとして処理され、CICS ABSTIME 形式にマップされます。

STRING

<xsd:dateTime> エレメントはテキストとして処理されます。

DEFAULT-CHAR-MAXLENGTH=255|value

マッピング・レベルが 1.2 以上のとき、Web サービス記述文書に長さが暗黙指定されていない場合に、マッピングについて文字データのデフォルトの配列長を文字数で指定します。このパラメーターの値は、1 から 2 147 483 647 の範囲の正整数です。

HTTPPROXY={domain name:port number}|{IP address:port number}

WSDL に、インターネット上にある他の WSDL ファイルへの参照が含まれており、DFHWS2LS を実行しているシステムがプロキシ・サーバーを使用してインターネットにアクセスする場合は、そのプロキシ・サーバーのドメイン名、IP アドレス、およびポート番号を指定します。例を次に示します。

HTTPPROXY=proxy.example.com:8080

その他の場合、このパラメーターは必要ありません。

HTTPPROXY-PASSWORD=value

HTTP プロキシ・パスワードを指定します。DFHWS2LS を実行するシステムが HTTP プロキシ・サーバーを使ってインターネットにアクセスする場合、HTTP プロキシ・サーバーが基本認証を使用するならば、

HTTPPROXY-USERNAME と共にこのパスワードを使用する必要があります。

HTTPPROXY も指定した場合にのみ、このパラメーターを使用できます。

HTTPPROXY-USERNAME=value

HTTP プロキシ・ユーザー名を指定します。DFHWS2LS を実行するシステムが HTTP プロキシ・サーバーを使ってインターネットにアクセスする場合、HTTP プロキシ・サーバーが基本認証を使用するならば、

HTTPPROXY-PASSWORD と共にこのユーザー名を使用する必要があります。

HTTPPROXY も指定した場合にのみ、このパラメーターを使用できます。

INLINE-MAXOCCURS-LIMIT=1|value

maxOccurs 属性に基づいて、インラインの可変コンテンツを繰り返し使用するかどうかを指定します。インラインにマップされる可変の繰り返しコンテンツは、生成される言語構造の残りの部分と共に現在のコンテナに入ります。可変の繰り返しコンテンツは、2 つの部分 (データの出現回数を格納するカウンターと、データのそれぞれの出現を格納する配列) に分けて格納されます。可変繰り返しコンテンツに代わるマッピングとして、コンテナ・ベースのマッピングがあります。この場合、データの出現回数、およびデータを収容するコンテナの名前を格納します。別個のコンテナにデータを格納するとパフォーマンスに影響を与えるため、インライン・マッピングの方が適しているかもしれません。

INLINE-MAXOCCURS-LIMIT パラメーターは、マッピング・レベル 2.1 以降でのみ使用できます。**INLINE-MAXOCCURS-LIMIT** の値は、0 から 32 767 の範囲の正整数です。値 0 は、インライン・マッピングを使用しないことを示します。値 1 を使用すると、オプション・エレメントがインラインでマップされます。maxOccurs 属性の value が **INLINE-MAXOCCURS-LIMIT** の value より大きい場合、コンテナに基づくマッピングが使用されます。それ以外の場合はインライン・マッピングが使用されます。

可変の繰り返しリストをインラインでマップするかどうか決定する際には、繰り返されるデータの単一項目の長さを考慮してください。長い項目が少ない回数だ

け出現する場合は、コンテナに基づくマッピングが適しています。短い項目が数多く出現する場合は、インライン・マッピングが適しています。

LANG=COBOL

高水準言語構造のプログラム言語が COBOL であることを指定します。

LANG=PLI-ENTERPRISE

高水準言語構造のプログラム言語が Enterprise PL/I であることを指定します。

LANG=PLI-OTHER

高水準言語構造のプログラム言語が Enterprise PL/I 以外の PL/I の水準であることを指定します。

LANG=C

高水準言語構造のプログラム言語が C であることを指定します。

LANG=CPP

高水準言語構造のプログラム言語が C++ であることを指定します。

LOGFILE=value

DFHWS2LS がアクティビティー・ログとトレース情報を書き込むファイルの完全修飾 z/OS UNIX 名です。DFHWS2LS は、このファイルが存在しない場合、ディレクトリー構造は作成しませんがファイルを作成します。

通常はこのファイルを使用しませんが、DFHWS2LS に問題が発生した場合、このファイルの提出を IBM のサービス組織から依頼される場合があります。

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.3|0}

Web サービス・バインディング・ファイルおよび言語構造を生成する際に、DFHWS2LS が使用するマッピングのレベルを指定します。以下のオプションを選択できます。

- 1.0** Web サービス・バインディング・ファイルおよび言語構造は CICS TS 3.1 マッピング・レベルを使用して生成されます。
- 1.1** XML 属性や <list> および <union> データ・タイプは言語構造にマップされます。最大長が 32 767 バイトより大きい文字データおよびバイナリー・データはコンテナにマップされます。コンテナ名は言語構造内に作成されます。
- 1.2** **CHAR-VARYING** および **CHAR-VARYING-LIMIT** パラメーターを使用して、実行時に文字データがマップおよび処理される方法を制御します。このパラメーターのどちらも指定しない場合、最大長が 32 768 バイトより小さいバイナリー・データおよび文字データは、C++ を除くすべての言語で VARYING 構造にマップされます。C++ では、文字データはヌル終了ストリングにマップされます。
- 2.0** このマッピング・レベルを CICS TS 3.2 領域 (またはそれ以上) で使用すると、言語構造と Web サービス・バインディング・ファイルの間のマッピング機能強化を利用できます。
- 2.1** このマッピング・レベルを APAR PK59794 適用済みの CICS TS 3.2 領域、または CICS TS 3.2 より上位の領域で使用すると、<xsd:any> および xsd:anyType がサポートされることに加えて、**INLINE-MAXOCCURS-LIMIT** パラメーターを使って可変の繰り返しコ

ンテンツをインラインでマップするオプションを使用でき、
<xsd:sequence>、<xsd:choice>、<xsd:all> で minOccurs="0" がサポート
されます。

2.2 このマッピング・レベルは、APAR PK69738 が適用済みの CICS TS
3.2 領域、または CICS TS 3.2 より上位の領域で使用します。以下の機
能がサポートされます。

- 固定値を持つエレメント
- <xsd:choice> エレメントの拡張サポート
- 抽象データ・タイプ
- 抽象エレメント
- 置換グループ

3.0 このマッピング・レベルは CICS TS 4.1 領域と共に使用します。この
マッピング・レベルでは、タイム・スタンプを CICS ABSTIME 形式に
変換することができます。

MINIMUM-RUNTIME-LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|CURRENT}

Web サービス・バインディング・ファイルを配置できる、最低限の CICS 実行
時環境を指定します。指定してある他のパラメーターと一致しないレベルを選択
すると、エラー・メッセージを受け取ります。以下のオプションを選択できま
す。

MINIMUM

指定したパラメーターを前提として、最低限可能な CICS 実行時レベル
が自動的に割り振られます。

- 1.0** 生成された Web サービス・バインディング・ファイルが、APAR
PK15904 および PK23547 を適用していない CICS TS 3.1 領域に正常
に配置されます。一部のパラメーターは、このランタイム・レベルでは
使用できません。
- 1.1** 生成された Web サービス・バインディング・ファイルが、少なくとも
APAR PK15904 を適用した CICS TS 3.1 領域に正常に配置されます。
MAPPING-LEVEL パラメーターには、マッピング・レベル 1.1 以下を
使用できます。一部のパラメーターは、このランタイム・レベルでは使
用できません。
- 1.2** 生成された Web サービス・バインディング・ファイルが、APAR
PK15904 および PK23547 の両方を適用している CICS TS 3.1 領域に
正常に配置されます。MAPPING-LEVEL パラメーターには、マッピン
グ・レベル 1.2 以下を使用できます。一部のパラメーターは、このラン
タイム・レベルでは使用できません。
- 2.0** 生成された Web サービス・バインディング・ファイルが、CICS TS
3.2 領域またはそれ以上で正常に配置されます。MAPPING-LEVEL パラ
メーターには、マッピング・レベル 2.0 以下を使用できます。一部のパ
ラメーターは、このランタイム・レベルでは使用できません。
- 2.1** 生成された Web サービス・バインディング・ファイルが、APAR
PK59794 適用済みの CICS TS 3.2 領域、または CICS TS 3.2 より上
位の領域に正常に配置されます。 **MAPPING-LEVEL** パラメーターに

はマッピング・レベル 2.1 以下を使用できます。このレベルでは任意のオプション・パラメーターを使用できます。

2.2 生成された Web サービス・バインディング・ファイルが、APAR PK69738 適用済みの CICS TS 3.2 領域、または CICS TS 3.2 より上位の領域に正常に配置されます。この実行時レベルでは、**MAPPING-LEVEL** パラメーターにマッピング・レベル 2.2 以下を使用できます。このレベルでは任意のオプション・パラメーターを使用できます。

3.0 生成された Web サービス・バインディング・ファイルが、CICS TS 4.1 領域またはそれ以上で正常に配置されます。この実行時レベルでは、**MAPPING-LEVEL** パラメーターにマッピング・レベル 3.0 以下を使用できます。このレベルでは任意のオプション・パラメーターを使用できます。

CURRENT

生成された Web サービス・バインディング・ファイルは、Web サービス・バインディング・ファイルの生成に使用するものと同じ実行時レベルで、CICS 領域に正常に配置されます。

OPERATIONS=value

Web サービス・リクエスター・アプリケーションについて、Web サービス・バインディング・ファイルを生成するために使用される Web サービス記述から、有効な <wsdl:Operation> エレメントのサブセットを指定します。各 <wsdl:Operation> エレメントはスペースで分離します。必要に応じて、リストは複数行にわたることがあります。このパラメーターは WSDL 1.1 文書および WSDL 2.0 文書の両方で使用できます。

PDSLIB=value

生成された高水準言語を含む区分データ・セットの名前を指定します。要求および応答に使用されるデータ・セット・メンバーは、それぞれ、**REQMEM** パラメーターおよび **RESPMEM** パラメーターで指定されます。

PDSCP=value

REQMEM および **RESPMEM** パラメーターに指定される区分データ・セット・メンバーで使用されるコード・ページを指定します。ここで、*value* は CCSID 番号または Java コード・ページ番号です。このパラメーターを指定しない場合は、z/OS UNIX システム・サービスのコード・ページが使用されます。例えば、**PDSCP=037** と指定できます。

PDSMEM=value

以下に示す抽象的なデータ・タイプの高水準言語構造体が格納されている区分データ・セット・メンバーの名前を生成するときに **DFHWS2LS** が使用する 1 から 6 文字の接頭部を指定します。このプログラムは、接頭部に 2 桁の数値を付加することによってメンバー名を生成します。

このパラメーターは、抽象データ・タイプに関連した言語構造の名前を指定するために、マッピング・レベル 2.2 以上で使用します。**PDSMEM** パラメーターを省略した場合、抽象データ・タイプの言語構造の名前は **REQMEM** パラメーターの値を使って指定されます。

PGMINT=CHANNEL|COMMAREA

サービス・プロバイダーの場合は、CICS がターゲット・アプリケーション・プログラムにデータを渡す方法を次のように指定します。

CHANNEL

CICS は、チャンネル・インターフェースを使用して、データをターゲット・アプリケーション・プログラムに渡します。

COMMAREA

CICS は、通信域を使用して、データをターゲット・アプリケーション・プログラムに渡します。

DFHWS2LS からの出力がサービス・リクエスターで使用される場合、このパラメーターは無視されます。

PGMNAME=value

CICS PROGRAM リソースの名前を指定します。

サービス・プロバイダーで使用する Web サービス・バインディング・ファイルを生成するために DFHWS2LS を使用する場合、このパラメーターを必ず指定する必要があります。このパラメーターは、Web サービスとして公開するアプリケーション・プログラムのリソース名を指定します。

サービス・リクエスターで使用する Web サービス・バインディング・ファイルを生成するために DFHWS2LS を使用する場合、このパラメーターを省略します。

REQMEM=value

以下に示す Web サービス要求の高水準言語構造体が格納されている区分データ・セット・メンバーの名前を生成するときに DFHWS2LS が使用する 1 から 6 文字の接頭部を指定します。

- サービス・プロバイダーの場合、Web サービス要求は、アプリケーション・プログラムの入力になります。
- サービス・リクエスターの場合、Web サービス要求は、アプリケーション・プログラムの出力になります。

DFHWS2LS は、操作ごとに、区分データ・セットのメンバーを生成します。このプログラムは、接頭部に 2 桁の数値を付加することによってメンバー名を生成します。

このパラメーターはオプションですが、Web サービス記述に要求の定義が記述されている場合は、指定する必要があります。

RESPMEM=value

以下に示す Web サービス応答の高水準言語構造体が格納されている区分データ・セット・メンバーの名前を生成するときに DFHWS2LS が使用する 1 から 6 文字の接頭部を指定します。

- サービス・プロバイダーの場合、Web サービス応答は、アプリケーション・プログラムの出力になります。
- サービス・リクエスターの場合、Web サービス応答は、アプリケーション・プログラムの入力になります。

DFHWS2LS は、操作ごとに、区分データ・セットのメンバーを生成します。このプログラムは、接頭部に 2 桁の数値を付加することによってメンバー名を生成します。

応答が存在しない場合 (つまり片方向のメッセージの場合) は、このパラメーターを省略します。

SSL-KEYSTORE=value

このオプション・パラメーターは、鍵ストア・ファイルの完全修飾された場所を指定します。

Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用してネットワークを介して IBM WebSphere Service Registry and Repository (WSRR) と通信する場合には、このパラメーターを使用してください。

SSL-KEYPWD=value

このオプション・パラメーターは、鍵ストアのパスワードを指定します。

Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用してネットワークを介して IBM WebSphere Service Registry and Repository (WSRR) と通信する場合には、このパラメーターを使用してください。

SSL-TRUSTSTORE=value

このオプション・パラメーターは、トラストストア・ファイルの完全修飾された場所を指定します。

Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用してネットワークを介して IBM WebSphere Service Registry and Repository (WSRR) と通信する場合には、このパラメーターを使用してください。

SSL-TRUSTPWD=value

このオプション・パラメーターは、トラストストアのパスワードを指定します。

Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用してネットワークを介して IBM WebSphere Service Registry and Repository (WSRR) と通信する場合には、このパラメーターを使用してください。

STRUCTURE=(request,response)

C と C++ の場合にのみ、要求構造体と応答構造体の名前を生成する方法を指定します。

生成された要求構造体と応答構造体には、*requestnn* および *responsenn* という名前が付けられます。ここで、*nn* は、操作ごとに構造体を区別するために生成される数値接尾部を表します。

いずれかの名前または両方の名前を省略した場合、構造体の名前は指定した **REQMEM** パラメーターおよび **RESPMEM** パラメーターを基に生成された区分データ・セット・メンバー名と同じ名前になります。

SYNCONRETURN=NO|YES

リモート Web サービスが同期点を発行できるかどうかを指定します。

NO リモート Web サービスは同期点を発行できません。これはデフォルトの値です。リモート Web サービスが同期点を発行すると、ADPL 異常終了になり失敗します。

YES リモート Web サービスは同期点を発行できます。YES を選択した場合、リモート Web サービスから制御が戻されたときにリモート・タスクが別個の作業単位としてコミットされます。リモート Web サービスがリカバリー可能リソースを更新して戻した後に障害が発生した場合、そのリソースの更新内容をバックアウトすることはできません。

TRANSACTION=*name*

サービス・プロバイダーで、このパラメーターは、応答を組み立てるためにパイプラインを開始できる 1 から 4 文字の別名トランザクションの名前を指定します。このパラメーターの値は、URIMAP リソースが PIPELINE スキャン・コマンドを使用して自動的に作成される際に、URIMAP リソースの TRANSACTION 属性を定義するために使用されます。

許容文字:

A - Z、a - z、0 - 9、\$、@、#、_、<、>

URI=*value*

サービス・プロバイダーでは、このパラメーターはクライアントが Web サービスのアクセスに使用する相対 URI を指定します。CICS は、DFHWS2LS によって作成された Web サービス・バインディング・ファイルから URIMAP リソースを生成するときに指定された値を使用します。このパラメーターは URIMAP 定義が適用される URI のパスのコンポーネントを指定します。

サービス・リクエスターでは、このパラメーターではターゲット Web サービスの URI は指定されません。Web サービス記述に `wsdl:port` から `soap:address` ロケーションが指定されていれば、これが使用されます。ただし、EXEC CICS INVOKE SERVICE コマンドの URI オプションを使用して指定変更することができます。

USERID=*id*

サービス・プロバイダーで、このパラメーターは、任意の Web クライアントによって使用できる 1 から 8 文字のユーザー ID を指定します。アプリケーション生成の応答または Web サービスでは、別名トランザクションがこのユーザー ID の下に付加されます。このパラメーターの値は、URIMAP リソースが PIPELINE スキャン・コマンドを使用して自動的に作成される際に、URIMAP リソースの USERID 属性を定義するために使用されます。

許容文字:

A - Z、a - z、0 - 9、\$、@、#

WSADDR-EPR-ANY=TRUE|FALSE

CICS で 1 つの WS-Addressing エンドポイント参照 (EPR) を言語構造のコンポーネント部分に変換するか、または EPR を 1 つの `<xsd:any>` タイプとして扱うかを指定します。EPR を 1 つの `<xsd:any>` として扱う場合、WSACONTEXT BUILD API は EPR XML を直接使用できます。

FALSE

DFHWS2LS は通常どおりに動作し、XML を高水準言語構造に変換します。

TRUE このオプションを TRUE に設定すると、ランタイム CICS は EPR 全体を `<xsd:any>` タイプとして扱い、EPR XML はアプリケーションが参照できるコンテナに入れられます。アプリケーションは EPR XML を WSACONTEXT BUILD API と共に使用して、アドレッシング・コンテキストで EPR を構成できます。

このパラメーターは実行時レベル 3.0 以降でのみ使用できます。

WSBIND=*value*

Web サービス・バインディング・ファイルの完全修飾 z/OS UNIX 名です。
DFHWS2LS は、このファイルが存在しない場合、ディレクトリー構造は作成しませんがファイルを作成します。デフォルトのファイル拡張子は .wsbind です。

WSDL=*value*

Web サービス記述を格納するファイルの完全修飾 z/OS UNIX 名です。WSRR を使って WSDL 文書を取り出す場合、このパラメーターは WSDL 文書のローカル・コピーの書き込み先となるファイル・システム上の場所を指定します。

WSDL-SERVICE=*value*

Web サービス記述に Binding エlement について複数の Service Element が含まれるときに使用する、wsdl:Service Element を指定します。**BINDING** パラメーターの値を指定する場合は、このパラメーターに指定する Service Element が、指定された Binding Element と整合している必要があります。このパラメーターは WSDL 1.1 文書または WSDL 2.0 文書のいずれかで使用できます。

WSRR-NAME=*value*

WSRR から取り出す WSDL 文書の名前を指定します。このパラメーターは、**WSRR-SERVER** パラメーターが指定された場合にのみ使用します。

WSRR-NAMESPACE=*value*

WSRR から取り出す WSDL 文書の名前空間を指定します。**WSRR-SERVER** パラメーターが指定されている場合、オプションでこのパラメーターを使用して、**WSRR-NAME** パラメーターに示される WSDL 文書名を完全修飾することができます。

WSRR-PASSWORD=*value*

WSRR へのアクセスにパスワードの入力が必要な場合は、このオプション・パラメーターを使用します。

WSRR-USERNAME パラメーターを指定する場合には、このパラメーターも指定する必要があります。

このパラメーターは、**WSRR-SERVER** パラメーターが指定された場合にのみ使用します。

WSRR-SERVER=*{domain name:port number}/{IP address:port number}*

このパラメーターを使用して、IBM WebSphere Service Registry and Repository (WSRR) サーバーの場所を指定します。このパラメーターを指定した場合、WSRR パラメーター検証が使用されます。

WSRR-USERNAME=*value*

WSRR へのアクセスでユーザー名を指定する必要がある場合は、このオプション・パラメーターを使用します。**WSRR** はこのユーザー名を使って所有者プロパティを設定します。

このパラメーターは、**WSRR-SERVER** パラメーターが指定された場合にのみ使用します。

WSRR-VERSION=value

WSRR から取り出す WSDL 文書のバージョンを指定します。このパラメーターは、**WSRR-SERVER** パラメーターが指定されている場合に限り使用できます。

XML-ONLY=TRUE/FALSE

CICS で SOAP メッセージ内の XML をアプリケーション・データに変換するかどうかを指定します。XML 自体を処理する Web サービス・アプリケーションを作成するには、**XML-ONLY** パラメーターを使用してください。

TRUE CICS は XML への変換をまったく実行しません。サービス・リクエスターまたはプロバイダー・アプリケーションは、DFHWS-BODY コンテナーの内容を処理して XML と高水準言語の間でデータを直接マップする必要があります。

FALSE

CICS は XML を高水準言語に変換します。

このパラメーターは実行時レベル 2.1 以降でのみ使用できます。

その他の情報

- DFHLS2SC を実行するユーザー ID は、UNIX システム・サービスを使用するように構成されていなければなりません。このユーザー ID には、CICS z/OS UNIX ファイル構造および PDS ライブラリーに対する読み取り権限と **LOGFILE**、**WSBIND**、および **WSDL** パラメーターに指定されたディレクトリーに対する書き込み権限が必要です。
- Java を実行するため、このユーザー ID には十分な大きさのストレージを割り振る必要があります。

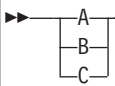
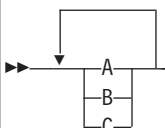
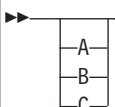
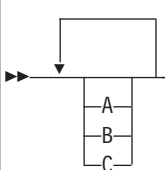
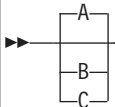
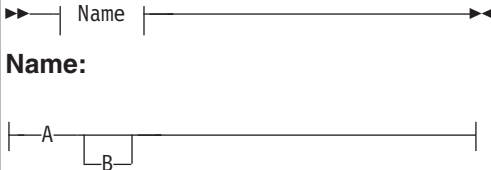
例

```
//WS2LS JOB 'accounting information',name,MSGCLASS=A
// SET QT=''
//JAVAPROG EXEC DFHWS2LS,
// TMPFILE=&QT.&SYSUID.&QT
//INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
REQMEM=CPYBK1
RESPMEM=CPYBK2
LANG=COBOL
LOGFILE=/u/exampleapp/wsbinding/example.log
MAPPING-LEVEL=3.0
CHAR-VARYING=NULL
INLINE-MAXOCCURS-LIMIT=2
PGMNAME=DFH0XCMN
URI=exampleApp/example
PGMINT=COMMAREA
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbinding/example.wsbinding
WSDL=/u/exampleapp/wsd1/example.wsd1
/*
```

構文表記

構文表記は、CICS コマンド、リソース定義、およびその他の多くの対象に対して指定できるオプションまたは属性の許容される組み合わせを指定します。

構文表記で使用する規則には、以下のものがあります。

表記	説明
	必須の選択肢のセットを表します。示された値のいずれか (1 つだけ) を指定する必要があります。
	必須の選択肢のセットを表します。示された値の少なくとも 1 つを指定する必要があります。それらを任意の順序で複数指定することができます。
	オプションの選択肢のセットを示します。値を指定しないか、示された値を 1 つ指定できます。
	オプションの選択肢のセットを示します。値を指定しないか、示された 1 つ以上の値を任意の順序で指定できます。
	オプションの選択肢のセットを示します。値を指定しないか、示された値を 1 つ指定できます。A は、何も指定しない場合に使用されるデフォルト値です。
	構文表記の指定されたセクションの参照。
	A= は、示されたとおりに入力する必要がある文字を示します。 <i>value</i> は、該当する値を指定する必要がある変数を示します。

CICS アシスタントのマッピング・レベル

マッピングは、言語構造と XML スキーマの間で情報が変換される方法を指定するのに使用される一連の規則です。最も高度なマッピングを活用するには、CICS アシスタントの **MAPPING-LEVEL** パラメーターを最新レベルに設定することをお勧めします。

マッピングの各レベルは、前のマッピングの機能を継承します。マッピングのレベルが一番高いと、一番優れた機能を利用できます。一番高いマッピング・レベルでは、実行時のデータ変換をさらに制御することができ、特定のデータ・タイプおよび XML エlement に対するサポートの制限も解除されます。

以前のレベルで有効だったアプリケーションを再展開したい場合には、**MAPPING-LEVEL** パラメーターをそのレベルに設定できます。

マッピング・レベル 3.0

マッピング・レベル 3.0 は CICS TS 4.1 領域と互換性があります。

このマッピング・レベルでは次に示すサポートが提供されます。

- DFHSC2LS および DFHWS2LS は xsd:dateTime データ・タイプを CICS ASKTIME フォーマットへマップする。
- DFHLS2WS は、1 つのコンテナではなく複数のコンテナを使用するアプリケーションから WSDL 文書および Web サービス・バインディングを生成できます。
- 固定長データ構造によって記述されているデータの切り捨ての許容。この動作は、CICS アシスタントの **DATA-TRUNCATION** パラメーターを使用して設定できます。

マッピング・レベル 2.2 以上

マッピング・レベル 2.2 は、APAR PK69738 およびそれ以上が適用された CICS TS 3.2 領域と互換性があります。

マッピング・レベル 2.2 以上では、DFHSC2LS および DFHWS2LS は次の XML マッピングをサポートします。

- Element の固定値
- 置換グループ
- 抽象データ・タイプ
- XML スキーマ <sequence> Element を <choice> Element 内にネストできる

DFHSC2LS および DFHWS2LS は次に示す XML マッピングに対する拡張サポートを提供します。

- 抽象Element
- XML スキーマ <choice> Element

マッピング・レベル 2.1 以上の場合

マッピング・レベル 2.1 は、APAR PK59794 およびそれ以上が適用された CICS TS 3.2 領域と互換性があります。

このマッピング・レベルでは、新規の **INLINE-MAXOCCURS-LIMIT** パラメーターによって、および **CHAR-VARYING** パラメーターの新規の値によって可変コンテンツを処理する方法をさらに制御できます。

マッピング・レベル 2.1 以上では、DFHSC2LS および DFHWS2LS は XML マッピングについて以下に示す新規および改良されたサポートを提供します。

- XML スキーマ <any> エレメント
- xsd:anyType タイプ
- 抽象エレメントの許容
- **INLINE-MAXOCCURS-LIMIT** パラメーター
- minOccurs 属性

INLINE-MAXOCCURS-LIMIT パラメーターは、可変の繰り返しリストがインラインにマップされるかどうかを指定します。可変の繰り返しコンテンツをインラインでマップすることに関して詳しくは 214 ページの『変化するエレメントの配列』を参照してください。

XML スキーマの <sequence>、<choice>、および <all> エレメントでの minOccurs 属性のサポートが拡張されています。minOccurs="0" の場合、CICS アシスタントはこれらのエレメントを、minOccurs="0" 属性がそのすべての子エレメントの属性であるかのように扱います。

マッピング・レベル 2.1 以上では、DFHLS2SC および DFHLS2WS は次の XML マッピングをサポートします。

- COBOL および PL/I での FILLER フィールドは無視されます
- **CHAR-VARYING** パラメーターに対する COLLAPSE 値
- **CHAR-VARYING** パラメーターに対する BINARY 値

COBOL および PL/I での FILLER フィールドは無視されます。生成される XML スキーマにはこれらが表示されず、相応のギャップが実行時にデータ構造に残されます。

COLLAPSE により、CICS はテキスト・フィールドの末尾スペースを無視するようになります。

BINARY は 2 進数フィールドのサポートを提供します。この値は、COBOL を XML スキーマに変換する際に役立ちます。このオプションは SBCS 文字配列でのみ使用可能です。配列が xsd:string フィールドではなく、固定長の xsd:base64Binary フィールドにマップされるようになります。

マッピング・レベル 1.2 以上の場合

マッピング・レベル 1.2 は CICS TS 3.1 領域およびそれ以上と互換性があります。

以下に示すバッチ・ツールの追加パラメーターを使用して、実行時の文字データおよびバイナリー・データの変換方法をさらに制御できます。

- **CHAR-VARYING**
- **CHAR-VARYING-LIMIT**
- **CHAR-MULTIPLIER**
- **DEFAULT-CHAR-MAXLENGTH**

DFHSC2LS または DFHWS2LS で **CHAR-MULTIPLIER** パラメーターを使用する場合は、このパラメーターの値を使用して文字データに必要なスペースの量を計算した後で、以下の規則が適用されることに注意してください。

- DFHSC2LS および DFHWS2LS は次のマッピングを提供します。
 - 最大長が 32 767 バイトより大きい可変長の文字データ・タイプはコンテナーにマップされます。 下限は **CHAR-VARYING-LIMIT** パラメーターを使用して設定できます。コンテナーの名前を格納するために、16 バイトのフィールドが言語構造に作成されます。 実行時に、文字データはコンテナーに格納され、コンテナー名は言語構造に入ります。
 - 最大長が 32 768 バイトより小さい可変長の文字データ・タイプは、C/C++ および Enterprise PL/I を除くすべての言語で VARYING 構造にマップされます。C/C++ ではこれらのデータはヌル終了ストリングにマップされ、Enterprise PL/I ではこれらのデータ・タイプは VARYINGZ 構造にマップされます。**CHAR-VARYING** パラメーターを使用して、可変長の文字データがマップされる方法を選択できます。
 - 最大長が 32 768 バイトより小さい可変長のバイナリー・データは、すべての言語で VARYING 構造にマップされます。最大長が 32 768 バイト以上である場合、データはコンテナーにマップされます。 コンテナーの名前を格納するために、16 バイトのフィールドが言語構造に作成されます。 実行時に、バイナリー・データはコンテナーに格納され、コンテナー名は言語構造に入ります。

XML スキーマの文字データ・タイプに、関連付けられた長さがいない場合は、DFHWS2LS または DFHSC2LS で **DEFAULT-CHAR-MAXLENGTH** パラメーターを使用してデフォルトの長さを割り当てることができます。

DFHLS2SC および DFHLS2WS は次のマッピングを提供します。

- 文字フィールドは `xsd:string` データ・タイプにマップされ、実行時に固定長フィールドまたはヌル終了ストリングとして処理できます。 PL/I を除くすべての言語での、実行時の可変長文字データの処理方法を、**CHAR-VARYING** パラメーターを使用して選択できます。
- Base64Binary データ・タイプはデータの最大長が 32 767 バイトより大きいか、長さが定義されていない場合に、コンテナーにマップされます。 データの長さが 32 767 以下である場合は、base64Binary データ・タイプがすべての言語で VARYING 構造にマップされます。

マッピング・レベル 1.1 以上の場合

マッピング・レベル 1.1 は CICS TS 3.1 領域およびそれ以上と互換性があります。

このマッピング・レベルでは XML 文字およびバイナリー・データ・タイプのマッピング、特に XML スキーマで異なる値で定義された `maxLength` および `minLength` 属性のある可変長のデータのマッピングが改良されています。 データは次のように処理されます。

- 16 MB より大きい固定長を持つ文字およびバイナリー・データ・タイプが、PL/I を除くすべての言語でコンテナーにマップされます。PL/I では、32 767 バイトより大きい固定長の文字およびバイナリー・データ・タイプがコンテナーにマップされます。 コンテナーの名前を格納するために、16 バイトのフィールドが言語構造に作成されます。 実行時に、固定長データはコンテナーに格納され、コンテナー名は言語構造に入ります。

コンテナの長さは可変なので、XML スキーマまたは Web サービス記述に指定された固定長に合わせるために、コンテナにマップされる固定長データが、スペースやヌルで埋め込まれたり、切り捨てられたりすることはありません。データの長さが重要である場合は、長さをチェックするようにアプリケーションを作成するか、CICS 領域で検証をオンにすることができます。SOAP および XML 検証はどちらもパフォーマンスに大きく影響します。

- XML スキーマ `<list>` および `<union>` データ・タイプは文字フィールドにマップされます。
- スキーマ定義の XML 属性は無視されるのではなく、マップされます。最大 255 の属性が各 XML エLEMENT で許可されます。詳しくは、219 ページの『XML 属性のサポート』を参照してください。
- `xsi:nil` 属性がサポートされます。詳しくは、219 ページの『XML 属性のサポート』を参照してください。

マッピング・レベル 1.1 のみ

マッピング・レベル 1.1 は CICS TS 3.1 領域およびそれ以上と互換性があります。

このマッピング・レベルでは XML 文字およびバイナリー・データ・タイプのマッピング、特に XML スキーマで異なる値で定義された `maxLength` および `minLength` 属性のある可変長のデータのマッピングが改良されています。データは次のように処理されます。

- 可変長のバイナリー・データ・タイプがコンテナにマップされます。コンテナの名前を格納するために、16 バイトのフィールドが言語構造に作成されます。実行時に、バイナリー・データはコンテナに格納され、コンテナ名は言語構造に入ります。
- 最大長が 32 767 バイトより大きい可変長の文字データ・タイプはコンテナにマップされます。コンテナの名前を格納するために、16 バイトのフィールドが言語構造に作成されます。実行時に、文字データはコンテナに格納され、コンテナ名は言語構造に入ります。
- 16 MB より小さい固定長を持つ文字およびバイナリー・データ・タイプが、PL/I を除くすべての言語で固定長フィールドにマップされます。PL/I では、32 767 バイト以下の固定長の文字およびバイナリー・データ・タイプが固定長フィールドにマップされます。
- CICS はデータを `hexBinary` フォーマットでエンコードおよびデコードしますが、`base64Binary` フォーマットでは行いません。XML スキーマの `Base64Binary` データ・タイプは言語構造内のフィールドにマップされます。フィールドのサイズは次の公式を使用して計算されます。 $4 \times (\text{ceil}(z/3))$ 。ここで、
 - z は XML スキーマのデータ・タイプの長さです。
 - $\text{ceil}(x)$ は x 以上の最小の整数です。

z の長さが 24 566 バイトより大きい場合、結果として生成される言語構造のコンパイルは失敗します。24 566 バイトより大きい `base64Binary` データがある場合は、マッピング・レベル 1.2 の使用をお勧めします。マッピング・レベル 1.2 では、言語構造内のフィールドを使用する代わりに、`base64Binary` データをコンテナにマップできます。

マッピング・レベル 1.0 のみ

マッピング・レベル 1.0 は CICS TS 3.1 領域およびそれ以上と互換性があります。

次の制限事項 (以後のマッピング・レベルでは変更されている) に注意してください。

- DFHSC2LS および DFHWS2LS は XML スキーマの文字およびバイナリー・データ・タイプを、言語構造内の固定長フィールドにマップします。次の部分的な XML スキーマをご覧ください。

```
<xsd:element name="example">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="33000"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

この XML スキーマの一部分は、COBOL 言語構造で次のように表示されます。

```
15 example      PIC X(33000)
```

- CICS はデータを hexBinary フォーマットでエンコードおよびデコードしますが、base64Binary フォーマットでは行いません。DFHSC2LS および DFHWS2LS は Base64Binary データを固定長文字フィールドにマップします。この内容は、アプリケーション・プログラムによってエンコードまたはデコードされる必要があります。
- DFHSC2LS および DFHWS2LS は処理中に XML 属性を無視します。
- DFHLS2SC および DFHLS2WS は、言語構造内の文字フィールドおよびバイナリー・フィールドを固定長フィールドとして解釈し、これらのフィールドを、maxLength 属性を持つ XML エレメントにマップします。十分なデータがない場合には、実行時に言語構造内のフィールドがスペースやヌルで埋められます。

高水準言語と XML のスキーマ・マッピング

CICS アシスタントを使用して高水準言語構造と XML スキーマまたは WSDL ドキュメントとのマッピングを生成します。さらに、CICS アシスタントは、高水準言語構造から XML スキーマまたは WSDL 文書を (またはその逆を) 生成します。

ユーティリティー・プログラム DFHSC2LS と DFHLS2SC を合わせて CICS XML アシスタントといいます。ユーティリティー・プログラム DFHWS2LS と DFHLS2WS を合わせて CICS Web サービス・アシスタントといいます。

- DFHLS2SC および DFHLS2WS は高水準言語構造を XML スキーマと WSDL 文書にそれぞれマップします。
- DFHSC2LS および DFHWS2LS は XML スキーマと WSDL 文書を高水準言語構造にマップします。

以下に示すように、2 つのマッピングは対称的ではありません。

- DFHLS2SC または DFHLS2WS を使って言語データ構造を処理した結果として生成される XML スキーマまたは WSDL 文書を、相補的なユーティリティー・プログラム (それぞれ DFHSC2LS または DFHWS2LS) を使って処理する場合、最

最終的なデータ構造が最初のデータ構造と同じになると期待することはできません。ただし、最終的なデータ構造は、最初のデータ構造と論理的には同等です。

- DFHSC2LS または DFHWS2LS を使って XML スキーマまたは WSDL 文書処理した結果として生成される言語構造を、相補的なユーティリティー・プログラム (それぞれ DFHLS2SC または DFHLS2WS) を使って処理する場合、最終的な XML スキーマまたは WSDL 文書の中の XML スキーマが最初のものと同じになることは期待できません。
- 場合によっては、DFHLS2SC および DFHLS2WS でサポートされない言語構造が DFHSC2LS および DFHWS2LS によって生成されることがあります。

DFHLS2SC および DFHLS2WS で処理される言語構造は、CICS がサポートする言語コンパイラで実装されるその言語の規則に従って正しくコーディングする必要があります。

CICS アシスタントのデータ・マッピングの制限

CICS は高水準言語構造と XML スキーマ、または高水準言語と WSDL バージョン 1.1 もしくは 2.0 に準拠する WSDL 文書との間の双方向のデータ・マッピングを制限付きでサポートしています。これらの制限は DFHWS2LS および DFHSC2LS ツールにのみ適用され、マッピング・レベルによって異なります。

全マッピング・レベルでの制限

- リテラル・エンコードを使用する SOAP バインディングのみがサポートされます。そのため、`use` 属性を `literal` の値に設定する必要があります。
`use="encoded"` はサポートされません。
- データ・タイプ定義を、XML スキーマ定義言語 (XSD) を使用してエンコードする必要があります。スキーマ内では、SOAP メッセージで使用されるデータ・タイプを明示的に宣言する必要があります。
- Web サービス記述内の一部のキーワードの長さには制限があります。例えば、操作名、バインディング名、およびパート名の長さは、255 文字に制限されます。場合によっては、操作名の最大長がこれより多少短い場合があります。
- Web サービス記述で定義された SOAP 障害はすべて無視されます。サービス・プロバイダー・アプリケーションで SOAP 障害メッセージを送信するには、`EXEC CICS SOAPFAULT` コマンドを使用します。
- DFHWS2LS および DFHSC2LS が、特定のスコープでサポートする `<xsd:any>` エレメントは 1 つまでです。例えば、次のスキーマ・フラグメントはサポートされません。

```
<xsd:sequence>
  <xsd:any/>
  <xsd:any/>
</xsd:sequence>
```

ここで、必要に応じて `<xsd:any>` を使用して `minOccurs` および `maxOccurs` を指定できます。例えば、次のスキーマ・フラグメントはサポートされます。

```
<xsd:sequence>
  <xsd:any minOccurs="2" maxOccurs="2"/>
</xsd:sequence>
```

- 循環参照はサポートされません。例えば、型 A に型 B が含まれ、さらには型 B に型 A が含まれる場合などです。

- グループ・エレメント (<xsd:choice>、<xsd:sequence>、<xsd:group>、または <xsd:all> エレメント) などにおける繰り返しはサポートされません。例えば、次のスキーマ・フラグメントはサポートされません。

```
<xsd:choice maxOccurs="2">
  <xsd:element name="name1" type="string"/>
</xsd:choice>
```

例外として、マッピング・レベル 2.1 以上では、これらのエレメントに対して maxOccurs="1" および minOccurs="0" がサポートされます。

- DFHSC2LS および DFHWS2LS は、xsi:type 属性または置換グループの XML スキーマで宣言されているデータ・タイプおよびエレメントから派生した、SOAP メッセージ内のデータ・タイプおよびエレメントをサポートしません。マッピング・レベルが 2.2 以上で、親エレメントまたは親タイプが抽象であると定義されている場合を除きます。
- <xsd:choice> エレメントに組み込まれた <xsd:sequence> エレメントおよび <xsd:group> エレメントは、マッピング・レベル 2.2 より前のレベルではサポートされません。<xsd:choice> エレメントに組み込まれた <xsd:choice> エレメントおよび <xsd:all> エレメントは一切サポートされません。

マッピング・レベル 1.1 以上におけるサポートの向上

マッピング・レベルが 1.1 以上である場合、DFHWS2LS は次の XML エレメントおよびエレメント属性をサポートします。

- <xsd:list> エレメント。
- <xsd:union> エレメント。
- xsd:anySimpleType タイプ。
- <xsd:attribute> エレメント。マッピング・レベル 1.0 ではこのエレメントは無視されます。

マッピング・レベル 2.1 以上におけるサポートの向上

マッピング・レベルが 2.1 以上である場合、DFHWS2LS は次の XML エレメントおよびエレメント・タイプをサポートします。

- <xsd:any> エレメント。
- xsd:anyType タイプ。
- 抽象エレメント。それより前のマッピング・レベルでは、抽象エレメントは継承の階層の非端末型としてのみサポートされていました。
- <xsd:all>、<xsd:choice>、および <xsd:sequence> エレメントにおける maxOccurs および minOccurs 属性。maxOccurs="1" および minOccurs="0" の場合のみ。
- COBOL における "FILLER" フィールドおよび PL/I における "*" フィールドは抑制されます。フィールドは生成された WSDL に表示されず、相応のギャップが実行時にデータ構造に残されます。

マッピング・レベル 2.2 以上におけるサポートの向上

マッピング・レベルが 2.2 以上の場合、DFHSC2LS および DFHWS2LS では <xsd:choice> エレメントで最大 255 までのオプションをサポートするようになり、

<xsd:choice> エレメントのサポートが向上しています。 <xsd:choice> サポートについて詳しくは、223 ページの『<xsd:choice>のサポート』を参照してください。

マッピング・レベル 2.2 以上では、CICS アシスタントが次の XML マッピングをサポートします。

- 置換グループ
- エレメントの固定値
- 抽象データ・タイプ

<xsd:choice> エレメントに組み込まれた <xsd:sequence> エレメントおよび <xsd:group> エレメントはマッピング・レベル 2.2 以上でサポートされています。例えば、次のスキーマ・フラグメントはサポートされます。

```
<xsd:choice>
  <xsd:element name="name1" type="string"/>
  <xsd:sequence/>
</xsd:choice>
```

SOAP メッセージの親エレメントまたは親タイプが抽象と定義されている場合、DFHSC2LS および DFHWS2LS は XML スキーマの宣言されたデータ・タイプおよびエレメントから派生したデータ・タイプおよびエレメントをサポートします。

マッピング・レベル 3.0 以上におけるサポートの向上

マッピング・レベルが 3.0 以上の場合、CICS アシスタントが次のマッピングの向上をサポートします。

- DFHSC2LS および DFHWS2LS は xsd:dateTime データ・タイプを CICS ASKTIME フォーマットへマップする。
- DFHLS2WS は、1 つのコンテナではなく複数のコンテナを使用するアプリケーションから WSDL 文書および Web サービス・バインディングを生成できます。
- 固定長データ構造によって記述されているデータの切り捨ての許容。この動作は、CICS アシスタントの **DATA-TRUNCATION** パラメーターを使用して設定できます。

COBOL から XML スキーマへのマッピング

DFHLS2SC および DFHLS2WS ユーティリティー・プログラムは COBOL データ構造と XML スキーマ定義との間のマッピングをサポートします。

COBOL の名前は、次の規則に従って XML の名前に変換されます。

1. 重複した名前は、1 つ以上の数字を追加することによって固有にします。

例えば、year の 2 つのインスタンスは year と year1 になります。

2. ハイフンは、下線文字に置き換えられます。一連の連続するハイフンは、連続する下線で置き換えられます。

例えば、current-user--id は current_user_id になります。

3. ハイフンで区切られており、大文字のみを含む名前のセグメントは、小文字に変換されます。

例えば、CA-REQUEST-ID は ca_request_id になります。

4. 数字で始まる名前には、先頭に下線文字が追加されます。

例えば、9A-REQUEST-ID は _9a_request_id になります。

CICS は、次の表に従って COBOL データ記述エレメントをスキーマ・エレメントにマップします。この表にない COBOL データ記述エレメントは、DFHLS2SC または DFHLS2WS ではサポートされません。次の制約事項も適用されます。

- レベル番号が 66 および 77 のデータ記述項目はサポートされていません。レベル番号が 88 のデータ記述項目は無視されます。
- データ記述記入項目の次の文節はサポートされていません。

OCCURS DEPENDING ON

OCCURS INDEXED BY

REDEFINES

RENAMES (レベル 66)

DATE FORMAT

- データ記述項目の次の文節は無視されます。

BLANK WHEN ZERO

JUSTIFIED

VALUE

- SIGN 文節 SIGN TRAILING はサポートされます。SIGN 文節 SIGN LEADING は、DFHLS2SC または DFHLS2WS で指定されたマッピング・レベルが 1.2 以上の場合のみサポートされます。
- SIGN TRAILING 文節と SIGN LEADING 文節の両方では、SEPARATE CHARACTER はマッピング・レベル 1.2 以上でサポートされます。
- USAGE 文節の次の句はサポートされていません。

OBJECT REFERENCE

POINTER

FUNCTION-POINTER

PROCEDURE-POINTER

- USAGE 文節の次の句は、マッピング・レベル 1.2 以上でサポートされます。

COMPUTATIONAL-1

COMPUTATIONAL-2

- DISPLAY および COMPUTATIONAL-5 データ記述項目でサポートされる唯一の PICTURE 文字は 9、S、および Z です。
- PACKED-DECIMAL データ記述項目でサポートされる PICTURE 文字は、9、S、V、および Z です。
- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを NULL に設定すると、文字配列は xsd:string にマップされ、ヌル終了ストリングとして処理されます。
- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを BINARY に設定すると、文字配列は xsd:base64Binary にマップされ、バイナリー・データとして処理されます。

- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを **COLLAPSE** に設定すると、ストリングの末尾の空白文字は無視されます。

COBOL のデータ記述	スキーマの <code>simpleType</code>
PIC X(<i>n</i>) PIC A(<i>n</i>) PIC G(<i>n</i>) DISPLAY-1 PIC N(<i>n</i>)	<pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="<i>n</i>"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> ここで $m=n$
PIC S9 DISPLAY PIC S99 DISPLAY PIC S999 DISPLAY PIC S9999 DISPLAY	<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> ここで、n は、文字「9」のパターンによって表現できる最大値です。
PIC S9(<i>z</i>) DISPLAY ここで $5 \leq z \leq 9$ です	<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> ここで、n は、文字「9」のパターンによって表現できる最大値です。
PIC S9(<i>z</i>) DISPLAY ($9 < z$)	<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> <xsd:minInclusive value="-<i>n</i>"/> <xsd:maxInclusive value="<i>n</i>"/> </xsd:restriction> </xsd:simpleType></pre> ここで、n は、文字「9」のパターンによって表現できる最大値です。
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="<i>n</i>"/> </xsd:restriction> </xsd:simpleType></pre> ここで、n は、文字「9」のパターンによって表現できる最大値です。
PIC 9(<i>z</i>) DISPLAY ここで $5 \leq z \leq 9$ です	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="<i>n</i>"/> </xsd:restriction> </xsd:simpleType></pre> ここで、n は、文字「9」のパターンによって表現できる最大値です。

COBOL のデータ記述	スキーマの simpleType
PIC 9(z) DISPLAY (9 < z)	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="n"/> </xsd:restriction> </xsd:simpleType></pre> ここで、n は、文字「9」のパターンによって表現できる最大値です。
PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY ここで $n \leq 4$ です。	<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>
PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY ここで、 $5 \leq n \leq 9$ です。	<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>
PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY ここで $9 < n$	<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>
PIC 9(n) COMP PIC 9(n) COMP-4 PIC 9(n) COMP-5 PIC 9(n) BINARY ここで $n \leq 4$ です。	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>
PIC 9(n) COMP PIC 9(n) COMP-4 PIC 9(n) COMP-5 PIC 9(n) BINARY ここで、 $5 \leq n \leq 9$ です。	<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 ここで $9 < n$	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>
PIC S9(m)V9(n) COMP-3	<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="p"/> <xsd:fractionDigits value="n"/> </xsd:restriction> </xsd:simpleType></pre> ここで $p = m + n$

COBOL のデータ記述	スキーマの simpleType
PIC 9(m)V9(n) COMP-3	<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="p"/> <xsd:fractionDigits value="n"/> <xsd:minInclusive value="0"/> </xsd:restriction> </xsd:simpleType></pre> ここで $p = m + n$
PIC S9(m) COMP-3 DATETIME=PACKED15 の場合マッピング・レベル 3.0 でサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre> タイム・スタンプのフォーマットは CICS ABSTIME です。
PIC S9(m)V9(n) DISPLAY マッピング・レベル 1.2 以上でサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="p"/> <xsd:fractionDigits value="n"/> </xsd:restriction> </xsd:simpleType></pre> ここで $p = m + n$
COMP-1 マッピング・レベル 1.2 以上でサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
COMP-2 マッピング・レベル 1.2 以上でサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

XML スキーマから COBOL へのマッピング

DFHSC2LS および DFHWS2LS のユーティリティ・プログラムは、XML スキーマ定義および COBOL データ構造との間のマッピングをサポートします。

CICS アシスタントは、次の規則を使用してスキーマ・エレメント名から COBOL 変数の固有で有効な名前を生成します。

1. COBOL 予約語には接頭部「X」が付きます。

例えば、DISPLAY は XDISPLAY になります。

2. A から Z、a から z、0 から 9、またはハイフン以外の文字は、「X」で置き換えられます。

例えば、monthly_total は monthlyXtotal になります。

3. 最後の文字がハイフンである場合は、「X」で置き換えられます。

例えば、ca-request- は ca-requestX になります。

4. スキーマで変数が変化する基数を持つように指定する場合 (xsd:element で minOccurs および maxOccurs を異なる値で指定する場合)、スキーマ・エレメント名が 23 文字を超えると、この長さに切り捨てられます。

スキーマで変数が固定の基数を持つように指定する場合、スキーマ・エレメント名が 28 文字を超えると、この長さに切り捨てられます。

5. 同じスコープ内の重複した名前は、名前の 2 番目以降のインスタンスに 1 つまたは 2 つの数字を追加することによって固有にします。

例えば、year の 3 つのインスタンスは year、year1、および year2 になります。

6. スキーマで変数が変化する基数を持つように指定する場合 (minOccurs および maxOccurs を異なる値で指定する場合) に使用される文字列 -cont または -num 用に、5 文字が予約されます。

詳しくは、214 ページの『変化するエレメントの配列』を参照してください。

7. 属性では、前の規則がエレメント名に適用されます。接頭部 attr- がエレメント名に追加され、この後に -value または -exist が付きます。全長が 28 文字を超える場合、エレメント名は切り捨てられます。詳しくは、219 ページの『XML 属性のサポート』を参照してください。

nillable 属性には特別な規則があります。接頭部 attr- が追加されますが、エレメント名の先頭には nil- も追加されます。エレメント名の後には -value が付きます。全長が 28 文字を超える場合、エレメント名は切り捨てられます。

結果として生成される名前の全長は、30 文字以下になります。

DFHSC2LS および DFHWS2LS は、指定されたマッピング・レベルを使用して、スキーマ・タイプを次の表に従って、COBOL のデータ記述エレメントにマップします。次の点に注意してください。

- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを NULL に設定すると、可変長文字データはヌル終了文字列にマップされ、ヌル終止符として追加された 1 つの文字が割り振られます。
- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを YES に設定すると、可変長文字データは 2 つの関連エレメント (長さフィールドとデータ・フィールド) にマップされます。例を次に示します。

```
<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"/>
```

これは、次にマップします。

```
15 textString-length PIC S9999 COMP-5 SYNC
15 textString        PIC X(10000)
```

スキーマの単純型	COBOL のデータ記述
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 2.0 以下の場合 サポートされていない マッピング・レベル 2.1 の場合 サポートされる

スキーマの単純型	COBOL のデータ記述
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.0 の場合 サポートされていない マッピング・レベル 1.1 以上の場合 PIC X(255)
<pre><xsd:simpleType> <xsd:restriction base="xsd:type" <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> ここで、<i>type</i> は以下のいずれかです。 - string - normalizedString - token - Name - NMTOKEN - language - NCName - ID - IDREF - ENTITY - hexBinary	すべてのマッピング・レベル PIC X(z)
<pre><xsd:simpleType> <xsd:restriction base="xsd:type" </xsd:restriction> </xsd:simpleType></pre> ここで、<i>type</i> は以下のいずれかです。 - duration - date - time - gDay - gMonth - gYear - gMonthDay - gYearMonth	すべてのマッピング・レベル PIC X(32)
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.2 以下の場合 PIC X(32) マッピング・レベル 2.0 以上の場合 PIC X(40) マッピング・レベル 3.0 以上の場合 PIC S9(15) COMP-3 フォーマットは CICS ABSTIME です。

スキーマの単純型	COBOL のデータ記述
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> </xsd:restriction> </xsd:simpleType></pre> ここで、<i>type</i> は以下のいずれかです。 <pre>byte unsignedByte</pre>	すべてのマッピング・レベル PIC X DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル PIC S9999 COMP-5 SYNC または PIC S9999 DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル PIC 9999 COMP-5 SYNC または PIC 9999 DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル PIC S9(18) COMP-3
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル PIC S9(9) COMP-5 SYNC または PIC S9(9) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル PIC 9(9) COMP-5 SYNC または PIC 9(9) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル PIC S9(18) COMP-5 SYNC または PIC S9(18) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル PIC 9(18) COMP-5 SYNC または PIC 9(18) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="m"/> <xsd:fractionDigits value="n"/> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル PIC 9(p)V9(n) COMP-3 ここで $p = m - n$
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル PIC X DISPLAY

スキーマの単純型	COBOL のデータ記述
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	マッピング・レベル 1.0 の場合 サポートされていない マッピング・レベル 1.1 以上の場合 PIC X(255)
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	マッピング・レベル 1.0 の場合 サポートされていない マッピング・レベル 1.1 以上の場合 PIC X(255)
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> 長さは定義されていません。	マッピング・レベル 1.0 の場合 サポートされていない マッピング・レベル 1.1 の場合 PIC X(y) ここで $y = 4 \times (\text{ceil}(z/3))$。 $\text{ceil}(x)$ は x 以上の最小の整数です。 マッピング・レベル 1.2 以上の場合 PIC X(z) 固定長。 PIC X(16) 長さは定義されていません。このフィールドにはバイナリー・データを保管するコンテナの 16 バイトの名前が入ります。
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.1 以下の場合 PIC X(32) マッピング・レベル 1.2 以上の場合 COMP-1
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.1 以下の場合 PIC X(32) マッピング・レベル 1.2 以上の場合 COMP-2

表に示したスキーマ・タイプの一部は、minInclusive および maxInclusive ファセットに指定された値 (値がある場合) に応じて、COMP-5 SYNC または DISPLAY の COBOL 形式にマップします。

- 符号付き型 (short、int、および long) の場合、次のように指定するとき DISPLAY が使用されます。

```
<xsd:minInclusive value="-a"/>
<xsd:maxInclusive value="a"/>
```

ここで、*a* は 9 から成るストリングです。

- 符号なし型 (unsignedShort、unsignedInt、および unsignedLong) の場合は、次のように指定するとき DISPLAY が使用されます。

```
<xsd:minInclusive value="0"/>
<xsd:maxInclusive value="a"/>
```

ここで、*a* は 9 から成るストリングです。

この他の値を指定した場合、あるいは値を指定しなかった場合、COMP-5 SYNC が使用されます。

C および C++ から XML スキーマへのマッピング

DFHLS2SC および DFHLS2WS のユーティリティ・プログラムは C および C++ データ・タイプと XML スキーマ定義との間のマッピングをサポートします。

C および C++ の名前は、次の規則に従って XML の名前に変換されます。

1. XML エlement名で無効な文字は、「X」で置き換えられます。

例えば、monthly-total は monthlyXtotal になります。

2. 重複した名前は、1 つ以上の数字を追加することによって固有にします。

例えば、year の 2 つのインスタンスは year と year1 になります。

DFHLS2SC および DFHLS2WS は、次の表に従って、C および C++ のデータ・タイプをスキーマ・Elementにマップします。この表にない C および C++ のタイプは DFHLS2SC または DFHLS2WS ではサポートされません。次の制約事項も適用されます。

- ヘッダー・ファイルには、最上位の struct インスタンスが含まれていなければなりません。
- 自身をメンバーとして含む構造化タイプを宣言することはできません。
- 次の C および C++ のデータ・タイプはサポートされません。

```
decimal
long double
wchar_t (C++ のみ)
```

- 次のデータ・タイプは、ヘッダー・ファイル内に存在しても無視されます。

ストレージ・クラス指定子:

```
auto
register
static
extern
mutable
```

修飾子

```
const
volatile
_Export (C++ のみ)
_Packed (C のみ)
```

関数指定子

```
inline (C++ のみ)
```


virtual (C++ のみ)

初期値

- ヘッダー・ファイルには、以下の項目を指定できません。

共用体

クラス宣言

列挙型データ・タイプ

ポインター型変数

テンプレート宣言

定義済みマクロ (名前の先頭と末尾が 2 つの下線文字 (__) のマクロ)

行連結シーケンス (改行文字の直後にある ¥ 記号)

プロトタイプ関数宣言子

プリプロセッサ・ディレクティブ

ビット・フィールド

__cdecl (または _cdecl) キーワード (C++ のみ)

- アプリケーション・プログラマーは、32 ビットのコンパイラを使用して int が 4 バイトに確実にマップされるようにする必要があります。
- 次の C++ 予約済みキーワードはサポートされません。

explicit

using

namespace

typename

typeid

- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを NULL に設定すると、文字配列は xsd:string にマップされ、ヌル終了ストリングとして処理されます。
- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを BINARY に設定すると、文字配列は xsd:base64Binary にマップされ、バイナリー・データとして処理されます。
- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを COLLAPSE に設定すると、<xsd:whiteSpace value="collapse"/> がストリング用に生成されます。

C および C++ のデータ・タイプ	スキーマの simpleType
char[z]	<pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre>
char[8] DATETIME=PACKED15 の場合マッピング・レベル 3.0 以上でサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre> タイム・スタンプのフォーマットは CICS ABSTIME です。

C および C++ のデータ・タイプ	スキーマの simpleType
char	<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>
unsigned char	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>
short	<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>
unsigned short	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>
int long	<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>
unsigned int unsigned long	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>
long long	<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>
unsigned long long	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>
bool (C++ のみ)	<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>
float マッピング・レベル 1.2 以上でサポ ートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
double マッピング・レベル 1.2 以上でサポ ートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

XML スキーマから C および C++ へのマッピング

DFHSC2LS および DFHWS2LS のユーティリティー・プログラムは各 Web サービス記述に組み込まれている XML スキーマ定義と、C および C++ データ・タイプとの間のマッピングをサポートします。

CICS アシスタントは、次の規則を使用してスキーマ・エレメント名から C および C++ 変数の固有で有効な名前を生成します。

1. A から Z、a から z、0 から 9、または _ 以外の文字は、「X」で置き換えられます。

例えば、monthly-total は monthlyXtotal になります。

2. 先頭文字が英字ではない場合は、先頭文字が「X」に置換されます。

例えば、_monthlysummary は Xmonthlysummary になります。

3. スキーマ・エレメント名が 50 文字を超えた場合は、この長さに切り捨てられます。
4. 同じスコープ内の重複した名前は、1 つ以上の数字を追加することによって固有にします。

例えば、year の 2 つのインスタンスは year と year1 になります。

5. スキーマで変数が増える基数を持つように指定する場合 (xsd:element で minOccurs および maxOccurs を指定する場合) に使用されるストリング _cont または _num 用に、5 文字が予約されます。

詳しくは、214 ページの『変化するエレメントの配列』を参照してください。

6. 属性では、前の規則がエレメント名に適用されます。接頭部 attr_ がエレメント名に追加され、この後に _value または _exist が付きます。全長が 28 文字を超える場合、エレメント名は切り捨てられます。

nilable 属性には特別な規則があります。接頭部 attr_ が追加されますが、エレメント名の先頭には nil_ も追加されます。エレメント名の後には _value が付きます。全長が 28 文字を超える場合、エレメント名は切り捨てられます。

結果として生成される名前の全長は、57 文字以下になります。

DFHSC2LS および DFHWS2LS は、次の表に従って、スキーマ・タイプを C および C++ のデータ・タイプにマップします。次の規則も適用されます。

- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを NULL に設定すると、可変長文字データはヌル終了ストリングにマップされ、ヌル終止符として追加された 1 つの文字が割り振られます。
- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを YES に設定すると、可変長文字データは 2 つの関連エレメント (長さフィールドとデータ・フィールド) にマップされます。

スキーマの simpleType	C および C++ のデータ・タイプ
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 2.0 以下の場合 サポートされていない マッピング・レベル 2.1 以上の場合 サポートされる
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.0 の場合 サポートされていない マッピング・レベル 1.1 以上の場合 char[255]

スキーマの <code>simpleType</code>	C および C++ のデータ・タイプ
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> ここで、<code>type</code> は以下のいずれかです。 <code>string</code> <code>normalizedString</code> <code>token</code> <code>Name</code> <code>NMTOKEN</code> <code>language</code> <code>NCName</code> <code>ID</code> <code>IDREF</code> <code>ENTITY</code> <code>hexBinary</code>	すべてのマッピング・レベル <code>char[z]</code>
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> </xsd:restriction> </xsd:simpleType></pre> ここで、<code>type</code> は以下のいずれかです。 <code>duration</code> <code>date</code> <code>decimal</code> <code>time</code> <code>gDay</code> <code>gMonth</code> <code>gYear</code> <code>gMonthDay</code> <code>gYearMonth</code>	すべてのマッピング・レベル <code>char[32]</code>
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.2 以下の場合 <code>char[32]</code> マッピング・レベル 2.0 以上の場合 <code>char[40]</code> マッピング・レベル 3.0 以上の場合 <code>char[8]</code> タイム・スタンプのフォーマットは CICS ABSTIME です。

スキーマの simpleType	C および C++ のデータ・タイプ
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル signed char
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル char
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル short
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル unsigned short
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル char[33]
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル int
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル unsigned int
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル long long
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル unsigned long long
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル bool (C++ のみ) short (C のみ)
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	マッピング・レベル 1.0 の場合 サポートされていない マッピング・レベル 1.1 以上の場合 char[255]
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	マッピング・レベル 1.0 の場合 サポートされていない マッピング・レベル 1.1 以上の場合 char[255]

スキーマの simpleType	C および C++ のデータ・タイプ
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> 長さは定義されていません。	マッピング・レベル 1.1 以下の場合 char[y] ここで $y = 4 \times (\text{ceil}(z/3))$。ceil(x) は x 以上の最小の整数です。 マッピング・レベル 1.2 以上の場合 char[z] 固定長。 char[16] 長さが定義されていない場合にバイナリー・データを保管するコンテナの名前。
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.1 以下の場合 char[32] マッピング・レベル 1.2 以上の場合 float(*)
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.0 以下の場合 char[32] マッピング・レベル 1.2 以上の場合 double(*)

PL/I から XML スキーマへのマッピング

DFHLS2SC および DFHLS2WS ユーティリティー・プログラムは PL/I データ構造と XML スキーマ定義との間のマッピングをサポートします。Enterprise PL/I コンパイラと古い PL/I コンパイラの間には相違点があるため、PLI-ENTERPRISE と PLI-OTHER の 2 つの言語オプションがサポートされます。

PL/I の名前は、次の規則に従って XML の名前に変換されます。

1. XML エLEMENT 名で無効な文字は、「x」で置き換えられます。

例えば、monthly\$total は monthlyxtotal になります。

2. 重複した名前は、1 つ以上の数字を追加することによって固有にします。

例えば、year の 2 つのインスタンスは year と year1 になります。

DFHLS2SC および DFHLS2WS は次の表に従って PL/I データ・タイプをスキーマ・エレメントにマップします。この表にない PL/I タイプは DFHLS2SC または DFHLS2WS でサポートされません。次の制約事項も適用されます。

- COMPLEX 属性を持つデータ項目はサポートされません。
- FLOAT 属性を持つデータ項目は、マッピング・レベル 1.2 以上でサポートされます。Enterprise PL/I FLOAT IEEE はサポートされません。
- VARYING および VARYINGZ の純粋な DBCS スtring は、マッピング・レベル 1.2 以上でサポートされます。

- **DECIMAL(p,q)** として指定されたデータ項目は、 $p \geq q$ の場合のみサポートされます。
- **BINARY(p,q)** として指定されたデータ項目は、 $q = 0$ の場合のみサポートされます。
- データ項目に **PRECISION** 属性を指定しても、この属性は無視されます。
- **PICTURE** スtringはサポートされません。
- **ORDINAL** データ項目は、**FIXED BINARY(7)** データ・タイプとして扱われます。
- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを **NULL** に設定すると、文字配列は **xsd:string** にマップされ、ヌル終了Stringとして処理されます。このマッピングは Enterprise PL/I には適用されません。
- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを **BINARY** に設定すると、文字配列は **xsd:base64Binary** にマップされ、バイナリー・データとして処理されます。
- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを **COLLAPSE** に設定すると、`<xsd:whiteSpace value="collapse"/>` がString用に生成されます。

DFHLS2SC および DFHLS2WS は、PL/I の埋め込みアルゴリズムを完全には実装しないため、データ構造で埋め込みバイトを明示的に宣言する必要があります。DFHLS2SC および DFHLS2WS は、埋め込みバイトがないことを検出すると、メッセージを発行します。それぞれの最上位構造は、ダブルワード境界で開始して、構造内のそれぞれのバイトは正しい境界にマップされている必要があります。次のコード・フラグメントについて考えてみます。

```
3 FIELD1 FIXED BINARY(7),
3 FIELD2 FIXED BINARY(31),
3 FIELD3 FIXED BINARY(63);
```

この例では、次のようになります。

- **FIELD1** の長さは 1 バイトで、任意の境界に合わせることができます。
- **FIELD2** の長さは 4 バイトで、フルワード境界に合わせる必要があります。
- **FIELD3** の長さは 8 バイトで、ダブルワード境界に合わせる必要があります。

Enterprise PL/I コンパイラーはフィールドを以下の順序に調整します。

1. **FIELD3** の境界要件が最も厳しいため、**FIELD3** が最初に配置されます。
2. **FIELD2** は **FIELD3** の直前のフルワード境界に合わせます。
3. **FIELD1** が **FIELD3** の直前のバイト境界に合わせます。

最後に、構造全体がフルワード境界に合うように、コンパイラーは **FIELD1** の直前に 3 つの埋め込みバイトを挿入します。

DFHLS2WS は同等の埋め込みバイトを挿入しないため、DFHLS2WS が構造を処理する前にこれらの埋め込みバイトを明示的に宣言する必要があります。例を次に示します。


```

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);

```

または、すべてのフィールドを「位置合わせされていない」として宣言するよう構造を変更して、構造を使用するアプリケーションを再コンパイルすることもできます。PL/I 構造上のメモリー位置合わせ要件について詳しくは、「*Enterprise PL/I Language Reference*」を参照してください。

PL/I のデータ記述	スキーマ
FIXED BINARY (<i>n</i>) ここで $n \leq 7$	<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"/> </xsd:simpleType></pre>
FIXED BINARY (<i>n</i>) ここで $8 \leq n \leq 15$	<pre><xsd:simpleType> <xsd:restriction base="xsd:short"/> </xsd:simpleType></pre>
FIXED BINARY (<i>n</i>) ここで $16 \leq n \leq 31$	<pre><xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType></pre>
FIXED BINARY (<i>n</i>) ここで $32 \leq n \leq 63$ 制約事項: Enterprise PL/I のみ	<pre><xsd:simpleType> <xsd:restriction base="xsd:long"/> </xsd:simpleType></pre>
UNSIGNED FIXED BINARY(<i>n</i>) ここで $n \leq 8$ 制約事項: Enterprise PL/I のみ	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"/> </xsd:simpleType></pre>
UNSIGNED FIXED BINARY(<i>n</i>) ここで $9 \leq n \leq 16$ 制約事項: Enterprise PL/I のみ	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"/> </xsd:simpleType></pre>
UNSIGNED FIXED BINARY(<i>n</i>) ここで $17 \leq n \leq 32$ 制約事項: Enterprise PL/I のみ	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType></pre>
UNSIGNED FIXED BINARY(<i>n</i>) ここで $33 \leq n \leq 64$ 制約事項: Enterprise PL/I のみ	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"/> </xsd:simpleType></pre>
FIXED DECIMAL(<i>n,m</i>)	<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="<i>n</i>"/> <xsd:fractionDigits value="<i>m</i>"/> </xsd:restriction> </xsd:simpleType></pre>
FIXED DECIMAL(15) DATETIME=PACKED15 の場合マッピング・レベル 3.0 以上でサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre> タイム・スタンプのフォーマットは CICS ABSTIME です。

PL/I のデータ記述	スキーマ
BIT(n) ここで、 n は 8 の倍数です。この他の値はサポートされません。	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="m"/> </xsd:restriction> </xsd:simpleType></pre> ここで $m = n/8$
CHARACTER(n) VARYING および VARYINGZ はマッピング・レベル 1.2 以上でもサポートされる 制約事項: VARYINGZ は Enterprise PL/I でのみサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre>
GRAPHIC(n) VARYING および VARYINGZ はマッピング・レベル 1.2 以上でもサポートされる 制約事項: VARYINGZ は Enterprise PL/I でのみサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="m"/> </xsd:restriction> </xsd:simpleType></pre> マッピング・レベル 1.0 および 1.1 では、$m = 2*n$ <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> マッピング・レベル 1.2 以上
WIDECHAR(n) 制約事項: Enterprise PL/I のみ	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="m"/> </xsd:restriction> </xsd:simpleType></pre> マッピング・レベル 1.0 および 1.1 では、$m = 2*n$ <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="n"/> </xsd:restriction> </xsd:simpleType></pre> マッピング・レベル 1.2 以上
ORDINAL 制約事項: Enterprise PL/I のみ	<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"/> </xsd:simpleType></pre>
BINARY FLOAT(n) ここで $n \leq 21$ マッピング・レベル 1.2 以上でサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>

PL/I のデータ記述	スキーマ
BINARY FLOAT(<i>n</i>) ここで $21 < n \leq 53$ 53 より大きい値はサポートされない。 マッピング・レベル 1.2 以上でサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>
DECIMAL FLOAT(<i>n</i>) ここで $n \leq 6$ マッピング・レベル 1.2 以上でサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
DECIMAL FLOAT(<i>n</i>) ここで $6 < n \leq 16$ 16 より大きい値はサポートされない。 マッピング・レベル 1.2 以上でサポートされる	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

XML スキーマから PL/I へのマッピング

DFHSC2LS および DFHWS2LS ユーティリティー・プログラムは XML スキーマ定義と PL/I データ構造の間のマッピングをサポートします。Enterprise PL/I コンパイラと古い PL/I コンパイラの間には相違点があるため、PLI-ENTERPRISE と PLI-OTHER の 2 つの言語オプションがサポートされます。

CICS アシスタントは、次の規則を使用してスキーマ・エレメント名から PL/I 変数の固有で有効な名前を生成します。

1. A から Z、a から z、0 から 9、@、#、または \$ 以外の文字は、「X」で置き換えられます。

例えば、monthly-total は monthlyXtotal になります。

2. スキーマで変数が変化する基数を持つように指定する場合 (xsd:element で minOccurs 属性および maxOccurs 属性を異なる値で指定する場合)、スキーマ・エレメント名が 24 文字を超えると、この長さに切り捨てられます。

スキーマで変数が固定の基数を持つように指定する場合、スキーマ・エレメント名が 29 文字を超えると、この長さに切り捨てられます。

3. 同じスコープ内の重複した名前は、名前の 2 番目以降のインスタンスに 1 つ以上の数字を追加することによって固有にします。

例えば、year の 3 つのインスタンスは year、year1、および year2 になります。

4. スキーマで変数が変化する基数を持つように指定する場合 (minOccurs 属性および maxOccurs 属性を異なる値で指定する場合) に使用されるストリング _cont または _num 用に、5 文字が予約されます。

詳しくは、214 ページの『変化するエレメントの配列』を参照してください。

5. 属性では、前の規則がエレメント名に適用されます。接頭部 `attr-` がエレメント名に追加され、その後に `-value` または `-exist` が付きます。全長が 28 文字を超える場合、エレメント名は切り捨てられます。詳しくは、219 ページの『XML 属性のサポート』を参照してください。

`nillable` 属性には特別な規則があります。接頭部 `attr-` が追加されますが、エレメント名の先頭には `nil-` も追加されます。エレメント名の後には `-value` が付きます。全長が 28 文字を超える場合、エレメント名は切り捨てられます。

結果として生成される名前の全長は、31 文字以下になります。

DFHSC2LS および DFHWS2LS は、次の表に従ってスキーマ・タイプを PL/I のデータ・タイプにマップします。以下の点にも注意してください。

- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを `NULL` に設定すると、可変長文字データはヌル終了ストリングにマップされ、ヌル終止符として追加された 1 つの文字が割り振られます。
- **MAPPING-LEVEL** パラメーターを 1.2 以上に設定して、**CHAR-VARYING** パラメーターを指定しないと、可変長文字データはデフォルトでは、Enterprise PL/I の場合は `VARYINGZ` データ・タイプにマップされ、その他の PL/I の場合は `VARYING` データ・タイプにマップされます。
- 可変長バイナリー・データは、32 768 バイトより少ない場合は `VARYING` データ・タイプにマップされ、32 768 バイトを超える場合はコンテナにマップされます。

スキーマ	PL/I のデータ記述
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 2.0 以下の場合 サポートされていない マッピング・レベル 2.1 以上の場合 サポートされる
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.1 以上の場合CHAR(255)

スキーマ	PL/I のデータ記述
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> <xsd:maxLength value="z"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> ここで、<i>type</i> は以下のいずれかです。 - string - normalizedString - token - Name - NMTOKEN - language - NCName - ID - IDREF - ENTITY	すべてのマッピング・レベル CHARACTER(z)
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> </xsd:restriction> </xsd:simpleType></pre> ここで、<i>type</i> は以下のいずれかです。 - duration - date - time - gDay - gMonth - gYear - gMonthDay - gYearMonth	すべてのマッピング・レベル CHAR(32)
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.2 以下の場合 CHAR(32) マッピング・レベル 2.0 以上の場合 CHAR(40) マッピング・レベル 3.0 以上の場合 FIXED DECIMAL(15) タイム・スタンプのフォーマットは CICS ABSTIME です。

スキーマ	PL/I のデータ記述
<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="y"/> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.1 以下の場合 BIT(z) ここで $z = 8 \times y$ および $z < 4095$ バイト CHAR(z) ここで $z = 8 \times y$ および $z > 4095$ バイト マッピング・レベル 1.2 以上の場合 CHAR(y)
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル Enterprise PL/I SIGNED FIXED BINARY (7) その他の PL/I FIXED BINARY (7)
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル Enterprise PL/I UNSIGNED FIXED BINARY (8) その他の PL/I FIXED BINARY (8)
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル Enterprise PL/I SIGNED FIXED BINARY (15) その他の PL/I FIXED BINARY (15)
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル Enterprise PL/I UNSIGNED FIXED BINARY (16) その他の PL/I FIXED BINARY (16)
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル Enterprise PL/I FIXED DECIMAL(31,0) その他の PL/I FIXED DECIMAL(15,0)
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル Enterprise PL/I SIGNED FIXED BINARY (31) その他の PL/I FIXED BINARY (31)

スキーマ	PL/I のデータ記述
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.1 以下の場合 Enterprise PL/I UNSIGNED FIXED BINARY(32) マッピング・レベル 1.2 以上の場合 Enterprise PL/I CHAR(<i>y</i>) ここで <i>y</i> は 16 MB より小さい固定長です。 すべてのマッピング・レベル その他の PL/I BIT(64)
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.1 以下の場合 Enterprise PL/I SIGNED FIXED BINARY(63) マッピング・レベル 1.2 以上の場合 Enterprise PL/I CHAR(<i>y</i>) ここで <i>y</i> は 16 MB より小さい固定長です。 すべてのマッピング・レベル その他の PL/I BIT(64)
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.1 以下の場合 Enterprise PL/I UNSIGNED FIXED BINARY(64) マッピング・レベル 1.2 以上の場合 Enterprise PL/I CHAR(<i>y</i>) ここで <i>y</i> は 16 MB より小さい固定長です。 すべてのマッピング・レベル その他の PL/I BIT(64)

スキーマ	PL/I のデータ記述
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.1 以下の場合 Enterprise PL/I SIGNED FIXED BINARY (7) その他の PL/I FIXED BINARY (7) マッピング・レベル 1.2 以上の場合 Enterprise PL/I BIT(7) BIT(1) その他の PL/I BIT(7) BIT(1) ここで、BIT(7) は位置合わせのために提供されており、BIT(1) にはブールでマップされた値が含まれます。
<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="n"/> <xsd:fractionDigits value="m"/> </xsd:restriction> </xsd:simpleType></pre>	すべてのマッピング・レベル FIXED DECIMAL(n, m)
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	すべてのマッピング・レベル CHAR(255)
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	すべてのマッピング・レベル CHAR(255)
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value="y"/> </xsd:restriction> </xsd:simpleType></pre> 長さは定義されていません。	マッピング・レベル 1.0 の場合 サポートされていない マッピング・レベル 1.1 の場合 CHAR(z) ここで $z = 4 \times (\text{ceil}(y/3))$ です。 $\text{ceil}(x)$ は x 以上の最小の整数です。 マッピング・レベル 1.2 以上の場合 CHAR(y) 固定長。 CHAR(16) 長さは定義されていません。このフィールドには、バイナリー・データを保管するコンテナの 16 バイトの名前が入ります。

スキーマ	PL/I のデータ記述
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.0 および 1.1 の場合 CHAR(32) マッピング・レベル 1.2 以上の場合 Enterprise PL/I DECIMAL FLOAT(6) HEXADEC その他の PL/I DECIMAL FLOAT(6)
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	マッピング・レベル 1.0 および 1.1 の場合 CHAR(32) マッピング・レベル 1.2 以上の場合 Enterprise PL/I DECIMAL FLOAT(16) HEXADEC その他の PL/I DECIMAL FLOAT(16)

変化するエレメントの配列

XML は、エレメント数が増える配列を持つことができます。一般に、エレメント数が増える WSDL 文書や XML スキーマは 1 つの高水準言語データ構造に効率よくマップすることはできません。CICS はコンテナ・ベースのマッピングまたはインライン・マッピングを使用して XML におけるエレメント数の変化に対応します。

エレメント数が増える配列は、XML スキーマ内でエレメント宣言の `minOccurs` 属性および `maxOccurs` 属性を使用して表現されます。

- `minOccurs` 属性は、エレメントが発生できる最小の回数を指定します。この属性には、値 0 または任意の正の整数を指定できます。
- `maxOccurs` 属性は、エレメントが発生できる最大の回数を指定します。この属性には、`minOccurs` 属性の値以上の任意の正の整数値を指定することができます。エレメントが発生できる回数に上限がないことを示す値 `unbounded` を指定することもできます。
- これらの属性のデフォルト値は両方とも 1 です。

これは、アプリケーションの XML または SOAP メッセージ内でオプション、つまりゼロ回または 1 回発生が可能な 8 バイトのストリングを示します。

```
<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>
```

次の例は、1 回以上発生する必要がある 8 バイトのストリングを示しています。

```
<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>

```

一般に、エレメント数が増える WSDL 文書は 1 つの高水準言語データ構造に効率よくマップすることはできません。したがって、このような場合に対処するため、CICS は一連のコンテナとしてアプリケーション・プログラムに渡される結合された一続きのデータ構造を使用します。これらの構造はアプリケーションへの入出力として使用されます。

- CICS が XML をアプリケーション・データに変換する場合、これらの構造にアプリケーション・データが追加され、アプリケーションがそのデータを読み取ります。
- CICS がアプリケーション・データを XML に変換する場合、アプリケーションによって構造内に追加されたアプリケーション・データが読み取られます。

これらのデータ構造のフォーマットは、一連の例を使用すると分かりやすくなります。SOAP メッセージまたはアプリケーションの XML を使用できます。これらの例では、単純な 8 バイト・フィールドの配列を使用しています。ただし、モデルでは複合データ・タイプの配列、および他の配列を含むデータ・タイプの配列をサポートしています。

固定のエレメント数

最初の例では、ちょうど 3 回発生するエレメントを示しています。

```

<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>

```

この例では、エレメントが発生する回数があらかじめ分かっているため、次のように単純な COBOL 宣言 (または他の言語のこれに相当するもの) 内の固定長配列として表現することができます。

```
05 component PIC X(8) OCCURS 3 TIMES
```

マッピング・レベル 2 以下における変化するエレメント数

この例では、1 回から 5 回まで発生できる必須エレメントを示しています。

```

<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>

```

主データ構造には、2 つのフィールドの宣言が格納されます。CICS が XML をバイナリー・データに変換すると、最初のフィールドである component-num にはエレメントが XML 内に現れる回数が格納され、2 番目のフィールドである component-cont には、コンテナの名前が格納されます。

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

2 番目のデータ構造には、次のようなエレメントそのものの宣言が格納されます。

```
01 DFHWS-component
  02 component PIC X(8)
```

エレメントが発生する回数を割り出すには、`component-num` (1 から 5 までの範囲の値が入ります) の値を検査する必要があります。エレメント・コンテンツは `component-cont` で指定されたコンテナ内にあります。このコンテナはエレメントの配列を保持しており、各エレメントは `DFHWS-component` データ構造によってマップされています。

`minOccurs="0"` および `maxOccurs="1"` である場合、このエレメントはオプションです。アプリケーション・プログラムでデータ構造を処理するには、以下のように `component-num` の値を検査する必要があります。

- この値がゼロの場合、メッセージ内に `component` エレメントはなく、`component-cont` の内容は未定義です。
- この値が 1 の場合、`component` エレメントは `component-cont` で指定されたコンテナ内にあります。

コンテナの内容は、`DFHWS-component` データ構造によってマップされます。

注: SOAP メッセージが単一の繰り返しエレメントで構成される場合、`DFHWS2LS` は 2 つの言語構造を生成します。主要な言語構造は、配列エレメントの数と、エレメントの配列を保持するコンテナの名前を含んでいます。2 番目の言語構造は、繰り返しエレメントの単一インスタンスをマップします。

マッピング・レベル 2.1 以上における変化するエレメント数

マッピング・レベル 2.1 以上では、CICS アシスタントで **INLINE-MAXOCCURS-LIMIT** パラメーターを使用できます。**INLINE-MAXOCCURS-LIMIT** パラメーターは変化するエレメント数を処理する方法を指定します。変化するエレメント数のマッピング・オプションは、215 ページの『マッピング・レベル 2 以下における変化するエレメント数』に記述されているコンテナ・ベースのマッピング、またはインライン・マッピングです。このパラメーターの *value* は、0 - 32767 までの範囲の正整数です。

- **INLINE-MAXOCCURS-LIMIT** のデフォルト値は 1 であり、オプション・エレメントがインラインで確実にマップされます。
- **INLINE-MAXOCCURS-LIMIT** パラメーターに対して 0 の値を指定するとインライン・マッピングが抑止されます。
- `maxOccurs` が **INLINE-MAXOCCURS-LIMIT** の値以下の場合、インライン・マッピングが使用されます。
- `maxOccurs` の値が **INLINE-MAXOCCURS-LIMIT** の値より大きい場合、コンテナ・ベースのマッピングが使用されます。

変化するエレメント数をインラインでマップすると、配列 (上記の例では発生するオカレンスが固定) およびカウンターが生成されます。`component-num` フィールドは存在するエレメントのインスタンス数を示し、それらのインスタンスは配列によって示されます。215 ページの『マッピング・レベル 2 以下における変化するエ

メント数』で示される例では、**INLINE-MAXOCCURS-LIMIT** が 5 以下の場合に生成されるデータ構造は次のようになります。

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

最初のフィールド (component-num) は前のセクションで示されたコンテナ・ベース・マッピングの例の出力と同じです。2 番目のフィールドには長さが 5 の配列があり、生成される可能性のある最大エレメント数を収容できる大きさです。

インライン・マッピングは、エレメントの出現回数およびデータが収容されるコンテナの名前を格納するコンテナ・ベースのマッピングとは異なり、現行のコンテナにあるすべてのデータを格納します。現行のコンテナにデータを格納するとパフォーマンスが向上するので、インライン・マッピングの方が一般的には望ましいといえます。

ネストされた変数配列

複雑な WSDL 文書および XML スキーマには可変の繰り返しエレメントを含めることができ、そのエレメントにはさらに可変の繰り返しエレメントを含めることができます。この場合、記述される構造は、例で説明した 2 つのレベルを超えます。

この例は、1 回から 5 回出現する可能性がある必須エレメント (<component1>) 内でネストされたオプションのエレメント (<component2>) を示します。

```
<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>
```

最上位のデータ構造は、前の例のデータ構造とまったく同じです。

```
05 component1-num PIC S9(9) COMP-5  
05 component1-cont PIC X(16)
```

ただし、2 番目のデータ構造には、以下のエレメントが格納されます。

```
01 DFHWS-component1  
  02 component2-num PIC S9(9) COMP-5  
  02 component2-cont PIC X(16)
```

3 番目の構造には以下のエレメントが格納されます。

```
01 DFHWS-component2  
  02 component2 PIC X(8)
```

最外部のエレメント (<component1>) の出現回数は component1-num 内にあります。

component1-cont で指定されたコンテナには、2 番目のデータ構造 (DFHWS-component1) のその数のインスタンスを持つ配列が含まれています。

component2-cont の各インスタンスは、異なるコンテナを指定します。それぞれ、第 3 レベルの構造 (DFHWS-component2) によってマップされたデータ構造が含まれています。

この構造を説明するために、例と一致する次のような XML のフラグメントについて考えてみます。

```
<component1><component2>string1</component2></component1>
<component1><component2>string2</component2></component1>
<component1></component1>
```

<component1> は 3 回発生します。最初の 2 つには、それぞれ <component2> のインスタンスが含まれていますが、3 番目のインスタンスには含まれていません。

最上位のデータ構造では、component1-num に値 3 が含まれています。

component1-cont で指定されたコンテナには、DFHWS-component1 の次の 3 つのインスタンスがあります。

1. 第 1 のインスタンスでは、component2-num の値は 1 で、component2-cont で指定されたコンテナは *string1* を保持します。
2. 第 2 のインスタンスでは、component2-num の値は 1 で、component2-cont で指定されたコンテナは *string2* を保持します。
3. 第 3 のインスタンスでは、component2-num の値は 0 で、component2-cont の内容は未定義です。

このインスタンスでは、完全なデータ構造は、次の合計 4 つのコンテナによって表現されます。

- コンテナ DFHWS-DATA 内のルート・データ構造
- component1-cont で指定されたコンテナ
- component2-cont の最初の 2 つのインスタンスで指定された 2 つのコンテナ

オプションの構造および xsd:choice

DFHWS2LS および DFHSC2LS はマッピング・レベル 2.1 以上で、

<xsd:sequence>、<xsd:choice>、および <xsd:all> エレメントでの maxOccurs および minOccurs の使用をサポートします。この場合、minOccurs 属性および maxOccurs 属性は minOccurs="0" および maxOccurs="1" に設定されています。

アシスタントは、エレメント内の子エレメントがそれぞれオプションであるかのようにしてエレメントを処理するマッピングを生成します。これらのエレメントを使用してアプリケーションを実装する場合、アプリケーションが無効なオプションの組み合わせを生成しないように確認してください。それぞれのエレメントの生成された言語構造には count フィールドがありますが、これらのすべてが「0」に設定されるか、すべてが「1」に設定される必要があります。これ以外の値の組み合わせは、<xsd:choice> エレメントの場合を除き、無効です。

<xsd:choice> エレメントは、そのエレメントのオプションが 1 つのみ使用できることを示します。これはすべてのマッピング・レベルでサポートされます。アシスタントは <xsd:choice> のそれぞれのオプションを、minOccurs="0" および maxOccurs="1" である <xsd:sequence> エレメントのように扱います。<xsd:choice> エレメントを使用してアプリケーションを実装する場合は、アプリケーションが無効なオプションの組み合わせを生成しないように確認してください。それぞれのエ

メントの生成された言語構造には count フィールドがありますが、そのうちの 1 つが「1」に、他のすべてが「0」に設定される必要があります。これ以外の組み合わせはすべて無効です。例外は、<xsd:choice> のエレメント自体がオプションの場合であり、すべてのフィールドが 0 に設定されていても有効です。

XML 属性のサポート

XML スキーマは、XML で許可される属性または必須の属性を指定することができます。CICS アシスタント・ユーティリティである DFHWS2LS および DFHSC2LS はデフォルトで XML 属性を無視します。XML スキーマに定義されている XML 属性を処理するには、**MAPPING-LEVEL** パラメーターの値が 1.1 以上である必要があります。

オプションの属性

属性にはオプションの属性と必須属性があり、SOAP メッセージ内またはアプリケーションの XML 内の任意のエレメントに関連付けることができます。スキーマで定義されたすべてのオプションの属性ごとに、次に示す 2 つのフィールドが適切な言語構造で生成されます。

1. 存在フラグ。このフィールドはブール・データ・タイプとして扱われ、通常は長さが 1 バイトです。
2. 値。このフィールドは、同等のタイプの XML エレメントと同じ方法でマップされます。例えば、タイプ NMTOKEN の属性は、タイプ NMTOKEN の XML エレメントと同じ方法でマップされます。

属性の存在フィールドと値フィールドは、関連付けられたエレメントのフィールドの前に、生成された言語構造で表示されます。インスタンス文書に現れる予期しない属性は無視されます。

例えば、次のスキーマ属性定義について考えてみます。

```
<xsd:attribute name="age" type="xsd:short" use="optional" />
```

このオプションの属性は次に示す COBOL 構造にマップされます。

```
05 attr-age-exist PIC X DISPLAY  
05 attr-age-value PIC S9999 COMP-5 SYNC
```

オプションの属性のランタイム処理

オプションの属性では次に示すランタイム処理が実行されます。

- 属性が存在する場合は、存在フラグが設定され、値はマップされます。
- 属性が存在しない場合は、存在フラグは設定されません。
- 属性がデフォルト値を持ち、存在する場合は、値がマップされます。
- 属性がデフォルト値を持ち、存在しない場合は、デフォルト値がマップされます。

デフォルト値を持つオプションの属性は、必須属性として扱われます。

CICS がデータを XML に変換すると、次のランタイム処理が行われます。

- 存在フラグが設定される場合は、属性は変換されて XML に含められます。
- 存在フラグが設定されない場合は、属性は XML に含められません。

必須属性とランタイム処理

すべての必須属性で、値フィールドのみが適切な言語構造で生成されます。

属性が XML に存在する場合は、値がマップされます。属性が存在しない場合、次に示す処理が行われます。

- アプリケーションが Web サービス・プロバイダーである場合、CICS はクライアントの SOAP メッセージにエラーがあることを示す SOAP 障害メッセージを生成します。
- アプリケーションが Web サービス・リクエスターである場合、CICS はメッセージを発行し変換エラー応答 (RESP2 コード 13) をアプリケーションに戻します。
- アプリケーションが TRANSFORM XMLTODATA コマンドを使用している場合、CICS はメッセージを発行し、無効な要求応答 (RESP2 コード 3) をアプリケーションに戻します。

CICS が COMMAREA またはコンテナの内容を基にして SOAP メッセージを生成すると、属性は変換されて、メッセージに含められます。アプリケーションが TRANSFORM DATATOXML コマンドを使用する場合、CICS は属性の変換も行って XML に含めます。

nillable 属性

nillable 属性とは、XML スキーマの `xsd:element` で指定される場合のある特殊な属性です。これは、XML のエレメントで `xsi:nil` 属性が有効であることを指定します。エレメントで `xsi:nil` 属性が指定されている場合、エレメントは存在するが値がないため、このエレメントには内容が関連付けられていないことを示します。

XML スキーマが nillable 属性を `true` として定義した場合は、ブール値を使用する必須属性としてマップされます。

CICS が `xsi:nil` 属性を含む SOAP メッセージを受信した場合や、この属性を含む XML をアプリケーション用に変換する必要がある場合、属性の値は `true` または `false` です。値が `true` の場合は、`xsi:nil` 属性のスコープ内のエレメントまたはネストされたエレメントは、アプリケーションによって無視される必要があります。

CICS が、`xsi:nil` 属性の値が `true` である COMMAREA またはコンテナの内容を基にして SOAP メッセージまたは XML を生成すると、次に示す処理が行われます。

- `xsi:nil` 属性が XML または SOAP メッセージに生成されます。
- 関連するエレメントの値は無視されます。
- エレメント内でネストされたエレメントはすべて無視されます。

SOAP メッセージの例

WSDL 文書の一部になる場合がある次の例の XML スキーマについて考えてみます。

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="root" nillable="true">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element nillable="true" name="num" type="xsd:int" maxOccurs="3" minOccurs="3"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```

    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>

```

このスキーマに適合する部分的な SOAP メッセージの例を、次に示します。

```

<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <num xsi:nil="true"/>
  <num>15</num>
  <num xsi:nil="true"/>
</root>

```

COBOL では、この SOAP メッセージは次のエレメントにマップされます。

```

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)

```

<xsd:any> および xsd:anyType のサポート

DFHWS2LS および DFHSC2LS は XML スキーマにおける <xsd:any> および xsd:anyType の使用をサポートしています。<xsd:any> XML スキーマ・エレメントを使用して未定義の内容を持つ XML 文書のセクションを記述できます。xsd:anyType とは、すべての単純および複合データ・タイプの派生元となる基本のデータ・タイプです。データ内容に制限や制約はありません。

CICS アシスタントで <xsd:any> および xsd:anyType を使用する前に、次に示すパラメーターを設定してください。

- **MAPPING-LEVEL** パラメーターを 2.1 以上に設定します。
- Web サービス・プロバイダー・アプリケーションの場合、**PGMINT** パラメーターを CHANNEL に設定します。

<xsd:any> の例

この例では <xsd:any> エレメントを使用して、"Customer" タグの "Surname" タグの後に、オプションの非構造化 XML コンテンツを記述しています。

```

<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>

```

この XML スキーマに適合する SOAP メッセージの例を以下に示します。

```

<xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <Customer xmlns="http://www.example.org/anyExample">
      <Title xmlns="">Mr</Title>
      <FirstName xmlns="">John</FirstName>
      <Surname xmlns="">Smith</Surname>
      <ExtraInformation xmlns="http://www.example.org/ExtraInformation">

```

```

    <!-- This 'ExtraInformation' tag is associated with the optional xsd:any from the XML schema.
         It can contain any well formed XML. -->
    <ExampleField1>one</ExampleField1>
    <ExampleField2>two</ExampleField2>
  </ExtraInformation>
</Customer>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

この SOAP メッセージが CICS に送信されると、CICS は Customer-xml-cont コンテナに次に示す XML データを追加します。

```

<ExtraInformation xmlns="http://www.example.org/ExtraInformation">
  <!-- This 'ExtraInformation' tag is associated with the optional xsd:any from the XML schema.
       It can contain any well formed XML. -->
  <ExampleField1>one</ExampleField1>
  <ExampleField2>two</ExampleField2>
</ExtraInformation>

```

CICS はまた Customer-xmlns-cont コンテナに、次に示すスコープ内の XML 名前空間宣言を追加します。これらの宣言はスペースで区切られています。

```
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns="http://www.example.org/anyExample"
```

xsd:anyType の例

xsd:anyType とは、すべての単純および複合データ・タイプの派生元となる基本のデータ・タイプです。これはデータ内容を制限しません。データ・タイプを指定しないと、デフォルトとして xsd:anyType に設定されます。例えば、これらの 2 つの XML フラグメントは等価です。

```

<xsd:element name="Name" type="xsd:anyType"/>
<xsd:element name="Name"/>

```

生成される言語構造

< xsd:any > または xsd:anyType 用に生成された言語構造の形式は、COBOL では次のようになります。他の言語ではこれに相当する形式になります。

elementName-xml-cont PIC X(16)

未加工の XML を格納するコンテナの名前。CICS は、着信 SOAP メッセージを処理する際、<xsd:any> または xsd:anyType が定義する SOAP メッセージのサブセットをこのコンテナに置きます。アプリケーションは XML データをネイティブでのみ処理できます。アプリケーションは XML を生成し、このコンテナにデータを取り込み、コンテナ名を提供する必要があります。

このコンテナへの取り込みはテキスト・モードで行われる必要があります。CICS がこのコンテナにデータを取り込む場合、Web サービスが使用するように定義された EBCDIC のバリエーションと同じものを使用して取り込みが行われます。ターゲットの EBCDIC コード・ページに存在しない文字は置換文字によって置き換えられます。アプリケーションによってコンテナが UTF-8 で読み取られる場合も同様です。

elementName-xmlns-cont PIC X(16)

スコープ内の任意の名前空間の接頭部宣言を格納しているコンテナの名前。SOAP エンベロープ・タグのサブセットだけではなく、スコープ内にあ

り関連性のある名前空間の宣言がすべて含まれている点を除けば、このコンテナの内容は DFHWS-XMLNS コンテナの内容と似ています。

このコンテナへの取り込みはテキスト・モードで行われる必要があります。CICS がこのコンテナにデータを取り込む場合、Web サービスが使用するように定義された EBCDIC のバリエーションと同じものを使用して取り込みが行われます。ターゲットの EBCDIC コード・ページに存在しない文字は置換文字によって置き換えられます。アプリケーションによってコンテナが UTF-8 で読み取られる場合も同様です。

このコンテナは CICS に送信された SOAP メッセージを処理するときのみ使用されます。出力 SOAP メッセージの生成時にアプリケーションがコンテナに名前空間宣言を提供しようとする場合、コンテナとその内容は CICS によって無視されます。CICS では、アプリケーションが提供する XML に名前空間宣言が含まれていることが必要とされます。

`<xsd:any>` エレメントを含んでいる XML エレメントの名前は、`<xsd:any>` エレメント用に生成される変数名に組み込まれます。`<xsd:any>` の例では、`<xsd:any>` エレメントは `<xsd:element name="Customer">` エレメント内にネストされており、`<xsd:any>` エレメント用に生成される変数名は `Customer-xml-cont PIC X(16)` および `Customer-xmlns-cont PIC X(16)` です。

`xsd:anyType` タイプの場合、直接の XML エレメント名が使用されます。上記の `xsd:anyType` 例では、変数名は `Name-xml-cont PIC X(16)` および `Name-xmlns-cont PIC X(16)` です。

`<xsd:choice>`のサポート

`<xsd:choice>` エレメントは、そのエレメントでオプションが 1 つだけ使用できることを示します。CICS アシスタントは、それぞれのマッピング・レベルに応じて `<xsd:choice>` エレメントをさまざまな度合いでサポートします。

マッピング・レベル 2.2 以上における `<xsd:choice>`のサポート

マッピング・レベル 2.2 以上では、DFHWS2LS および DFHSC2LS の `<xsd:choice>` エレメントのサポートが改善されています。アシスタントは `<xsd:choice>` エレメントに関連した値を格納する新しいコンテナを生成します。アシスタントは、以下に示すように新規コンテナ名および追加のフィールド名を含む言語構造を生成します。

fieldname-enum

`<xsd:choice>` エレメントが使用するオプションを区別して示すフィールド。

fieldname-cont

使用されるオプションを格納するコンテナの名前。オプションの値をマップするために、さらに言語構造が生成されます。

次に示す XML スキーマのフラグメントには `<xsd:choice>` エレメントが含まれています。

```
<xsd:element name="choiceExample">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="option1" type="xsd:string" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

```

        <xsd:element name="option2" type="xsd:int" />
        <xsd:element name="option3" type="xsd:short" maxOccurs="2" minOccurs="2" />
    </xsd:choice>
</xsd:complexType>
</xsd:element>

```

この XML スキーマ・フラグメントがマッピング・レベル 2.2 以上で処理される場合、アシスタントは次に示す COBOL 言語構造を生成します。

```

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.

```

マッピング・レベル 2.2 以上における <xsd:choice> の制限

DFHSC2LS および DFHWS2LS はネストされた <xsd:choice> エレメントをサポートしません。例えば、次に示す XML はサポートされません。

```

<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 および DFHWS2LS は循環する <xsd:choice> エレメントをサポートしません。例えば、次の XML はサポートされません。

```

<xsd:choice maxOccurs="2">
  <xsd:element name="name1" type="string"/>
</xsd:choice>

```

DFHSC2LS および DFHWS2LS では、1 つの <xsd:choice> エレメントで最大 255 までのオプションをサポートします。

マッピング・レベル 2.1 以下における <xsd:choice> のサポート

2.1 以下のマッピング・レベルでは、DFHWS2LS が <xsd:choice> エレメントに関して提供するサポートが限定されます。DFHWS2LS は <xsd:choice> エレメントのそれぞれのオプションを、最大で 1 回生じる可能性のある <xsd:sequence> エレメントのように扱います。

<xsd:choice> エレメントのオプションのうち 1 つしか使用できないため、<xsd:choice> エレメントを使用してアプリケーションを実装する場合は有効なオプションの組み合わせのみが生成されるかどうかに注意してください。それぞれのエ

レメントの生成された言語構造には count フィールドがありますが、そのうちの 1 つが 1 に、他のすべてが 0 に設定される必要があります。これ以外の組み合わせはすべて無効です。例外は、<xsd:choice> 自体がオプションの場合であり、すべてのフィールドが 0 に設定されても有効です。

関連資料

221 ページの『<xsd:any> および xsd:anyType のサポート』

DFHWS2LS および DFHSC2LS は XML スキーマにおける <xsd:any> および xsd:anyType の使用をサポートしています。<xsd:any> XML スキーマ・エレメントを使用して未定義の内容を持つ XML 文書のセクションを記述できます。xsd:anyType とは、すべての単純および複合データ・タイプの派生元となる基本のデータ・タイプです。データ内容に制限や制約はありません。

226 ページの『抽象エレメントおよび抽象データ・タイプのサポート』

CICS アシスタントはマッピング・レベル 2.2 以上で抽象エレメントおよび抽象データ・タイプをサポートします。CICS アシスタントは、置換グループと同様の方法で抽象エレメントおよび抽象データ・タイプをマップします。

『置換グループのサポート』

置換グループを使用して、交換可能な XML エLEMENT のグループを定義できます。CICS アシスタントはマッピング・レベル 2.2 以上で置換グループをサポートしています。

置換グループのサポート

置換グループを使用して、交換可能な XML エLEMENT のグループを定義できます。CICS アシスタントはマッピング・レベル 2.2 以上で置換グループをサポートしています。

マッピング・レベル 2.2 以上では、DFHSC2LS および DFHWS2LS は <xsd:choice> エLEMENT に使用するマッピングに似たマッピングを使用して置換グループをサポートします。アシスタントは列挙フィールドと新規のコンテナ名を、その言語構造で生成します。

次に示す XML スキーマ・フラグメントには 2 つの subGroupParent エLEMENT の配列が含まれており、それぞれ replacementOption1 または replacementOption2 と置換可能です。

```
<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" />
```

この XML フラグメントをアシスタントで処理すると、次に示す COBOL 言語構造が生成されます。

```
03 subGroupExample.
  06 subGroupParent OCCURS 2.
    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'.
009 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).

```

置換グループについて詳しくは、「*W3C XML Schema Part 1: Structures Second Edition specification*」(http://www.w3.org/TR/xmlschema-1/#Elements_Equivalence_Class)を参照してください。

関連資料

221 ページの『<xsd:any> および xsd:anyType のサポート』

DFHWS2LS および DFHSC2LS は XML スキーマにおける <xsd:any> および xsd:anyType の使用をサポートしています。<xsd:any> XML スキーマ・エレメントを使用して未定義の内容を持つ XML 文書のセクションを記述できます。xsd:anyType とは、すべての単純および複合データ・タイプの派生元となる基本のデータ・タイプです。データ内容に制限や制約はありません。

223 ページの『<xsd:choice>のサポート』

<xsd:choice> エレメントは、そのエレメントでオプションが 1 つだけ使用できることを示します。CICS アシスタントは、それぞれのマッピング・レベルに応じて <xsd:choice> エレメントをさまざまな度合いでサポートします。

『抽象エレメントおよび抽象データ・タイプのサポート』

CICS アシスタントはマッピング・レベル 2.2 以上で抽象エレメントおよび抽象データ・タイプをサポートします。CICS アシスタントは、置換グループと同様の方法で抽象エレメントおよび抽象データ・タイプをマップします。

抽象エレメントおよび抽象データ・タイプのサポート

CICS アシスタントはマッピング・レベル 2.2 以上で抽象エレメントおよび抽象データ・タイプをサポートします。CICS アシスタントは、置換グループと同様の方法で抽象エレメントおよび抽象データ・タイプをマップします。

マッピング・レベル 2.2 以上における抽象エレメントのサポート

マッピング・レベル 2.2 以上では、DFHSC2LS および DFHWS2LS は、抽象エレメントが置換グループの有効なメンバーではないことを除くと、抽象エレメントを置換グループとほぼ同じ方法で扱います。置換可能なエレメントがない場合、抽象エレメントは <xsd:any> エレメントとして扱われ、マッピング・レベル 2.1 における <xsd:any> エレメントと同じマッピングを使用します。

次に示す XML スキーマ・フラグメントは、抽象エレメントの代わりに使用できる 2 つのオプションを指定します。抽象エレメント自体は有効なオプションではありません。


```

<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" />

```

この XML フラグメントをアシスタントで処理すると、次に示す COBOL 言語構造が生成されます。

```

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.

```

抽象エレメントについて詳しくは、「*W3C XML Schema Part 0: Primer Second Edition specification*」(<http://www.w3.org/TR/xmlschema-0/#SubsGroups>)を参照してください。

マッピング・レベル 2.2 以上における抽象データ・タイプのサポート

マッピング・レベル 2.2 以上では、DFHSC2LS および DFHWS2LS が抽象データ・タイプを置換グループとして扱います。アシスタントは列挙フィールドと新規のコンテナ名を、その言語構造で生成します。

次に示す XML スキーマ・フラグメントは、抽象タイプの代わりに使用できる 2 つのオプションを指定します。

```

<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>

```

この XML フラグメントをアシスタントで処理すると、次に示す COBOL 言語構造が生成されます。

```
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).
```

言語構造は別々のコピーブックに生成されます。option1 用に生成された言語構造は次に示すとおり 1 つのコピーブックに生成されます。

```
03 option1                       PIC X(5).
```

option2 の言語構造は次に示すとおり別のコピーブックに生成されます。

```
03 option2                       PIC X(10).
```

抽象データ・タイプについて詳しくは、「*W3C XML Schema Part 0: Primer Second Edition specification*」(<http://www.w3.org/TR/xmlschema-0/#SubsGroups>) を参照してください。

関連資料

221 ページの『<xsd:any> および xsd:anyType のサポート』

DFHWS2LS および DFHSC2LS は XML スキーマにおける <xsd:any> および xsd:anyType の使用をサポートしています。<xsd:any> XML スキーマ・エレメントを使用して未定義の内容を持つ XML 文書のセクションを記述できます。xsd:anyType とは、すべての単純および複合データ・タイプの派生元となる基本のデータ・タイプです。データ内容に制限や制約はありません。

223 ページの『<xsd:choice>のサポート』

<xsd:choice> エレメントは、そのエレメントでオプションが 1 つだけ使用できることを示します。CICS アシスタントは、それぞれのマッピング・レベルに応じて <xsd:choice> エレメントをさまざまな度合いでサポートします。

225 ページの『置換グループのサポート』

置換グループを使用して、交換可能な XML エレメントのグループを定義できます。CICS アシスタントはマッピング・レベル 2.2 以上で置換グループをサポートしています。

COBOL における可変の繰り返しコンテンツの処理方法

COBOL では、データの各インスタンスを扱うポインター演算を使用して、可変繰り返しコンテンツを処理することはできません。他のプログラミング言語にはこの制限はありません。この例では、Web サービス・アプリケーションのために COBOL で可変の繰り返しコンテンツを処理する方法を示します。

この方法は、TRANSFORM API コマンドを使用する XML のアプリケーション・データへの変換にも適用できます。次の WSDL 文書の例は、繰り返し回数が可変である 8 文字のストリングからなるアプリケーション・データを含む Web サービスを表します。

```
<?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>

```

この WSDL 文書を DFHWS2LS を使用して処理すると、次に示す COBOL 言語構造が生成されます。

```

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).

```

8 文字の component フィールドが、DFHWS-component という名前の別の構造で定義されていることに注意してください。メインのデータ構造は applicationData と呼ばれており、component-num および component-cont という 2 つのフィールドを含んでいます。component-num フィールドは、component データのインスタンス数を示し、component-cont フィールドは component フィールドの連結リストを格納するコンテナの名前を示します。

次に示す COBOL コードは、可変の繰り返しデータのリストを処理する方法の 1 つを例示しています。ここでは linkage section 配列を使用してデータの後続インスタンスを処理しています。インスタンスはそれぞれ DISPLAY ステートメントを使用して表示されます。

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EXVARY.

    ENVIRONMENT DIVISION.
    DATA DIVISION.
    WORKING-STORAGE SECTION.

    * working storage variables
    01 APP-DATA-PTR          USAGE IS POINTER.
    01 APP-DATA-LENGTH      PIC S9(8) COMP.
    01 COMPONENT-PTR        USAGE IS POINTER.
    01 COMPONENT-DATA-LENGTH PIC S9(8) COMP.
    01 COMPONENT-COUNT      PIC S9(8) COMP-4 VALUE 0.
    01 COMPONENT-LENGTH    PIC S9(8) COMP.

    LINKAGE SECTION.

    * a large linkage section array
    01 BIG-ARRAY PIC X(659999).

    * application data structures produced by DFHWS2LS
    * this is normally referenced with a COPY statement
    01 DFHWS2LS-data.
        03 applicationData.
            06 component-num PIC S9(9) COMP-5 SYNC.
            06 component-cont PIC X(16).

    01 DFHWS-component.
        03 component      PIC X(8).

    PROCEDURE DIVISION USING DFHEIBLK.
    A-CONTROL SECTION.
    A010-CONTROL.

    * Get the DFHWS-DATA container
    EXEC CICS GET CONTAINER('DFHWS-DATA')
        SET(APP-DATA-PTR)
        FLENGTH(APP-DATA-LENGTH)
    END-EXEC
    SET ADDRESS OF DFHWS2LS-data TO APP-DATA-PTR

    * Get the recurring component data
    EXEC CICS GET CONTAINER(component-cont)
        SET(COMPONENT-PTR)
        FLENGTH(COMPONENT-DATA-LENGTH)
    END-EXEC

    * Point the component structure at the first instance of the data
    SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

    * Store the length of a single component
    MOVE LENGTH OF DFHWS-component TO COMPONENT-LENGTH

    * process each instance of component data in turn
    PERFORM WITH TEST AFTER
        UNTIL COMPONENT-COUNT = component-num

    * display the current instance of the data
    DISPLAY 'component value is: ' component

    * address the next instance of the component data
    SET ADDRESS OF BIG-ARRAY TO ADDRESS OF DFHWS-component
    SET ADDRESS OF DFHWS-component
        TO ADDRESS OF BIG-ARRAY (COMPONENT-LENGTH + 1:1)
    ADD 1 TO COMPONENT-COUNT

```

```

* end the loop
  END-PERFORM.

* Point the component structure back at the first instance of
* of the data, for any further processing we may want to perform
  SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

* return to CICS.

  EXEC CICS
    RETURN
  END-EXEC

  GOBACK.

```

上記のコードは可変の繰り返しコンテンツの一般的な処理方法を提供しています。配列 BIG-ARRAY はそれぞれのコンポーネントの先頭に順次移動し、データの先頭に固定されません。コンポーネントのデータ構造は、次のコンポーネントの最初のバイトを指すように移動します。COMPONENT-PTR を使用して、必要に応じてコンポーネント・データの開始位置を回復できます。

WSDL 文書に準拠する SOAP メッセージの例を次に示します。

```

<?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>

```

COBOL プログラムが SOAP メッセージを処理する際に生成される出力は次のとおりです。

```

CPIH 20080115103151 component value is: VALUE1
CPIH 20080115103151 component value is: VALUE2
CPIH 20080115103151 component value is: VALUE3

```

Web サービス・アシスタントを使用した Web サービス・プロバイダーの作成

WSDL 1.1 または WSDL 2.0 に準拠する Web サービス記述から、または高水準言語データ構造から、サービス・プロバイダー・アプリケーションを作成できます。CICS Web サービス・アシスタントは、サービス・プロバイダーの設定における CICS アプリケーションの配置を支援します。

このタスクについて

このアシスタントを使用し、CICS アプリケーションをサービス・プロバイダーとして配置する場合は、以下の 2 つのオプションがあります。

- Web サービス記述から開始し、Web サービス・アシスタントを使用して言語データ構造を生成する。

既存の Web サービス記述に適合するサービス・プロバイダーを実装する場合は、このオプションを使用します。

- 言語データ構造から開始し、Web サービス・アシスタントを使用して Web サービス記述を生成する。

既存のプログラムを Web サービスとして公開しており、このプログラムのインターフェースの外観を Web サービス記述と SOAP メッセージで公開しようとする場合は、このオプションを使用します。

Web サービス記述を基にしたサービス・プロバイダー・アプリケーションの作成

CICS Web サービス・アシスタントを使用すると、WSDL 1.1 または WSDL 2.0 に準拠する Web サービス記述を基にしてサービス・プロバイダー・アプリケーションを作成できます。

始める前に

サービス・プロバイダー・アプリケーションを作成する準備として、以下の点を確認してください。

- Web サービス記述が z/OS の UNIX ファイルに含まれている必要があり、適切なプロバイダー・モード・パイプラインを CICS 領域にインストールする必要があります。
- DFHWS2LS の実行に使用するユーザー ID を OMVS に定義する必要があります。
- このユーザー ID には、z/OS UNIX ライブラリーと PDS ライブラリーに対する読み取り権限と、**LOGFILE**、**WSBIND**、および **WSDL** パラメーターに指定されたディレクトリーに対する書き込み権限が必要です。
- Java を実行するための十分なストレージをこのユーザー ID に割り振る必要があります。

このタスクについて

入力として使用する Web サービス記述では、以下のステップを実行します。

1. Web サービス・バインディング・ファイルおよび 1 つ以上の言語データ構造を生成する場合は、DFHWS2LS バッチ・プログラムを使用します。DFHWS2LS には、アプリケーションで必要なバインディング・ファイルと言語構造を作成する際に柔軟性を提供する、多数のオプション・パラメーターが含まれています。既存のアプリケーションを Web サービスで使用可能にするときは、以下のオプションを検討してください。
 - サービス・プロバイダー・アプリケーション・プログラムにデータを渡すために CICS が使用すべきメカニズムは? チャネルを使用してコンテナ内のデータを渡すか、**COMMAREA** を使用することができます。チャネルとコンテナが推奨されます。これは、**PGMINT** パラメーターを使用して指定します。
 - 生成する言語は? DFHWS2LS は、COBOL、C/C++、または PL/I 言語データ構造を生成できます。言語は、**LANG** パラメーターを使用して指定します。
 - 使用するマッピング・レベルは? マッピング・レベルが高いほど、実行時に文字とバイナリー・データの処理の制御とサポートが行いやすくなります。一部

のオプション・パラメーターは、高いマッピング・レベルでしか使用できません。使用できる最高のマッピング・レベルを使用することをお勧めします。

- Web サービス・リクエスターで使用する URI は? **URI** パラメーターを使用して相対 URI を指定します。例えば、URI=/my/test/webservice です。この値は、URIMAP リソースの作成時に CICS によって使用されます。
- Web サービス要求と応答を実行するためのトランザクションとユーザー ID は? 別名トランザクションを使用すると、サービス・リクエスターへの応答を構成するためのアプリケーションを実行することができます。別名トランザクションは、ユーザー ID を基にして接続されます。これは、**TRANSACTION** パラメーターと **USERID** パラメーターを使用して指定します。これらの値は、URIMAP リソースの作成時に使用されます。特定のトランザクションを使用したくない場合は、これらのパラメーターを使用しないでください。
- WSDL 文書の保管場所は? ローカル・ファイル・システムからではなく、IBM WebSphere Service Registry and Repository (WSRR) サーバーから WSDL 文書を取り出す場合は、DFHWS2LS でいくつかのパラメーターを指定する必要があります。少なくとも、**WSRR-SERVER** パラメーターで WSRR サーバーの場所を指定し、**WSRR-NAME** パラメーターでは WSRR から取り出す WSDL 文書の名前を指定する必要があります。WSRR を使用する場合に指定可能な他のパラメーターについては、インフォメーション・センターのトピック『DFHWS2LS: WSDL から高水準言語への変換』を参照してください。
- WSDL 文書を WSRR サーバーから取り出す場合、セキュア接続を使用しますか? Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用して、安全に WSRR と相互運用することができます。例については、333 ページの『Web サービス・アシスタントと WSRR で SSL を使用する方法の例』を参照してください。

DFHWS2LS を実行依頼すると、CICS は Web サービス・バインディング・ファイルを生成して、ユーザーが **WSBIND** パラメーターを使用して指定した場所に置きます。言語構造は、ユーザーが **PDSLIB** パラメーターを使用して指定した区分データ・セットに置かれます。

2. 生成された Web サービス・バインディング・ファイルを、Web サービス・アプリケーションに使用するプロバイダー・モード PIPELINE リソースの pickup ディレクトリーにコピーします。バインディング・ファイルはバイナリー・モードでコピーする必要があります。
3. オプション: Web サービス記述を、Web サービス・バインディング・ファイルと同じディレクトリーにコピーします。このコピーを使用して、Web サービスが実行時に期待どおりに機能することをテストできます。
4. 生成された言語構造とインターフェースを取るサービス・プロバイダー・アプリケーション・プログラムを作成して、必要なビジネス・ロジックを実装します。
5. RDO を使用して必要なリソースを作成することも可能ですが、PIPELINE SCAN コマンドを使用して、WEBSERVICE リソースと URIMAP リソースを動的に作成することをお勧めします。
 - WEBSERVICE リソースは、CICS で Web サービス・バインディング・ファイルをカプセル化し、実行時に使用されます。
 - URIMAP リソースは、WEBSERVICE リソースを特定の URI に関連付けるための情報を CICS に提供します。

タスクの結果

DFHWS2LS の実行依頼時に問題が発生した場合や、リソースが正しくインストールされない場合は、335 ページの『配置エラーの診断』を参照してください。

データ構造を基にしたサービス・プロバイダー・アプリケーションの作成

CICS Web サービス・アシスタントを使用すると、高水準言語のデータ構造を基にしてサービス・プロバイダー・アプリケーションを作成できます。

始める前に

サービス・プロバイダー・アプリケーションを作成する準備として、以下の前提条件がすべて満たされていることを確認してください。

- 高水準言語データ構造は、次の基準を満たす必要があります。
 - データ構造は、例えば COBOL コピーブック内など、ソース・プログラムとは別個に定義される必要があります。
 - PL/I または COBOL アプリケーション・プログラムが使用するデータ構造が入力と出力で異なる場合、このデータ構造は、区分データ・セットに存在する 2 つの異なるメンバーに定義されます。入力と出力で同じデータ構造を使用している場合は、1 つのメンバーにデータ構造を定義してください。

C および C++ では、区分データ・セット内の同じメンバーにデータ構造を置くことができます。

- ラッパー・プログラムを使用するかどうかに応じて、処理対象のデータ構造は次のように異なります。
 - ラッパー・プログラムを使用している場合、コピーブックはラッパー・プログラムのインターフェースになります。
 - ラッパー・プログラムを使用していない場合、コピーブックはビジネス・ロジックのインターフェースになります。
- 言語構造が区分データ・セットで使用可能になっており、適切なパイプラインが CICS 領域にインストールされていなければなりません。
 - DFHLS2WS の実行に使用するユーザー ID を OMVS に定義する必要があります。
 - このユーザー ID には、z/OS UNIX ライブラリーと PDS ライブラリーに対する読み取り権限と、LOGFILE、WSBIND、および WSDL パラメーターに指定されたディレクトリーに対する書き込み権限が必要です。
 - また、Java を実行するため、このユーザー ID に十分な大きさのストレージを割り振る必要があります。

このタスクについて

データ構造を基にしたサービス・プロバイダー・アプリケーションを作成するには、以下のステップを実行します。

1. サービス・プロバイダー・アプリケーション・インターフェースがチャネルおよび多数のコンテナを使用する場合には、XML でインターフェースを記述するチャネル記述文書を作成してください。チャネル記述文書を z/OS UNIX 上の適

切なディレクトリーに格納する必要があります。CICSはこの文書を使用して、チャンネル上のコンテナから SOAP メッセージを構成および分解します。あるいは、チャンネル記述文書を作成せずに、チャンネル上の 1 つのコンテナを使用することもできます。

チャンネル記述文書を作成する方法について、詳しくは、237 ページの『チャンネル記述文書の作成』を参照してください。

2. 言語構造を基にして Web サービス・バインディング・ファイルおよび Web サービス記述を生成する場合は、DFHLS2WS バッチ・プログラムを使用します。DFHLS2WS には、アプリケーションに必要なバインディング・ファイルと言語構造を作成する際に柔軟性を提供する、多数のオプション・パラメーターが含まれています。既存のアプリケーションを Web サービスとして使用可能にするために検討すべきオプションは、次のとおりです。

- サービス・プロバイダー・アプリケーション・プログラムにデータを渡すために CICS が使用すべきメカニズムは? チャンネルを使用してコンテナ内のデータを渡すか、COMMAREA を使用することができます。メカニズムは、PGMINT パラメーターを使用して指定します。

アプリケーション・インターフェースがチャンネルおよび多数のコンテナを使用する場合には、**REQUEST-CHANNEL** パラメーターを指定し、オプションで **RESPONSE-CHANNEL** を指定します。これらのパラメーターは、マッピング・レベルが 3.0 以上の場合にのみ使用できます。

- 生成する Web サービス記述 (WSDL 文書) のレベルは? CICS は、WSDL 1.1 文書または WSDL 2.0 文書と準拠する記述を生成します。両方のレベルの WSDL と準拠する要求をサービス・プロバイダー・アプリケーションでサポートするには、**WSDL_1.1** パラメーターと **WSDL_2.0** パラメーターに値を指定します。複数の WSDL パラメーターを使用する場合は、異なるファイル名になるようにします。この仕様によって、2 つの Web サービス記述とバインディング・ファイルが生成されます。
- 使用する SOAP プロトコルのバージョンは? **SOAPVER** パラメーターを使ってバージョンを指定できます。ALL 値の使用をお勧めします。その場合、SOAP 1.2 メッセージ・ハンドラーで構成されたパイプラインに Web サービスをインストールしなければならなくなりますが、SOAP 1.1 または SOAP 1.2 のいずれかを Web サービス記述のバインディングとして柔軟に使用できます。このパラメーターを使用できるのは、**MINIMUM-RUNTIME-LEVEL** が 2.0 以上の場合だけです。
- 使用するマッピング・レベルは? マッピング・レベルが高いほど、実行時に文字とバイナリー・データの処理の制御とサポートが行いやすくなります。一部のオプション・パラメーターは、高いマッピング・レベルでしか使用できません。使用できる最高のマッピング・レベルを使用することをお勧めします。
- Web サービス・リクエスターで使用する URI は? **URI** パラメーターを使用して絶対 URI を指定します。例えば、**URI=http://www.example.org:80/my/test/webservice** です。このアドレスの相対部分 (/my/test/webservice) は、URIMAP リソースの作成時に使用されます。完全 URI は、Web サービス記述で <soap:address> エレメントとして使用されます。この使用法は、HTTP URI と WMQ URI の両方に当てはまります。

- WSDL 文書を IBM WebSphere Service Registry and Repository (WSRR) に公開しますか? WSDL 文書を WSRR に公開する場合、DFHLS2WS で **WSRR-SERVER** パラメーターを指定する必要があります。WSRR を使用する場合に指定できるパラメーターについて、詳しくは、153 ページの『DFHLS2WS: 高水準言語から WSDL への変換』を参照してください。
- WSDL 文書を WSRR サーバー上に公開する場合、セキュア接続を使用しますか? Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用して、安全に WSRR と相互運用することができます。例については、333 ページの『Web サービス・アシスタントと WSRR で SSL を使用する方法の例』を参照してください。

DFHLS2WS を実行依頼すると、CICS は Web サービス・バインディング・ファイルを生成して、ユーザーが **WSBIND** パラメーターを使用して指定した場所に置きます。生成された Web サービス記述は、ユーザーが **WSDL**、**WSDL_1.1**、または **WSDL_2.0** パラメーターを使用して指定した場所に置かれます。

DFHLS2WS で WSRR パラメーターを使用した場合には、指定された WSRR サーバーに WSDL 文書が公開されます。

3. 生成された Web サービス記述を確認して、必要なカスタマイズを行います。詳しくは、238 ページの『生成した Web サービス記述文書のカスタマイズ』を参照してください。
4. Web サービス・バインディング・ファイルを、Web サービス・アプリケーションに使用するプロバイダー・モード・パイプラインのピックアップ・ディレクトリーにコピーします。Web サービス・バインディング・ファイルはバイナリー・モードでコピーする必要があります。
5. オプション: Web サービス記述を、Web サービス・バインディング・ファイルと同じディレクトリーにコピーします。このコピーを使って検証を行うことにより、Web サービスが実行時に期待どおりに機能することをテストできます。
6. RDO を使用して必要なリソースを作成することも可能ですが、PIPELINE SCAN コマンドを使用して、WEBSERVICE リソースと URIMAP リソースを動的に作成することをお勧めします。
 - WEBSERVICE リソースは、CICS で Web サービス・バインディング・ファイルをカプセル化し、実行時に使用されます。
 - URIMAP リソースは、WEBSERVICE リソースを特定の URI に関連付けるための情報を CICS に提供します。

タスクの結果

CICS リソースを正常に作成したら、サービス・プロバイダー・アプリケーションの作成は完了です。

DFHLS2WS の実行依頼時に問題が発生した場合や、リソースが正しくインストールされない場合は、335 ページの『配置エラーの診断』を参照してください。

次のタスク

サービスにアクセスする Web サービス・リクエスターを作成する必要があるすべてのユーザーが、Web サービス記述を使用することです。

チャネル記述文書の作成

サービス・プロバイダー・アプリケーションで多数のコンテナを持つチャネル・インターフェースを使用する場合、チャネル記述文書を作成します。

このタスクについて

XML エディターを使用して、チャネル記述文書を作成します。チャネル記述のスキーマは、channel.xsd という名前で、z/OS UNIX 上の /usr/lpp/cicsts/cicsts41/schemas/channel/ ディレクトリーにあります。

1. 次のようにして、<channel> エレメントおよび CICS チャネル名前空間を持つ XML 文書を作成します。

```
<channel name="myChannel" xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```

2. アプリケーション・プログラム・インターフェースがチャネルで使用するコンテナごとに 1 つの <container> エレメントを追加します。それぞれのコンテナを記述する名前、タイプ、および使用属性を使用する必要があります。以下の例は、異なる属性値を持つ 6 つのコンテナを示しています。

```
<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>
```

この structure エレメントは、区分データ・セット・メンバーにある言語構造で内容が定義されていることを示します。

3. XML 文書を z/OS UNIX で保存します。

チャネル・スキーマ

チャネル記述文書は、以下のスキーマに従う必要があります。

```
<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>
```

```

        </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. このエレメントは CICS チャネルを表します。
2. このエレメントはチャネル内の 1 つの CICS コンテナを表します。
3. structure は「ビット」モードのコンテナでのみ使用できます。 location 属性は、コンテナの内容をマップするファイルの場所を示します。 structure 属性を C および C++ で使用して、構造体の名前を示すことができます。

次のタスク

DFHLS2WS を実行して、Web サービス・プロバイダー・アプリケーション用のマッピングと WSDL 文書を作成します。DFHLS2WS は、コンテナがチャネル記述文書で指定されている順序で、チャネルのマッピングを WSDL 文書に入れます。

生成した Web サービス記述文書のカスタマイズ

DFHLS2WS によって生成される Web サービス記述 (WSDL) 文書には、公開前にユーザーが変更した方がよい場合がある、自動的に生成される内容が含まれています。WSDL 文書をカスタマイズすると、Web サービス・バインディング・ファイルが再生成されることがあります。また、場合によっては、ラッパー・プログラムが作成されることもあります。

このタスクについて

以下の手順に従って、生成された Web サービス記述文書をカスタマイズします。

1. HTTPS のサポートを公示するか、WebSphere MQ を使用して通信するには、DFHLS2WS で **URI** パラメーターを使用して、絶対 URI を設定します。**URI** パラメーターを使用していない場合は、WSDL 文書の最後にある <wsdl:service> エレメントと <wsdl:binding> エレメントを変更する必要があります。生成される WSDL には、これらの変更を行う際に役立つコメントが記載されています。これらのエレメントを変更するときに、Web サービス・バインディング・ファイルを再生成する必要はありません。
2. Web サービスのネットワークの場所を指定するには、DFHLS2WS で **URI** パラメーターを使用して、絶対 URI を設定します。**URI** パラメーターを使用していない場合は、wsdl:service エレメント内の soap:address に詳細を追加します。

- a. HTTP ベースのプロトコルを使用する場合は、*my-server* を CICS 領域の TCP/IP ホスト名に置き換えて、*my-port* を TCPIP SERVICE リソースのポート番号に置き換えます。
- b. WebSphere MQ をトランスポート・プロトコルとして使用する場合は、*myQueue* を適切なキューの名前に置き換えます。

これらの変更を行うために、Web サービス・バインディング・ファイルを変更する必要はありません。

WSBind ファイルを再生成せずにポート名と名前空間を変更する場合、実行時レベル 2.1 以降では間違ったモニター情報になる可能性があります。

3. WSDL 文書内の自動的に生成された名前が目的に合っているかどうかを検討します。以下の値を名前変更できます。

- WSDL 文書の `targetNamespace`
- WSDL 文書内の XML スキーマの `targetNamespace`
- `<wsdl:portType>` 名
- `<wsdl:operation>` 名
- `<wsdl:binding>` 名
- `<wsdl:service>` 名
- WSDL 文書内の XML スキーマにあるフィールドの名前

これらの値は、クライアント・プログラムをコーディングするプログラマチック・インターフェースの一部を形成します。生成された名前に十分な意味がない場合、長期間にわたるアプリケーション・コードの保守が困難になる可能性があります。DFHLS2WS の **REQUEST-NAMESPACE** および **RESPONSE-NAMESPACE** パラメーターを使って XML スキーマの `targetNamespace` を変更し、**WSDL-NAMESPACE** パラメーターを使って WSDL 文書の `targetNamespace` を変更してください。

これらの値のいずれかを変更する場合は、DFHWS2LS を使用して Web サービス・バインディング・ファイルを再生成する必要があります。生成される言語構造は、既存の言語構造と同じではありませんが、既存のアプリケーションとの互換性はあるため、アプリケーションを変更する必要はありません。ただし、新規の言語構造を無視して、元の構造を持つ新規の Web サービス・バインディング・ファイルを使用することができます。

4. XML スキーマで公開されている **COMMAREA** フィールドが適切かどうかを検討します。次のような、Web サービス・クライアント開発者にとって有益ではないフィールドを除去することを考慮することも必要です。
 - 出力値のためにのみ使用されるフィールドを、入力データ構造をマップするスキーマから除去することができます。
 - 充てん文字フィールド。
 - 自動的に生成される注釈

これらの変更を行う場合は、DFHWS2LS を使用して Web サービス・バインディング・ファイルを再生成する必要があります。生成される新規の言語構造には、元の言語構造との互換性がないため、データを新規の表現から古い表現にマップするためのラッパー・プログラムを作成する必要があります。このラッパ

ー・プログラムは、ターゲット・アプリケーション・プログラムに対して EXEC CICS LINK コマンドを実行してから、戻されたデータをマップする必要があります。

このレベルのカスタマイズでは最も多くの作業が必要ですが、Web サービス・クライアント開発者のプログラマチック・インターフェースが最も価値のあるものになります。

5. DFHWS2LS を介して、生成された WSDL 文書から新しい言語構造を作成する場合には、WSDL 文書内の注釈を保持するかどうかを決定してください。DFHWS2LS が言語構造を生成するとき、注釈は通常のマッピング規則を指定変更します。マッピング規則を指定変更する際には、生成された言語構造と DFHLS2WS で使用されたバージョンとの互換性があることを確認してください。デフォルトのマッピング規則を使って言語構造を生成したい場合には、注釈を除去してください。

タスクの結果

カスタマイズされた WSDL 文書を IBM WebSphere Service Registry and Repository (WSRR) サーバーに公開するには、WSRR インターフェースを使って手動で公開する必要があります。WSRR の詳細情報については、<http://www-01.ibm.com/software/integration/wsrr/>を参照してください。

例

WSDL 文書の例については、`.././com.ibm.cics.ts.exampleapplication.doc/topics/dfhxa_wsdlxample.dita`を参照してください。

SOAP 障害の送信

サービス・プロバイダーでは、CICS API を使用して、SOAP 障害を Web サービス・リクエスターに送信できます。この障害は、サービス・プロバイダー・アプリケーション、またはパイプラインのヘッダー処理プログラムのいずれかによって発行されます。

始める前に

API を使用するには、サービス・プロバイダー・アプリケーションは、チャンネルおよびコンテナを使用する必要があります。アプリケーションが COMMAREA を使用する場合、チャンネルおよびコンテナを使用して SOAP 障害メッセージを作成するラッパー・プログラムを記述します。CICS 提供の SOAP メッセージ・ハンドラーから直接 API が呼び出された場合、ヘッダー処理プログラムでのみ API を使用できます。

このタスクについて

ご使用のアプリケーション・ロジックでは要求を満たすことができない場合 (要求メッセージに根本的な問題が存在する場合など)、Web サービス・リクエスターに SOAP 障害を発行します。CICS は、SOAP 障害の発行をエラーとして見なさないため、エラー処理ではなく、標準的なメッセージ応答のパイプライン処理が実行されることに注意してください。いずれかのトランザクションをロールバックするには、アプリケーション・プログラムを使用する必要があります。

SOAP 障害を送信するには、以下の手順に従います。

1. プログラムで、EXEC CICS SOAPFAULT CREATE コマンドを使用して、SOAP 障害を送信します。SOAPFAULT CREATE を参照してください。
2. コマンドに CLIENT または SERVER オプションを追加します。このオプションは、クライアント側またはサーバー側のいずれかで、問題が発生したことを示します。
 - CLIENT は、受信された要求メッセージに問題があることを示します。
 - SERVER は、要求メッセージが CICS で処理されるときに問題が発生したことを示します。これは、アプリケーション・プログラムの問題である可能性があります (例えば、要求を満たすことができない)。あるいは、パイプライン処理中に発生する内在的な問題である可能性もあります。
3. サービス・プロバイダーによって障害が発行された理由を要約する
FAULTSTRING オプションと、その長さを表す FAULTSTRLEN オプションを追加します。このオプションの内容は、XML でなくてはなりません。アプリケーションが提供するすべてのデータは、XML 文書に直接含めるのに適した形式でなければなりません。アプリケーションは、いくつかの文字を XML エンティティとして指定する必要があるかもしれません。例えば、XML タグの始まり以外の場所で < 文字が使われる場合、アプリケーションはそれを < に変更する必要があります。以下の例は、間違った FAULTSTRING オプションを示しています。

```
dcl msg_faultString char(*) constant('Error: Value A < Value B');
```

この FAULTSTRING オプションを指定する正しい方法は、次のとおりです。

```
dcl msg_faultString char(*) constant('Error: Value A &lt; Value B');
```

ヒント: XML エンティティの使用を避けるために、XML CDATA 構成体でデータをラッパーすることができます。XML プロセッサは、この構成体の中の文字データを構文解析しません。この方式を使って、次のような FAULTSTRING オプションを指定できます。

```
dcl msg_faultString char(*) constant('<![CDATA[Error: Value A < Value B]]>');
```

4. サービス・プロバイダーによって障害が発行された理由を詳しく記述する
DETAIL オプションと、その長さを表す DETAILLENGTH オプションをコーディングします。このオプションの内容は、XML でなくてはなりません。
FAULTSTRING オプションと同じ指針が DETAIL オプションにも該当します。
5. オプション: ヘッダー処理プログラムから API を起動した場合は、パイプライン構成ファイルでそのプログラムを定義します。ヘッダー処理プログラムは、<cics_soap_1.1_handler> または <cics_soap_1.2_handler> エレメントのいずれかで定義されます。詳しくは、82 ページの『<cics_soap_1.1_handler>エレメント』 または 85 ページの『<cics_soap_1.2_handler>エレメント』を参照してください。

タスクの結果

プログラムがこのコマンドを発行した場合、CICS は、該当する SOAP レベルで SOAP 障害の応答メッセージを作成します。サービス・プロバイダー・アプリケーションがコマンドを発行した場合は、SOAP 応答を作成する必要はなく、コマンドを DFHRESPONSE コンテナに入れます。パイプラインは、メッセージ・ハンド

ラーによって SOAP 障害を処理し、その応答を Web サービス・プロバイダーに送信します。

例

SOAPFAULT CREATE コマンドには多くのオプションがあり、Web サービス・リクエスターに適切に応答できる柔軟性を提供します。次に示す簡単な例は、SOAP 障害を作成する、SOAP 1.1 および SOAP 1.2 の両方で使用できる完全なコマンドです。

```
EXEC CICS SOAPFAULT CREATE CLIENT
      DETAIL(msg_detail)
      DETAILLENGTH(length(msg_detail))
      FAULTSTRING(msg_faultString)
      FAULTSTRLLEN(length(msg_faultString));
```

ここで *msg_detail* および *msg_faultString* は、以下の値でコーディングされる可能性があります。

```
dc1 msg_detail char(*) constant('<ati:ExampleFault xmlns="http://www.example.org/faults"
xmlns:ati="http://www.example.org/faults">Detailed error message goes here.</ati:ExampleFault>');
dc1 msg_faultString char(*) constant('Something went wrong');
```

Web サービス・アシスタントを使用した Web サービス・リクエスターの作成

WSDL 1.1 または WSDL 2.0 に準拠する Web サービス記述から、サービス・リクエスター・アプリケーションを作成できます。CICS Web サービス・アシスタントは、サービス・リクエスターの設定における CICS アプリケーションの配置を支援します。

始める前に

Web サービス記述が z/OS UNIX のファイルに存在するか、IBM WebSphere Services Registry and Repository (WSRR) サーバー上に公開される必要があります。また、適切なリクエスター・モード・パイプラインが CICS 領域にインストールされている必要があります。

このタスクについて

CICS Web サービス・アシスタントを使用して、CICS アプリケーションをサービス・リクエスターとして配置する場合は、Web サービス記述から開始し、これを基に言語データ構造を生成する必要があります。Web サービス記述からサービス・リクエスター・アプリケーションを作成するには、以下の手順に従います。

1. Web サービス・バインディング・ファイルおよび 1 つ以上の言語構造を生成する場合は、DFHWS2LS パッチ・プログラムを使用します。Web サービス記述からサービス・リクエスター・アプリケーションを作成する場合、次のようなオプションを考慮してください。
 - a. 使用するマッピング・レベルは? マッピング・レベルが高いほど、実行時に文字とバイナリー・データの処理の制御とサポートが行いやすくなります。一部のオプション・パラメーターは、高いマッピング・レベルでしか使用できません。使用できる最高のマッピング・レベルを使用することをお勧めします。

- b. 実行時にデータを変換する際に使用するコード・ページは? CICS 領域のコード・ページと異なる特定のコード・ページをアプリケーションに使用する場合は、**CCSID** パラメーターを使用します。コード・ページは EBCDIC でなければならず、Java と z/OS の両方の変換サービスによってサポートされている必要があります。
- c. Web サービス記述で宣言された操作のサブセットをサポートしますか? Web サービス記述が非常に大きい場合、特定の数の操作だけをサービス・リクエスター・アプリケーションでサポートするには、**OPERATION** パラメーターを使って必要な操作をリストします。それぞれの操作はスペースで区切る必要があります、大/小文字が区別されます。
- d. WSDL 文書の保管場所は? DFHWS2LS への入力として使用する WSDL 文書が WSRR サーバーに格納されている場合、いくつかのパラメーターを指定して DFHWS2LS を実行することにより、これを取り出すことができます。**WSRR-SERVER** パラメーターを使って WSRR サーバーの場所を指定し、**WSRR-NAME** パラメーターを使用して、取り出す WSDL 文書の名前を指定します。WSRR との対話で使用できる DFHWS2LS の他のパラメーターについては、166 ページの『DFHWS2LS: WSDL から高水準言語への変換』を参照してください。
- e. WSDL 文書を WSRR サーバーから取り出す場合、セキュア接続を使用しますか? Web サービス・アシスタントで Secure Sockets Layer (SSL) 暗号化を使用して、安全に WSRR と相互運用することができます。例については、333 ページの『Web サービス・アシスタントと WSRR で SSL を使用方法の例』を参照してください。

DFHWS2LS の使用時には、**PROGRAM**、**URI**、**TRANSACTION**、および **USERID** などのパラメーターは指定しないでください。これらのパラメーターは、サービス・プロバイダー・アプリケーションのみに適用されます。指定すると、プロバイダー・モードの Web サービス・バインディング・ファイルが生成されます。Web サービス・バインディング・ファイルに加えて、このプログラムは言語データ構造を生成します。

- 2. ログ・ファイルを調べて、DFHWS2LS がバインディング・ファイルおよび言語構造を生成したときに問題が発生したかどうかを確認します。Web サービス記述には、CICS でサポートされないエレメントやオプションが含まれる可能性があります。警告またはエラー・メッセージが出される場合、記されているアドバイスを読んで適切な処置を行ってください。バッチ・プログラムを再実行しなければならないことがあります。
- 3. Web サービス・バインディング・ファイルを、Web サービス・アプリケーションに使用するリクエスター・モード・パイプラインのピックアップ・ディレクトリにコピーします。次のような目的で INQUIRE PIPELINE コマンドを使用します。
 - a. PIPELINE リソースがサービス・リクエスター・アプリケーションに対して構成されていることを確認します。**MODE** パラメーターの値は、インストールされたパイプラインがサポートするのがリクエスター Web サービス・アプリケーションかプロバイダー Web サービス・アプリケーションかを示します。
 - b. 正しい SOAP プロトコルが Web サービスのパイプラインでサポートされていることを確認します。**SOAPLEVEL** パラメーターは、サポートされるバ

ージョンを示します。サービス・リクエスター・モードでは、Web サービスのバインディングは、パイプラインでサポートされる SOAP のバージョンと一致している必要があります。SOAP 1.2 をサポートするサービス・リクエスター・パイプラインに SOAP 1.1 バインディングを使用する Web サービスをインストールすることはできません。

- c. パイプラインの構成済みタイムアウトがサービス・リクエスター・アプリケーションに適していることを確認します。タイムアウトは、PIPELINE リソースで RESPWAIT 属性の値として表示されます。パイプラインでタイムアウトが指定されていない場合は、トランスポートのデフォルトが使用されます。

- HTTP のデフォルト・タイムアウトは 10 秒です。
- WMQ のデフォルト・タイムアウトは 60 秒です。

CICS 領域内のトランザクションごとにディスパッチャー・タイムアウトがあります。この値がいずれかのプロトコルのデフォルトより小さい場合、ディスパッチャーによってタイムアウトになります。

4. オプション: Web サービス記述を、Web サービス・バインディング・ファイルと同じピックアップ・ディレクトリーにコピーします。これにより、実行時の Web サービスの検証をオンにすることができます。
5. ラッパー・プログラムを作成する場合は、ステップ 1 で生成した言語データ構造を使用します。Web サービスと通信する場合は、ラッパー・プログラムで EXEC CICS INVOKE WEBSERVICE コマンドを使用します。以下のオプションがコマンドに含まれます。
 - WEBSERVICE リソースの名前
 - Web サービスの呼び出し対象となる操作
6. RDO を使用して必要なリソースを作成することも可能ですが、PIPELINE SCAN コマンドを使用して、WEBSERVICE リソースと URIMAP リソースを動的に作成します。
 - WEBSERVICE リソースは、CICS で Web サービス・バインディング・ファイルをカプセル化し、実行時に使用されます。
 - URIMAP リソースは、WEBSERVICE リソースを特定の URI に関連付けるための情報を CICS に提供します。
7. コミュニケーション・ロジックを置換できるラッパー・プログラムを作成します。

タスクの結果

CICS リソースを正常に作成したら、サービス・リクエスター・アプリケーションの作成は完了です。

ツールを使用した Web サービスの作成

Web サービス・アシスタント JCL を使用する代わりに、Rational Developer for System z を使用するか、独自の Java プログラムを作成して、CICS で必要なファイルを作成することができます。

1. 以下の 2 つの選択肢があります。

- Rational Developer for System z ツールを使用して、Web サービス・バインディング・ファイル、および Web サービス記述または言語構造を作成します。このツールについて詳しくは、<http://www-306.ibm.com/software/awdtools/devzseries/> を参照してください。
- 提供されている API を使用して独自の Java プログラムを作成して、Web サービス・アシスタントを起動します。この API については、Web services assistant: Class Reference の Javadoc で説明されています。この資料には、クラスについて説明したコメントが記載されており、Web サービス・アシスタントの起動方法の例を示したサンプル・コードも提供されています。また、Javadoc には、必要な JAR ファイルと、z/OS UNIX でのそれらの場所を示す詳細なリストも含まれています。

Java プログラムは z/OS または Windows® プラットフォーム上で実行できます。Windows でこのプログラムを実行する場合は、FTP または同等のプロセスを使用して、生成済みの Web サービス・バインディング・ファイルをバイナリー・モードで適切なピックアップ・ディレクトリーに転送されます。

2. オプション: Web サービス記述を言語構造から生成する場合は、ファイルを検討し、必要なカスタマイズを実行します。詳しくは、238 ページの『生成した Web サービス記述文書のカスタマイズ』を参照してください。
3. 生成された Web サービス・バインディング・ファイルを、適切なパイプライン・ピックアップ・ディレクトリーに配置します。
4. オプション: Web サービス記述をパイプラインのピックアップ・ディレクトリーにコピーして、これによって、Web サービスの検証を実行して Web サービスが予想どおりに動作していることをチェックすることもできます。
5. Web サービス記述から始めた場合は、生成された言語構造とインターフェースをとるサービス・プロバイダーまたはリクエスターのアプリケーション・プログラムを作成します。
6. PIPELINE SCAN コマンドを実行して、必要な CICS リソースを自動的に作成します。

次のタスク

リソースが CICS 領域に正常にインストールされていることを確認してください。正常であれば、Web サービスのテストの準備ができています。

XML を認識する独自の Web サービス・アプリケーションの作成

CICS 提供のデータ・マッピングを使用しない場合、その代わりに、XML を認識するデータ・アプリケーションを 2 つの方法で独自に作成できます。DFHWS2LS で **XML-ONLY** パラメーターを使用するか、あるいはツールをまったく使用せずに独自にアプリケーションを作成することができます。 **XML-ONLY** パラメーターを使用する方法は、XML を処理対象データとしてアプリケーションに渡すよう CICS パイプライン・プロセスを構成する最も単純な方法です。

このタスクについて

XML を認識する独自のアプリケーションを作成するには、XML 文書を構文解析するコードと生成するコードを作成する必要があります。XML を認識する独自のア

アプリケーションを作成する 1 つの方法は、COBOL で XML PARSE および XML GENERATE ステートメントを使用することです。XML を認識する独自のアプリケーションを作成する別の方法として、他の IBM ツールを使用できます。例えば IBM Rational® Developer for System z® ツールを使用すると、アプリケーションから起動可能な COBOL XML コンバーター・プログラムを生成できます。

XML 認識サービス・プロバイダー・アプリケーション

XML を認識する独自のサービス・プロバイダー・アプリケーションは、渡されるコンテナと連携して、XML とプログラム言語の間のデータ変換を処理する必要があります。

このタスクについて

以下の手順に従うと、XML 認識アプリケーションを独自に作成できます。その際、いずれかの CICS ツールを使用するかどうかを決定できます。

1. DFHWS2LS を使って独自の XML 認識アプリケーション用の Web サービス・バインディング・ファイルを生成するかどうかを決定します。Web サービス・バインディング・ファイルを生成する利点は、検証などの CICS サービスを使用してグローバル・ユーザー出口を使用する Web サービスや CICS モニターをテストできることです。
 - Web サービス・バインディング・ファイルを生成するには、**XML-ONLY** パラメーター、および 2.1 以上の **MINIMUM-RUNTIME-LEVEL** を指定して DFHWS2LS を実行します。Web サービス・バインディング・ファイルにより、アプリケーション・プログラムは DFHWS-BODY コンテナの内容を直接処理できるようになります。これ以外のすべての点では、生成されるバインディング・ファイルの配置特性と実行時の動作は、**XML-ONLY** パラメーターなしで生成されるファイルと同じです。
 - Web サービス・バインディング・ファイルを使用しないことを決定した場合、Web サービス要求が独自の XML 認識アプリケーションに到達するようにサービス・プロバイダー・パイプラインを構成してください。独自の XML 認識アプリケーション・プログラムを使用するようにパイプライン構成ファイル内で端末ハンドラーを構成できます。または、パイプラインで受信される URI に応じて独自のアプリケーションに動的に切り替わるメッセージ・ハンドラーを作成することもできます。
2. 以下のコンテナで保持される Web サービス要求を処理するように、アプリケーションを作成します。

DFHWS-BODY

CICS 提供の SOAP メッセージ・ハンドラーがパイプラインに含まれる場合のインバウンド SOAP 要求に関する、SOAP 本体の内容。

DFHREQUEST

SOAP 要求のエンベロープを含むパイプラインから受信した完全な要求。

DFHWS-XMLNS

ネーム・スペースの接頭部を要求の XML の内容のためのネーム・スペースにマップする名前と値のペアのリスト。

DFHWS-SOAPACTION

コンテナ DFHWS-BODY 内の SOAP メッセージに関連付けられた SOAPAction ヘッダー。

API コマンドをコーディングしてコンテナで作業する場合に、CHANNEL オプションの指定はありません。コンテナはすべて現在のチャネル (プログラムに渡されたチャネル) に関連付けられているためです。チャネルの名前を知りたい場合は、EXEC CICS ASSIGN CHANNEL コマンドを使用します。

3. オプション: アプリケーションでは、パイプライン内のメッセージ・ハンドラーで使用可能な追加のコンテナを使用することもできます。また、メッセージ・ハンドラーの処理時に作成される他の任意のコンテナを使用することもできます。コンテナの詳細なリストについては、125 ページの『パイプラインで使用するコンテナ』を参照してください。
4. アプリケーションが要求を処理したら、以下のコンテナを使用して、Web サービス応答を構成します。

DFHRESPONSE

パイプラインに渡される完全な応答メッセージ。メッセージに SOAP を使用しない場合、またはプログラム内でエンベロープを含む完全な SOAP メッセージを作成する場合、このコンテナを使用します。

コンテナ DFHWS-BODY に SOAP 本体を指定する場合、DFHRESPONSE を空にする必要があります。DFHWS-BODY および DFHRESPONSE の両方に内容を指定した場合、CICS は DFHRESPONSE を使用します。

DFHWS-BODY

アウトバウンド SOAP 応答の場合は、SOAP 本体の内容。パイプラインの端末ハンドラーが CICS 提供の SOAP メッセージ・ハンドラーである場合、このコンテナを指定します。メッセージ・ハンドラーは、本体を含む完全な SOAP メッセージを作成します。

要求と応答が同一であっても、プログラムでこのコンテナを作成する必要があります。作成しないと、CICS が内部サーバー・エラーを発行します。

この他のいずれかのコンテナを使用して、アウトバウンド応答を処理するためにパイプラインが必要とする情報を渡すこともできます。

Web サービスが応答を戻さない場合は、応答がないことを示すコンテナ DFHNORESPONSE を戻す必要があります。メッセージ・ハンドラーはコンテナが存在するかどうかのみをチェックするため、コンテナの内容は重要ではありません。

5. URIMAP 応答を作成します。XML-ONLY パラメーターを使用し、しかも DFHWS2LS の URI パラメーターの値を指定した場合には、PIPELINE SCAN 処理中に URIMAP が自動的に作成されます。

XML 認識サービス・リクエスター・アプリケーションの作成

XML を認識する独自の Web サービス・リクエスター・アプリケーションは、XML とプログラミング言語の間のデータ変換を処理し、パイプライン内の制御コンテナにデータを設定します。

始める前に

DFHWS2LS で **XML-ONLY** パラメーターを使用することにより、独自の XML 認識サービス・リクエスター・アプリケーションを作成できます。または、ツールをまったく使用せずに作成することもできます。独自の XML 認識サービス・リクエスター・アプリケーションを作成する最も単純な方法は、DFHWS2LS で **XML-ONLY** パラメーターを使用することです。 **XML-ONLY** パラメーターは実行時レベル 2.1 以上で使用できます。

このタスクについて

XML-ONLY パラメーターを使用した結果として生成される WSBind ファイルは、アプリケーションが DFHWS-BODY コンテナの内容を直接処理することを CICS に対して示します。リクエスター・モードの **WEBSERVICE** リソースを作成するために、生成された WSBind ファイルはリクエスター・モードの **PIPELINE** にインストールされる必要があります。アプリケーションは Web サービス要求の本体用の XML を生成して、それを DFHWS-BODY コンテナに格納する必要があります。その後、EXEC CICS INVOKE SERVICE コマンドを呼び出す必要があります。アウトバウンド・メッセージが Web サービス・プロバイダーに送られます。呼び出しが完了した後、応答メッセージの本文もまた DFHWS-BODY コンテナに入ります。

XML を認識するリクエスター・アプリケーションは、リモート・プロバイダー・モード・アプリケーションから戻される SOAP 障害メッセージを受け取ることがあります。この場合、リクエスター・アプリケーションは SOAP 障害を解釈して、それを通常の応答メッセージと区別する必要があります。INVOKE SERVICE コマンドを **XML-ONLY WEBSERVICE** と共に使用する場合、CICS は、SOAP 障害の受信を示すために通常使われる応答コードを設定しません。

XML-ONLY オプションを使用せずに独自の XML 認識サービス・リクエスター・アプリケーションを作成する場合は、以下の手順を完了してください。

1. チャンネルを作成して、チャンネルにコンテナを取り込みます。各コンテナに次の情報を指定します。

DFHWS-PIPELINE

アウトバウンド要求に使用される **PIPELINE** リソースの名前。

DFHWS-URI

ターゲット Web サービスの URI。

DFHWS-BODY

アウトバウンド SOAP 要求の場合は、SOAP 本体の内容。パイプラインが CICS 提供の SOAP メッセージ・ハンドラーを含むとき、このコンテナを指定します。メッセージ・ハンドラーは、本体を含む完全な SOAP メッセージを作成します。

DFHREQUEST

パイプラインに渡される完全な要求メッセージ。メッセージに SOAP を使用しない場合、またはプログラム内でエンベロープを含む完全な SOAP メッセージを作成する場合、このコンテナを使用します。アウトバウンド・メッセージで SOAP ヘッダーが重複して送られるのを防ぐために、CICS 提供の SOAP メッセージ・ハンドラーがパイプラインに含まれてはなりません。

コンテナ DFHWS-BODY に SOAP 本体を指定する場合、DFHREQUEST を空にする必要があります。DFHWS-BODY および DFHREQUEST の両方に内容を指定した場合、CICS は DFHREQUEST を使用します。

DFHWS-XMLNS

ネーム・スペースの接頭部を要求の XML の内容のためのネーム・スペースにマップする名前と値のペアのリスト。

DFHWS-SOAPACTION

コンテナ DFHWS-BODY に指定された SOAP メッセージに追加される SOAPAction ヘッダー。

ヒント: コンテナ DFHWS-NOABEND をチャンネルに追加すると、DFHPIRT の呼び出し時に DFHPIRT 内部から異常終了が発行されることはありません。DFHERROR コンテナを介してエラーを処理できるため、C/C++ プログラムを実行する場合にはこれが役立ちます。

2. プログラム DFHPIRT にリンクします。次のコマンドを使用します。

```
EXEC CICS LINK PROGRAM(DFHPIRT) CHANNEL(userchannel)
```

ここで、*userchannel* はコンテナを含むチャンネルです。アウトバウンド・メッセージは、パイプライン内のメッセージ・ハンドラーおよびヘッダー処理プログラムによって処理され、Web サービス・プロバイダーに送られます。

3. 同じチャンネルからの Web サービス応答を格納するコンテナを取り出します。Web サービス・プロバイダーからの応答は、成功した応答か SOAP 障害の可能性もあります。Web サービス・リクエスター・アプリケーションは、サービス・プロバイダーからのいずれのタイプの応答も処理できる必要があります。次のコンテナに完全な応答が格納されます。

DFHRESPONSE

SOAP 応答のエンベロープを含む Web サービス・プロバイダーから受信した完全な応答。

DFHWS-BODY

パイプラインが CICS 提供の SOAP メッセージ・ハンドラーを含む場合は、SOAP 本体の内容。

DFHERROR

パイプラインからのエラー情報。

SOAP メッセージの検証

CICS Web サービス・アシスタントを使用してアプリケーションを配置する場合は、Web サービス記述に格納されているスキーマに SOAP メッセージが適合していることを確認するために、SOAP メッセージを実行時に検証することを指定できます。検証は、プロバイダー・モードとリクエスター・モードの両方で実行できます。

始める前に

Web サービス配置の開発中およびテスト時に完全な検証を実行すると、サービス・リクエスターとサービス・プロバイダー間のメッセージ交換の問題を検出するうえで役立ちます。ただし、SOAP メッセージの完全な検証によってかなりのオーバーヘッドが生じるため、完全にテストされる実動アプリケーションでメッセージを検証することはお勧めできません。

CICS では、Java プログラムを使用して SOAP メッセージを検証します。したがって、検証を実行するために、CICS 領域で Java サポートを使用可能にしておく必要があります。

このタスクについて

SOAP メッセージは、アプリケーション・データ構造に変換される前、およびアプリケーション・データ構造から SOAP メッセージが生成されるときに検証されます。SOAP メッセージは WSDL で XML スキーマを使用して検証され、CICS の変換要件に照らして再び検証されます。

検証をオフにすると、CICS は Java プログラムを使用しません。CICS が SOAP メッセージを検証する範囲は、メッセージが適切な形式の XML を含むことを確認し、メッセージを変換するために必要な範囲に限定されます。したがって、WSDL を使用して SOAP メッセージが正常に検証されても、実行時環境では失敗する可能性があります、その逆も起こり得ます。

SOAP メッセージを検証するには、以下のステップを実行します。

1. Web サービス記述を WEBSERVICE リソースに関連付けてあることを確認します。この関連付けが適用されるのは、パイプラインのピックアップ・ディレクトリーのスキャン時に Web サービス記述がそのディレクトリーに存在していた場合に自動作成される WEBSERVICE リソース定義です。

RDO で作成された WEBSERVICE 定義の場合、Web サービス記述は WSDLFILE 属性で指定されます。

2. Web サービスの検証をオンにします。次の CEMT コマンドまたは SPI コマンド: SET WEBSERVICE(name) VALIDATION を使用します。RDO で定義されている Web サービスでは、検証が必要なを VALIDATION 属性で指定できますが、この設定は、Web サービスのインストール後に、SET WEBSERVICE コマンドを使用すると変更できます。

タスクの結果

システム・ログを確認して、SOAP メッセージが有効かどうかを調べます。メッセージ DFHPI1002 は、SOAP メッセージが正常に検証されたことを示し、メッセー

ジ DFHPI1001 は、検証が失敗したことを示します。

次のタスク

Web サービスの検証が必要なくなったら、次のコマンド: SET WEBSERVICE(*name*) NOVALIDATION を使用して検証をオフにします。

第 9 章 Web サービスのための実行時処理

Web サービス・プロバイダーに要求を送信したり、Web サービス・リクエスターから要求を受信したりするには、アプリケーション (またはラッパー・プログラム) が CICS の Web サービス・サポートと正しく対話する必要があります。また、インバウンド/アウトバウンド要求の処理方法を決定するためにパイプラインで実行される処理を制御することもできます。

CICS による、Web サービス・アシスタントを使用して配置したサービス・プロバイダー・プログラムの呼び出し方法

CICS Web サービス・アシスタントを使用して配置したサービス・プロバイダー・アプリケーションが呼び出されると、CICS は COMMAREA またはチャンネルを使用して、そのサービス・プロバイダー・アプリケーションにリンクします。

JCL プロシージャ DFHWS2LS または DFHLS2WS を PGMINT パラメーターを指定して実行したときに使用されるインターフェースの種類を指定します。チャンネルを指定する場合は、**CONTID** パラメーターでコンテナの名前を指定できます。

- プログラムが COMMAREA インターフェースで呼び出される場合、COMMAREA には CICS が SOAP 要求から作成した最上位のデータ構造が格納されます。
- プログラムがチャンネル・インターフェースで呼び出される場合は、DFHWS2LS または DFHLS2WS の **CONTID** パラメーターに指定されたプログラムのコンテナに最上位のデータ構造が渡されます。**CONTID** パラメーターを指定しなかった場合、データはコンテナ DFHWS-DATA に渡されます。チャンネル・インターフェースでは、一続きのコンテナ内にある結合された一続きのデータ構造として表現される、エレメント数が変化する配列をサポートします。これらのコンテナも存在します。

API コマンドをコーディングしてコンテナで作業する場合に、CHANNEL オプションを指定する必要はありません。コンテナはすべて現在のチャンネル (プログラムに渡されたチャンネル) に関連付けられているためです。チャンネルの名前を知りたい場合は、EXEC CICS ASSIGN CHANNEL コマンドを使用します。

プログラムは要求の処理を終了すると、同じメカニズムを使用して応答を戻す必要があります。要求が COMMAREA で受信されている場合は、応答を COMMAREA に戻す必要があります。要求がコンテナで受信されている場合は、応答を同じコンテナに戻す必要があります。

アプリケーション・プログラムが応答メッセージを出す際にエラーになる場合は、アプリケーションが同期点を実行していない限り、CICS が変更をすべてロールバックします。

プログラムが提供する Web サービスが応答を戻す設計になっていない場合、プログラムが応答を戻しても CICS は COMMAREA またはコンテナ内の情報を無視します。

Web サービス・アシスタントを使用して配置したアプリケーションからの Web サービスの呼び出し

Web サービス・アシスタントを使用して配置したサービス・リクエスター・アプリケーションは、EXEC CICS INVOKE SERVICE コマンドを使用して Web サービスを呼び出します。要求と応答がコンテナ DFHWS-DATA のデータ構造にマップされます。

1. チャンネルを作成して、チャンネルにコンテナを取り込みます。少なくとも、コンテナ DFHWS-DATA が存在していることが必要です。このコンテナは、CICS が SOAP 要求に変換する最上位のデータ構造を格納します。SOAP 要求にエレメント数が増える配列が含まれている場合、このような配列は、一続きのコンテナ内の結合された一続きのデータ構造として表現されます。チャンネル内にこれらのコンテナも存在することが必要です。
2. ターゲット Web サービスを呼び出します。次のコマンドを使用します。

```
EXEC CICS INVOKE SERVICE(webservice)  
                      CHANNEL(userchannel)  
                      OPERATION(operation)
```

ここで、

webservice は、呼び出す Web サービスを定義する WEBSERVICE リソースの名前です。WEBSERVICE リソースは、Web サービス記述の場所を指定し、CICS が Web サービスと通信するときに使用する Web サービス・バインディング・ファイルの場所を指定します。

userchannel は、コンテナ DFHWS-DATA およびアプリケーションのデータ構造に関連付けられているその他のコンテナを持つチャンネルです。

operation は、ターゲット Web サービスで呼び出す操作の名前です。

URI(*uri*) を指定することもできます。*uri* は、呼び出す Web サービスの URI です。このオプションを省略する場合は、WEBSERVICE リソース定義に関連付けられている Web サービス・バインディング・ファイルにプロバイダーの URI (DFHWS2LS により Web サービス記述から取得される) またはプロバイダーのアプリケーション名 (DFHWS2LS の PGMNAME 入力パラメーターとして指定される) のいずれかが含まれていることが必要です。

WEBSERVICE リソースに関連付けられた Web サービス・バインディング・ファイルのプロバイダー・アプリケーション名は、Web サービス要求のローカルでの最適化を可能にする目的で使用されます。この最適化を使用すると、EXEC CICS INVOKE SERVICE コマンドが EXEC CICS LINK コマンドに最適化されます。この最適化は、Web サービスからの応答の送信が期待できない場合の EXEC CICS INVOKE SERVICE コマンドの動作に次のような影響があります。

- 最適化が行われない場合、要求メッセージが送信されるとすぐに EXEC CICS INVOKE SERVICE コマンドから制御が戻ります。

- 最適化が行われる場合は、ターゲット・プログラムが戻る場合にのみ EXEC CICS INVOKE SERVICE コマンドから制御が戻ります。

Web サービスからの応答の送信が期待できる場合は、応答が提供可能であるとき、コマンドから制御が戻ります。

3. コマンドが正常に実行された場合は、チャネルから応答コンテナを取り出します。少なくとも、コンテナ DFHWS-DATA は存在します。このコンテナは、CICS が SOAP 応答から作成した最上位のデータ構造を格納します。応答にエレメント数が増える配列が含まれている場合、このような配列は、一続きのコンテナ内の結合された一続きのデータ構造として表現されます。チャネル内にこれらのコンテナも存在します。
4. サービス・リクエスターが、呼び出される Web サービスから SOAP 障害メッセージを受け取る場合は、アプリケーション・プログラムで変更をロールバックするべきかを、決定する必要があります。SOAP 障害の場合、RESP2 値が 6 である INVREQ エラーがアプリケーション・プログラムに戻されます。ただし、最適化が有効であれば、リクエスターとプロバイダーの両方で同じトランザクションが使用されます。ローカルで最適化された Web サービス・プロバイダーでエラーが起こる場合は、このトランザクションによって行われたすべての処理が、プロバイダーとリクエスターの両方でロールバックされます。RESP2 値が 16 である INVREQ エラーがリクエスターに戻されます。

Web サービス・アシスタントによって生成されるコードの実行時の制限

CICS は実行時に、Web サービス記述 (WSDL) に準拠する有効な SOAP メッセージのほとんどすべてを、同等のデータ構造に変換できます。ただし、サービス・リクエスターまたはサービス・プロバイダーのアプリケーションを、Web サービス・アシスタントのバッチ・ジョブを使用して開発する際は、いくつか制限があります。

コード・ページ

CICS は、コード・ページを識別する適切な HTTP または WMQ ヘッダーがあれば、どのようなコード・ページで送信される SOAP メッセージでもサポートします。CICS は、アプリケーション・プログラムで必要とされるコード・ページに変換する前に、パイプラインで処理するために、SOAP メッセージを UTF-8 に変換します。パフォーマンス・インパクトを最小化するには、SOAP メッセージを CICS に送信する際に、UTF-8 コード・ページの使用をお勧めします。CICS では常に SOAP メッセージを UTF-8 で送信します。

CICS は、SOAP メッセージとアプリケーション・プログラムの間のデータの変換に使用されるコード・ページが EBCDIC である場合にのみ、SOAP メッセージを変換できます。UTF-8、ASCII および ISO8859-1 などのコード・ページでデータがエンコードされることを予期するアプリケーションはサポートされません。データ構造および SOAP メッセージで DBCS 文字を使用したい場合は、DBCS をサポートするコード・ページを指定する必要があります。ユーザーが選択する EBCDIC コード・ページは、Java および z/OS 両方の変換サービスによってもサポートされていなければなりません。また、z/OS 変換サービスは、SOAP メッセージのコード・ページから UTF-8 への変換をサポートするように構成されている必要があります。

適切なコード・ページを設定するには、**LOCALCCSID** システム初期設定パラメーターを使用するか、Web サービス・アシスタントのジョブでオプションの **CCSID** パラメーターを使用します。**CCSID** パラメーターを使用する場合、ユーザーが指定する値が、その特定の Web サービスについての **LOCALCCSID** コード・ページを指定変更します。**CCSID** パラメーターを指定しない場合は、データの変換には **LOCALCCSID** コード・ページが使用され、Web サービス・バインディング・ファイルが US EBCDIC (Cp037) でエンコードされます。

コンテナ

サービス・プロバイダー・モードでは、**PGMINT** パラメーターに値 **CHANNEL** を指定する場合、アプリケーション・データを保持するコンテナがバイナリー・モードで書き込みおよび読み取りを行う必要があります。このコンテナはデフォルトでは **DFHWS-DATA** です。**PUT CONTAINER** コマンドで **DATATYPE** オプションを **BIT** に設定するか、**FROMCCSID** オプションを除外して **BIT** をデフォルトのままにする必要があります。例えば次のコードは、現在のチャンネルにあるコンテナ **CUSTOMER-RECORD** 内にあるデータを、バイナリー・モードで書き込む必要があることを明確に示しています。

```
EXEC CICS PUT CONTAINER (CUSTOMER-RECORD)
              FROM (CREC)
              DATATYPE (BIT)
```

コンテナ自身がすべて **BIT** モードであっても、このデータをマップする言語構造内のすべてのテキスト・フィールドで、EBCDIC コード・ページ (**LOCALCCSID** または **CCSID** パラメーターに指定したものと同一コード・ページ) を使用する必要があります。Web サービス・バインディング・ファイルの生成に **DFHWS2LS** を使用する場合は、完全なデータ構造の一部を保持するコンテナがチャンネル上に多数あるかもしれません。この場合、これらの各コンテナのテキスト・フィールドは、同じコード・ページを使用して読み取りおよび書き込みを行う必要があります。

アプリケーション・プログラムが、**SOAP** メッセージに変換されるコンテナを組み込む場合は、アプリケーションがコンテナに適切な量のデータが含まれるか判断します。コンテナが保持するデータが予想より少ない場合は、**CICS** が変換エラーを出します。

アプリケーション・プログラムが **INVOKE SERVICE** コマンドを使用する場合、**CICS** に渡される任意のコンテナが再利用され、その中のデータが置き換えられる可能性があります。これらのコンテナ内のデータを保持したい場合には、プログラムを実行する前に、新しいチャンネルを作成してコンテナをそれにコピーしてください。リクエスター・モードの Web サービスでもあるプロバイダー・モードの Web サービスがある場合は、**INVOKE SERVICE** コマンドを使用する際に、最初に接続されていたデフォルトのチャンネルを使用するのではなく、別のチャンネルを使用することをお勧めします。アプリケーション・プログラムが **INVOKE SERVICE** コマンドを何度も使用する場合は、**CICS** を呼び出すたびに異なるチャンネルを使用するか、最初の要求における重要なデータをすべて保管してから、2 番目の要求を行うようにしてください。

Web サービス記述への準拠

Web サービス記述では、オプションで、考えられる SOAP メッセージの内容の一部を記述できます。この場合、DFHWS2LS が、生成された言語構造内でフィールドを割り振り、内容があるかどうかを示します。CICS は実行時にこれらのフィールドを適宜設定します。例えば存在フラグまたはオカレンス・フィールドなどのフィールドが、情報がないことを示す場合、アプリケーション・プログラムは、そのオプションの内容に関連付けられたフィールドを処理しません。

CICS が SOAP メッセージを変換する際に、SOAP メッセージの内容が一部欠落している場合、データ構造内のこれに相当するフィールドは、アプリケーション・プログラムに渡されるときに初期化されません。

Web サービス記述では、SOAP メッセージを読み取る際に使用する、空白文字の処理規則も指定できます。CICS は実行時にこの規則を実装します。

- `xsd:whiteSpace` ファセットの値が `replace` である場合、「タブ」および「復帰」などの空白文字はスペースで置き換えられます。
- `xsd:whiteSpace` ファセットの値が `collapse` である場合、先行または末尾の空白文字は、SOAP メッセージを生成する際に除去されます。

SOAP メッセージ

CICS は、SOAP メッセージの内容の導出をサポートしていません。例えば、SOAP メッセージは、`xsi:type` 属性を使用してエレメントに特定のタイプを指定し、併せて `xsi:schemaLocation` 属性でエレメントを記述するスキーマのロケーションを指定できます。CICS は、動的にスキーマを取得し、スキーマの内容に基づいてエレメントの値を変換する機能をサポートしていません。CICS は、Web サービス・アシスタントに設定されたマッピング・レベルが 1.1 以上であるときに `xsi:nil` 属性をサポートしますが、これはサポートされる唯一の XML スキーマ・インスタンス属性です。

DFHWS2LS は、SOAP メッセージ内のいくつかの値について、最大長や最大サイズを推測する必要があるかもしれません。例えば、XML スキーマが `xsd:string` の最大長を指定しない場合、DFHWS2LS は最大長を 255 文字と推測し、これに応じて言語構造を生成します。この値は、DFHWS2LS で **DEFAULT-CHAR-MAXLENGTH** パラメーターを使用して変更できます。CICS が実行時に、言語構成に割り振られたスペースより大きい値を持つ SOAP メッセージを検出すると、CICS は変換エラーを出します。

CICS がサービス・プロバイダーである場合は、SOAP 障害メッセージがリクエスターに戻されます。CICS がサービス・リクエスターである場合は、適切な RESP2 コードが `INVOKE SERVICE` コマンドから戻されます。

XML では、< および > 文字のように、特殊な意味を持つ文字があります。実行時に CICS が処理する文字配列内にこのような特殊文字が出現する場合は、同等のエンティティーで置き換えられます。サポートされる XML エンティティーは、次のとおりです。

文字	XML エンティティー
&	&

文字	XML エンティティ
<	<
>	>
"	"
'	'

CICS は、空白文字コードに使用される数字参照の正規形式もサポートします。

文字	XML エンティティ
タブ		
復帰	

改行	

このサポートは、呼び出されるどのパイプライン・ハンドラー・プログラムにも拡張されないことに注意してください。

ヌル文字 (x'00') は、どの XML 文書でも無効です。アプリケーション・プログラムに備えられた文字タイプ・フィールドがヌル文字を含む場合は、CICS がその時点でデータを切り捨てます。このため、文字配列をヌル終了ストリングとして処理できます。DFHWS2LS によって base64Binary または hexBinary の XML スキーマのデータ・タイプから生成される文字タイプのフィールドは、バイナリー・データを表示します。このフィールドは、ヌル文字を切り捨てずに含むことがあります。

SOAP 障害メッセージ

CICS がサービス・プロバイダーであり、アプリケーション・プログラムで SOAP 障害メッセージを出したい場合は、SOAPFAULT CREATE コマンドを使用します。この API コマンドを使用するには、Web サービス・アシスタントの **PGMINT** パラメーターに値 **CHANNEL** を指定する必要があります。この値を指定せずに、アプリケーション・プログラムが SOAPFAULT CREATE コマンドを呼び出すと、CICS が SOAP 応答メッセージを生成しようとしません。

リクエスター・パイプライン処理を制御するためのオプション

サービス・リクエスター・パイプラインにおいて、メッセージ・ハンドラーは URI を変更することによって Web サービス要求が送られる場所を決定できます。CICS はさまざまな URI 形式をサポートするため、パイプラインで Web サービス要求が処理される方法をかなり柔軟に制御することができます。

サービス・リクエスター・パイプラインの処理の終わりに達したとき、次のようなオプションの中から選ぶことができます。

プログラムにリンクする

`cics://PROGRAM/program` (*program* はターゲット・アプリケーション・プログラムの名前) という形式に URI を変更した場合、CICS は EXEC CICS LINK コマンドを使用して現在のチャンネルとそのコンテナ (または COMMAREA) をプログラムに渡します。

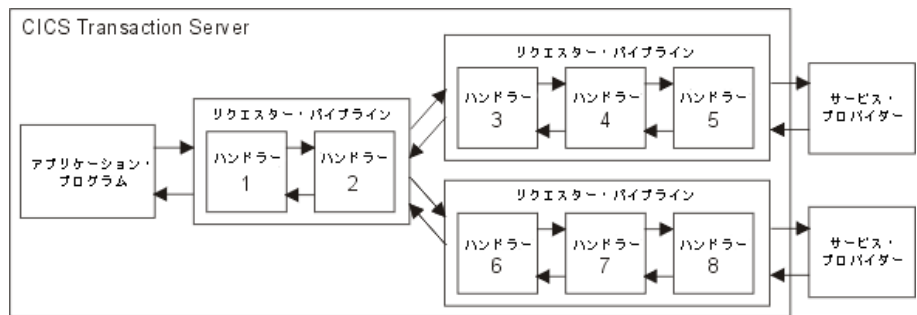
この処理は、サービス・リクエスターとサービス・プロバイダー・アプリケーションが同じ CICS 領域に属する場合に行われるローカル最適化に似て

います。しかし、この URI 形式を使用すると、パイプラインと任意のカスタム・メッセージ・ハンドラーを介して最初に要求が実行されるという利点があります。ターゲット・アプリケーション・プログラムは、コンテナまたは COMMAREA の内容を処理する必要があります。

別のリクエスター・モード・パイプラインを開始する

`cics://PIPELINE/pipeline?targetServiceUri=targetServiceUri` (`pipeline` は PIPELINE リソースの名前、`targetServiceUri` は DFHWS-URI コンテナに入れる URI) という形式に URI を変更した場合、CICS は指定されたリクエスター・パイプラインに現在のチャンネルとそのコンテナを渡します。要求をサービス・プロバイダーに送る前に、複数のリクエスター・パイプラインを互いにリンクするには、この URI を使用してください。連結できるリクエスター・パイプラインの数には制限がありません。

以下の例では、1 つの汎用リクエスター・パイプラインが 1 つのアプリケーションをサポートしています。メッセージ・ハンドラー 1 または 2 は、コンテナ内のアプリケーション・データに応じてそれぞれの要求の URI を変更することができ、異なるメッセージ・ハンドラーを含む 2 つのリクエスター・パイプラインのいずれかに要求を送ります。



この例は 1 つのサービス・リクエスター・アプリケーションだけを示していますが、多数のアプリケーションで同じ汎用リクエスター・パイプラインを使用して、要求が適切な Web サービス・プロバイダーに最終的に送られる前に、異なるリクエスター・パイプラインに要求を送ることもできます。

プロバイダー・モード・パイプラインに要求を直接送信する

`cics://SERVICE/service?targetServiceUri=targetServiceUri` (`service` はターゲット・サービスの名前、`targetServiceUri` はサービスのパス) という形式に URI を変更した場合、CICS はパスを URIMAP にマッチングすることによって要求を解決し、正しいプロバイダー・パイプラインに要求を渡します。ネットワークを使わずにリクエスター・パイプラインとプロバイダー・パイプラインの両方を介して要求を処理したい場合には、このオプションを使用してください。

また、リクエスター・アプリケーションとプロバイダー・アプリケーションが異なる言語で書かれ (または異なるマッピング・レベルを使用し)、異なるバイナリー・データを想定しているような場合にも、この URI が役立つでしょう。

関連タスク

261 ページの『URI を使用したリクエスター・パイプライン処理の制御』
サービス・リクエスター・パイプラインにおいて、メッセージ・ハンドラーは

URI を変更することにより Web サービス要求をどこに送るかを決定できます。URI 形式を変更すると、別のリクエスター・パイプラインを開始したり、ネットワークを介して要求を送信せずにサービス・プロバイダー・パイプラインを開始するなどの最適化を実行できます。

関連資料

138 ページの『DFHWS-URI コンテナ』

DFHWS-URI は、サービスの URI を格納する、DATATYPE(CHAR) のコンテナです。

リクエスター・パイプライン処理を制御するためのオプション

サービス・リクエスター・パイプラインにおいて、メッセージ・ハンドラーは URI を変更することによって Web サービス要求が送られる場所を決定できます。CICS はさまざまな URI 形式をサポートするため、パイプラインで Web サービス要求が処理される方法をかなり柔軟に制御することができます。

サービス・リクエスター・パイプラインの処理の終わりに達したとき、次のようなオプションの中から選ぶことができます。

プログラムにリンクする

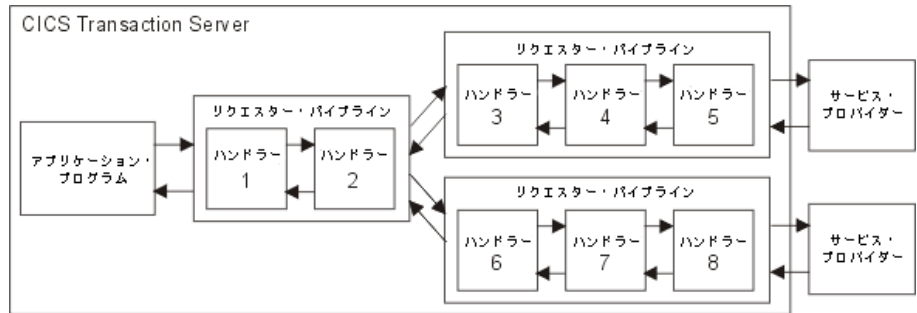
`cics://PROGRAM/program` (*program* はターゲット・アプリケーション・プログラムの名前) という形式に URI を変更した場合、CICS は EXEC CICS LINK コマンドを使用して現在のチャンネルとそのコンテナ (または COMMAREA) をプログラムに渡します。

この処理は、サービス・リクエスターとサービス・プロバイダー・アプリケーションが同じ CICS 領域に属する場合に行われるローカル最適化に似ています。しかし、この URI 形式を使用すると、パイプラインと任意のカスタム・メッセージ・ハンドラーを介して最初に要求が実行されるという利点があります。ターゲット・アプリケーション・プログラムは、コンテナまたは COMMAREA の内容を処理できる必要があります。

別のリクエスター・モード・パイプラインを開始する

`cics://PIPELINE/pipeline?targetServiceUri=targetServiceUri` (*pipeline* は PIPELINE リソースの名前、*targetServiceUri* は DFHWS-URI コンテナに入れる URI) という形式に URI を変更した場合、CICS は指定されたリクエスター・パイプラインに現在のチャンネルとそのコンテナを渡します。要求をサービス・プロバイダーに送る前に、複数のリクエスター・パイプラインを互いにリンクするには、この URI を使用してください。連結できるリクエスター・パイプラインの数には制限がありません。

以下の例では、1 つの汎用リクエスター・パイプラインが 1 つのアプリケーションをサポートしています。メッセージ・ハンドラー 1 または 2 は、コンテナ内のアプリケーション・データに応じてそれぞれの要求の URI を変更することができ、異なるメッセージ・ハンドラーを含む 2 つのリクエスター・パイプラインのいずれかに要求を送ります。



この例は 1 つのサービス・リクエスター・アプリケーションだけを示していますが、多数のアプリケーションで同じ汎用リクエスター・パイプラインを使用して、要求が適切な Web サービス・プロバイダーに最終的に送られる前に、異なるリクエスター・パイプラインに要求を送ることもできます。

プロバイダー・モード・パイプラインに要求を直接送信する

`cics://SERVICE/service?targetServiceUri=targetServiceUri` (*service* はターゲット・サービスの名前、*targetServiceUri* はサービスのパス) という形式に URI を変更した場合、CICS はパスを URIMAP にマッチングすることによって要求を解決し、正しいプロバイダー・パイプラインに要求を渡します。ネットワークを使わずにリクエスター・パイプラインとプロバイダー・パイプラインの両方を介して要求を処理したい場合には、このオプションを使用してください。

また、リクエスター・アプリケーションとプロバイダー・アプリケーションが異なる言語で書かれ (または異なるマッピング・レベルを使用し)、異なるバイナリー・データを想定しているような場合にも、この URI が役立つでしょう。

関連タスク

『URI を使用したリクエスター・パイプライン処理の制御』

サービス・リクエスター・パイプラインにおいて、メッセージ・ハンドラーは URI を変更することにより Web サービス要求をどこに送るかを決定できます。URI 形式を変更すると、別のリクエスター・パイプラインを開始したり、ネットワークを介して要求を送信せずにサービス・プロバイダー・パイプラインを開始するなどの最適化を実行できます。

関連資料

138 ページの『DFHWS-URI コンテナ』

DFHWS-URI は、サービスの URI を格納する、DATATYPE(CHAR) のコンテナです。

URI を使用したリクエスター・パイプライン処理の制御

サービス・リクエスター・パイプラインにおいて、メッセージ・ハンドラーは URI を変更することにより Web サービス要求をどこに送るかを決定できます。URI 形式を変更すると、別のリクエスター・パイプラインを開始したり、ネットワークを介して要求を送信せずにサービス・プロバイダー・パイプラインを開始するなどの最適化を実行できます。

始める前に

リクエスター・パイプラインにどのオプションを実装するかを決定します。詳しくは、258 ページの『リクエスター・パイプライン処理を制御するためのオプション』を参照してください。

このタスクについて

Web サービス・リクエスター・アプリケーションは EXEC CICS INVOKE SERVICE コマンドを使って DFHWS-URI コンテナのデータを設定できます。アプリケーションによって値が提供されない場合、CICS は Web サービス・バインディング・ファイル内の値を使ってコンテナのデータを設定します。URI を変更するには、このコンテナの内容を変更するメッセージ・ハンドラーを作成する必要があります。

1. 以下のいずれかの URI 形式に従って、DFHWS-URI コンテナを変更するメッセージ・ハンドラーを作成します。

- アプリケーション・プログラムにリンクするには、`cics://PROGRAM/program` という URI を使用します (*program* はターゲット・アプリケーション・プログラム)。データ変換は行われないため、アプリケーション・プログラムで現在のチャンネル上のコンテナの内容を処理できるようにする必要があります。アプリケーション・プログラムは、現在のチャンネルとコンテナ、あるいは COMMAREA を渡すことができます。
- ネットワークを通さずにプロバイダー・パイプラインを開始するには、`cics://SERVICE/service?targetServiceUri=targetServiceUri` という URI を使用します (*service* はサービスの名前、*targetServiceUri* はサービスのパス)。サービスのパスを使ってトランスポート・ハンドラーは要求を解決する URIMAP リソースを見つけた後、正しいプロバイダー・パイプラインにそれを渡します。CICS は、その処理ではサービスの名前を使用しません。

サービス用の URIMAP リソースがインストールされていない場合、エラーが発生します。URIMAP リソース定義では USAGE(PIPELINE) を指定する必要もあります。トランスポート・ハンドラーは **targetServiceUri** パラメーターの値を DFHWS-URI コンテナに入れて、プロバイダー・パイプラインを開始します。

- 別のリクエスター・パイプラインを開始するには、`cics://PIPELINE/pipeline?targetServiceUri=targetServiceUri` という URI を使用します (*pipeline* は開始する PIPELINE リソースの名前、*targetServiceUri* は DFHWS-URI コンテナで次のパイプラインに渡す値)。

それぞれのタイプの URI には追加のパラメーターがあり、照会ストリングとしてこれらを追加できます。これらの URI の形式とコーディング上の規則について、詳しくは、138 ページの『DFHWS-URI コンテナ』を参照してください。

2. XML エディターを使用して、次のようにメッセージ・ハンドラーをパイプライン構成ファイルに追加します。

```
<service>
  <service_handler_list>
    <handler>
```

```
<program>MYPROG</program>
</handler>
</service_handler_list>
</service>
```

3. この新しいメッセージ・ハンドラー・プログラムをパイプラインに含めるために、リクエスト・パイプライン用の PIPELINE リソースを使用不可にし、破棄して、再インストールします。
4. メッセージ・ハンドラー・プログラムを CICS 領域にインストールします。

タスクの結果

リクエスト・パイプラインを通り抜ける次のサービス要求は、新しいメッセージ・ハンドラーによって処理されます。

次のタスク

リクエスト・パイプラインの変更内容をテストすることによって、サービス要求が正しい場所に送られ、メッセージ・ハンドラー・プログラムが設計どおりに動作することを確認します。

関連概念

258 ページの『リクエスト・パイプライン処理を制御するためのオプション』サービス・リクエスト・パイプラインにおいて、メッセージ・ハンドラーは URI を変更することによって Web サービス要求が送られる場所を決定できます。CICS はさまざまな URI 形式をサポートするため、パイプラインで Web サービス要求が処理される方法をかなり柔軟に制御することができます。

関連資料

138 ページの『DFHWS-URI コンテナ』

DFHWS-URI は、サービスの URI を格納する、DATATYPE(CHAR) のコンテナです。

第 10 章 Web サービス・トランザクションのサポート

Web Services Atomic Transaction (または WS-AtomicTransaction) 仕様および Web Services Coordination (または WS-Coordination) 仕様では、複数のトランザクション処理システムを Web サービス環境で同時に運用できる短期間のトランザクション向けプロトコルが定義されます。WS-AtomicTransaction を使用するトランザクションには、原子性、一貫性、独立性および耐久性という *ACID* 特性があります。

これらの仕様は、<http://www.ibm.com/developerworks/library/specification/ws-tx/> で参照できます。

注: CICS は、2004 年 11 月レベルの仕様をサポートします。

Web サービス・プロバイダーまたは Web サービス・リクエスターとして配置される CICS アプリケーションは、これらの仕様をサポートするその他の Web サービス実装環境との分散トランザクションに参加できます。

登録サービス

登録サービスとは、アプリケーションを調整プロトコルに登録するための WS-Coordination モデルの一部のことです。分散トランザクションでは、参加しているシステム内で複数の登録サービスが互いに対話し、接続されているアプリケーションがこうしたプロトコルに参加できるようになります。

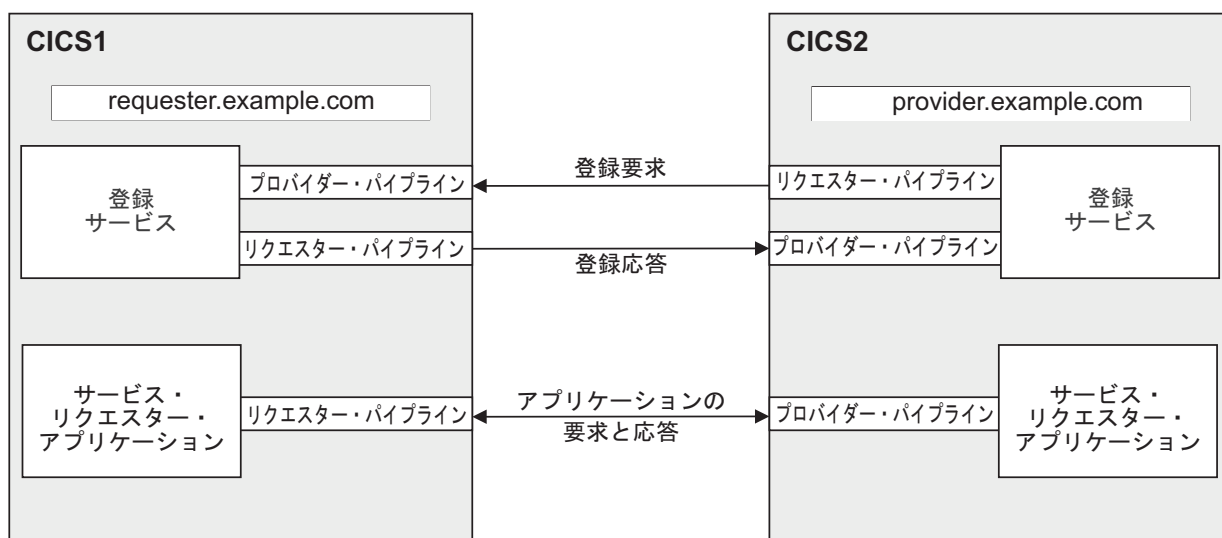


図 25. 登録サービス

図 25 には、2 つの CICS システムである CICS1 および CICS2 を示します。CICS1 のサービス・リクエスター・アプリケーションが、CICS2 のサービス・プロ

パイダー・アプリケーションを呼び出します。2 つの CICS 領域と 2 つのアプリケーションは、2 つのアプリケーションが WS-Coordination プロトコルを使用して 1 つの分散トランザクションに参加できるように構成されます。サービス・リクエスター・アプリケーションはコーディネーターで、サービス・プロバイダー・アプリケーションは参加プログラムです。

これらのプロトコルの補助として、2 つの CICS 領域の登録サービスがトランザクションの開始時とトランザクションの終了時に対話します。これらの対話中、2 つの領域の登録サービスは、サービス・プロバイダーおよびサービス・リクエスターとして、それぞれ別の時間に動作できます。したがって、各領域では、登録サービスがサービス・プロバイダー・パイプラインおよびサービス・リクエスター・パイプラインを使用します。パイプラインは、PIPELINE および関連リソースとともに CICS に定義されます。

各領域の登録サービスは、エンドポイント・アドレスと関連付けられます。このため、この例では、CICS1 の登録サービスには `requester.example.com` というエンドポイント・アドレスがあり、CICS2 の登録サービスには `provider.example.com` というエンドポイント・アドレスがあります。

CICSplex では、登録サービス・プロバイダー・パイプラインをさまざまな領域に分散できます。このことは、267 ページの図 26 に示します。

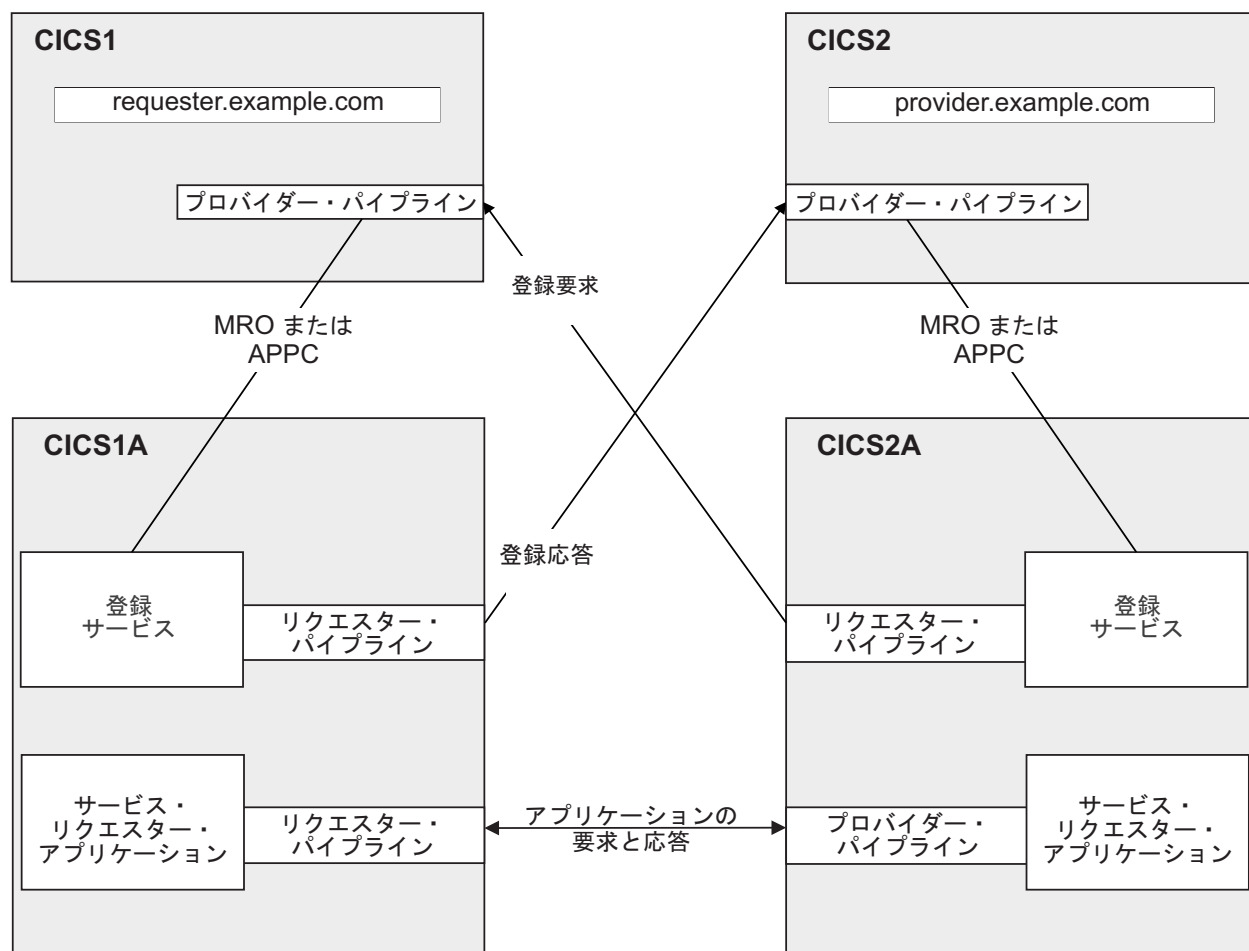


図 26. CICSplex® での登録サービス

この構成では、プロバイダー・パイプラインは MRO または APPC を使用して登録サービスと対話します。登録サービス・リクエスター・パイプラインは、アプリケーションのリクエスター・パイプラインと同じ領域に残る必要があります。

この構成は、サービス・リクエスター・アプリケーションおよびサービス・プロバイダー・アプリケーションが多数の領域にまたがって分散している場合に便利です。アプリケーションのパイプラインが Web サービス・トランザクションに参加するように構成する場合は、登録サービス・プロバイダー・パイプラインの IP アドレスとポート番号を入力して、登録サービスのエンドポイントに関する情報を提供する必要があります。エンドポイントを 1 つにすると、すべてのパイプラインに同じ情報が格納されるため、構成を単純化できます。例えば、図 26 では、アプリケーションのリクエスター・パイプラインに指定する IP アドレスは、`requester.example.com` になります。

サービス・プロバイダー・アプリケーションにも同じ引数が適用されます。この例では、プロバイダー・アプリケーションのパイプラインに指定する IP アドレスは `requester.example.com` になります。

Web サービス・トランザクションに合わせた CICS の構成

Web サービス・リクエスター・アプリケーションおよび Web サービス・プロバイダー・アプリケーションが Web サービス・トランザクションの場合、それに応じて CICS を構成する必要があります。このためには、いくつかの CICS リソースをインストールします。

始める前に

これらのリソースをインストールする前に、Web サービス・トランザクションをサポートするための CICS 提供のパイプライン構成ファイルの場所を確認しておく必要があります。デフォルトでは、構成ファイルは `/usr/lpp/cicsts/cicsts41/pipeline/configs` ディレクトリにあります。デフォルトのファイル・パスは CICS のインストール時に変更されている場合があります。

このタスクについて

Web サービス・トランザクション対応の CICS サポートでは、CICS 提供の登録サービスが使用されます。登録サービスは、サービス・プロバイダーとサービス・リクエスターで構成されます。アプリケーションがすべてサービス・プロバイダーであったり、すべてサービス・リクエスターであったりする場合でも、サービス・プロバイダーとサービス・リクエスター両方のリソースをインストールしなければなりません。

サービス・プロバイダー・アプリケーションおよびサービス・リクエスター・アプリケーションの実行時に呼び出されるヘッダー・ハンドラー・プログラムのプログラム定義もインストールする必要があります。

Web サービス・トランザクション用に CICS を構成するために必要なリソースはすべて DFHWSAT グループに提供されています。ただし例外として DFHPIDIR は DFHPIVS、DFHPIVR、または DFHPICF グループのいずれかに提供されています。DFHWSAT グループは DFHLIST リストに含まれないため、自動的にインストールされません。CICS によって提供される DFHWSAT グループのリソースを変更することはできません。

Web サービス・トランザクション用に CICS を構成するには、次のようにします。

1. DFHPIDIR データ・セットを始動 JCL に追加します。DFHPIDIR には、コンテキストとタスクの間のマッピングが格納されます。
 - a. DFHPIDIR データ・セット用の新しい DD ステートメントを CICS 始動 JCL に追加します。
 - b. DFHDEFDS.JCL の情報を使用して DFHPIDIR データ・セットを作成します。DFHPIDIR のデフォルトの RECORDSIZE は 1 KB で、ほとんどの用途ではこれが適しています。必要に応じて、より大きい RECORDSIZE を使って DFHPIDIR を作成することができます。
 - c. データ・セット用の適切なグループ (DFHPIVS、DFHPIVR、または DFHPICF) を CICS システムにインストールします。これらのグループについて詳しくは、WS-AT データ・セットの定義を参照してください。

複数の CICS 領域の間で DFHPIDIR ファイルを共有するには、MRO を介してそれらの領域が論理的に接続されている必要があります。論理サーバーとして機能している領域のグループにつき 1 つのデータ・セットをインストールする必要があります。

ヒント: 論理的に接続されていない領域間でのデータ・セットの共有は勧められていません。

2. DFHWSAT グループの内容を別のグループにコピーします。CICS によって提供される DFHWSAT グループのリソースを変更することはできません。ただし、PIPELINE リソース内の CONFIGFILE 属性の変更は必要になります。
3. 登録サービスのプロバイダー PIPELINE リソースを変更します。この PIPELINE には DFHWSATP という名前が付けられ、この PIPELINE によって、パイプライン構成ファイル `/usr/lpp/cicsts/cicsts41/pipeline/configs/registrationservicePROV.xml` が CONFIGFILE 属性に指定されます。
 - a. システム内のファイルの場所を反映するように CONFIGFILE 属性を変更します。
 - b. その他の属性は未変更のままにしておきます。

パイプライン構成ファイルは、提供されているまま使用して、内容を変更しないでください。

4. PIPELINE リソースをインストールします。登録サービス・プロバイダー PIPELINE リソースは、サービス・リクエスター・アプリケーションやサービス・プロバイダー・アプリケーションと同じ CICS 領域に置く必要はありませんが、適切な MRO 接続または APPC 接続により、同じ CICS 領域に接続しておく必要があります。
5. 登録サービス・プロバイダーが使用する URIMAP を、PIPELINE と同じ領域に変更しないでインストールします。この URIMAP には、DFHRSURI という名前が付けられます。
6. 登録サービスのリクエスター PIPELINE リソースを変更します。この PIPELINE には DFHWSATR という名前が付けられ、この PIPELINE によって、パイプライン構成ファイル `/usr/lpp/cicsts/cicsts41/pipeline/configs/registrationserviceREQ.xml` が CONFIGFILE 属性に指定されます。
 - a. システム内のファイルの場所を反映するように CONFIGFILE 属性を変更します。
 - b. その他の属性は未変更のままにしておきます。

パイプライン構成ファイルは、提供されているまま使用して、内容を変更しないでください。

7. PIPELINE リソースをインストールします。登録サービス・リクエスター PIPELINE リソースは、サービス・リクエスター・アプリケーションおよびサービス・プロバイダー・アプリケーションと同じ CICS 領域に置く必要があります。
8. 登録サービス・プロバイダー・パイプラインが使用するプログラムを、PIPELINE リソースと同じ領域にインストールします。このプログラムとは、DFHWSATX、DFHWSATR、および DFHPIRS です。2 つの PIPELINE リソースが異なる領域に存在する場合は、これらのプログラムを両方の領域にインストールする必要があります。

9. ヘッダー・ハンドラー・プログラムの PROGRAM リソース定義をインストールします。 このプログラムには、DFHWSATH という名前が付けられます。 PROGRAM は、サービス・プロバイダー・アプリケーションとサービス・リクエスター・アプリケーションが稼働する領域にインストールします。

タスクの結果

CICS は、これで、サービス・プロバイダー・アプリケーションおよびサービス・リクエスター・アプリケーションが、WS-AtomicTransaction プロトコルと WS-Coordination プロトコルを使用して分散トランザクションに参加できるように構成されました。

次のタスク

今度は、参加する各アプリケーションを個別に構成する必要があります。

Web サービス・トランザクションに合わせたサービス・プロバイダーの構成

サービス・プロバイダー・アプリケーションの目的が、Web サービス・トランザクションに参加することである場合、パイプライン構成ファイルには、`<headerprogram>` および `<service_parameter_list>` を指定する必要があります。

始める前に

サービス・プロバイダー・アプリケーションが Web サービス・トランザクションに参加できるようにするには、サービス・プロバイダー・アプリケーションが SOAP プロトコルを使用してサービス・リクエスターと通信し、かつパイプラインを構成して、CICS 提供の SOAP メッセージ・ハンドラーのいずれかを使用する必要があります。サービス・プロバイダー・アプリケーションを正しく構成した場合でも、サービス・リクエスター・アプリケーションの参加がセットアップされていると、サービス・プロバイダー・アプリケーションは、サービス・リクエスターとの Web サービス・トランザクションにのみ参加します。

このタスクについて

アプリケーションに固有のパイプライン構成情報に加えて、構成ファイルには、アプリケーションが Web サービス・トランザクションに必ず参加するように、CICS が使用する情報が格納されている必要があります。

CICS では、この情報が格納されているパイプライン構成ファイルの例を、ファイル `/usr/lpp/cicsts/cicsts41/samples/pipelines/wsatprovider.xml` で提供しています。

Web サービス・トランザクションに合わせたサービス・プロバイダーを構成します。

1. 端末ハンドラーの定義で、`<headerprogram>` エlementを `<cics_soap_1.1_handler>` エlementまたは `<cics_soap_1.2_handler>` エlement

ントの内部にコーディングします。 <program_name>、<namespace>、<localname>、<mandatory> の各エレメントを、次に示す例のとおり正確にコーディングします。

```
<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>
```

その他の <headerprogram> エレメントがアプリケーションで必要とされる場合には、それらも指定してください。

2. <registration_service_endpoint> エレメントを <service_parameter_list> の内部にコーディングします。 <registration_service_endpoint> を、次のようにコーディングします。

```
<registration_service_endpoint>
http://address:port/cicswsat/RegistrationService
</registration_service_endpoint>
```

ここで

address は、登録サービス・プロバイダー・パイプラインがインストールされている CICS 領域の IP アドレスです。

port は、登録サービス・プロバイダー・パイプラインが使用するポート番号です。

その他はすべて表示どおりに正確にコーディングします。ストリング *cicswsat/RegistrationService* は、URIMAP DFHRSURI の PATH 属性に対応します。

```
<registration_service_endpoint>
http://provider.example.com:7160/cicswsat/RegistrationService
</registration_service_endpoint>
```

Web サービス・トランザクションに合わせたサービス・リクエスターの構成

サービス・リクエスター・アプリケーションの目的が、Web サービス・トランザクションに参加することである場合、パイプライン構成ファイルには、<headerprogram> および <service_parameter_list> を指定する必要があります。

始める前に

サービス・リクエスター・アプリケーションが Web サービス・トランザクションに参加できるようにするには、サービス・リクエスター・アプリケーションが SOAP プロトコルを使用してサービス・プロバイダーと通信し、かつパイプラインを構成して、CICS 提供の SOAP メッセージ・ハンドラーのいずれかを使用する必要があります。サービス・リクエスター・アプリケーションを正しく構成した場合でも、サービス・プロバイダー・アプリケーションの参加がセットアップされていると、サービス・リクエスター・アプリケーションは、サービス・プロバイダーと

の Web サービス・トランザクションにのみ参加します。

このタスクについて

アプリケーションに固有のパイプライン構成情報に加えて、構成ファイルには、アプリケーションが Web サービス・トランザクションに必ず参加するように、CICS が使用する情報が格納されている必要があります。

CICS では、この情報が格納されているパイプライン構成ファイルの例を、ファイル /usr/lpp/cicsts/cicsts41/samples/pipelines/wsatrequester.xml で提供しています。

1. `<headerprogram>` エlementを `<cics_soap_1.1_handler>` Elementまたは `<cics_soap_1.2_handler>` Elementの内部にコーディングします。
`<program_name>`、`<namespace>`、`<localname>`、`<mandatory>` の各Elementを、次に示す例のとおり正確にコーディングします。 例を次に示します。

```
<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>
```

その他の `<headerprogram>` Elementがアプリケーションに必要な場合は、それらも指定できます。

2. `<registration_service_endpoint>` Elementを `<service_parameter_list>` の内部にコーディングします。 `<registration_service_endpoint>` を、次のようにコーディングします。

```
<registration_service_endpoint>
http://address:port/cicswsat/RegistrationService
</registration_service_endpoint>
```

ここで

address は、登録サービス・プロバイダー・パイプラインがインストールされている CICS 領域の IP アドレスです。

port は、登録サービス・プロバイダー・パイプラインが使用するポート番号です。

`<registration_service_endpoint>` Elementの始まり、その内容、および `<registration_service_endpoint>` Elementの終わりの間にスペースが存在してはなりません。この例では、分かりやすくするためにスペースを含めています。

3. 同じ作業単位内の複数の要求で同じトランザクション・コンテキストを使用する代わりに、要求ごとに CICS で新しいトランザクション・コンテキストを作成したい場合には、次のようにして `<service_parameter_list>` 内の空のElement `<new_tx_context_required/>` をパイプライン構成ファイルに追加します。

```

<service_parameter_list>
  <registration_service_endpoint>
    http://requester.example.com:7159/cicswsat/RegistrationService
  </registration_service_endpoint>
  <new_tx_context_required/>
</service_parameter_list>

```

<registration_service_endpoint> エレメントの始まり、その内容、および
<registration_service_endpoint> エレメントの終わりの間にスペースが存在してはなりません。この例では、分かりやすくするためにスペースを含めています。

<new_tx_context_required/> 設定は CICS のデフォルトではなく、サンプル・パイプライン構成ファイル wsatprovider.xml には含まれていません。

<service_parameter_list> 内の <new_tx_context_required/> をパイプライン構成ファイルに追加する場合、CICS のループバック呼び出しが可能です。この状態ではデッドロックが発生する可能性があることに注意してください。

SOAP メッセージがアトミック・トランザクションの一部であるかどうかの判別

CICS Web サービスがアトミック・トランザクション・パイプラインで呼び出されるときは、SOAP メッセージは必ずしもアトミック・トランザクションの一部である必要はありません。

このタスクについて

SOAP メッセージがアトミック・トランザクションの一部である場合、
<soapenv:Header> エレメントには特定の情報が格納されています。SOAP メッセージがアトミック・トランザクションの一部であるかどうかを調べるには、次のいずれかを行うことができます。

- トレースを使用して <soapenv:Header> エレメントの内容を調べます。
 1. コンポーネント PI を使用して補助トレースを実行し、トレース・レベルを 2 に設定します。
 2. トレース・ポイント PI 0A31 を探します。ここには、要求コンテナに関する情報が格納されています。特に、<cicswsa:Action> エレメントの直前に現れる PIIS EVENT - REQUEST_CNT を探します。
- DFHREQUEST コンテナにデータ RECEIVE-REQUEST が格納されている場合は、DFHWSATP パイプラインでユーザー作成のメッセージ・ハンドラー・プログラムを使用して、このコンテナの内容を表示します。この方法を選択する場合は、パイプライン構成ファイルで忘れずにメッセージ・ハンドラー・プログラムを定義します。

例

以下の例は、SOAP エンベロープ・ヘッダーに表示されるアトミック・トランザクションに関する情報を示しています。

```

<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 は、SOAP メッセージがアトミック・トランザクションに参加することを意図していることを示しています。ここには、調整サービスの一部になる Web サービス・プロバイダーに必要な情報が格納されており、プロバイダーはヘッダーを認識して処理するよう構成されていると仮定されています。
2. CoordinationType は、調整コンテキストが準拠する WS-AT 仕様のバージョンを示します。
3. 調整の RegistrationService は、コーディネーターの登録ポイントの場所、および参加している Web サービスがアトミック・トランザクションのコンポーネントとして登録しようとしたときにコーディネーターに戻す必要がある情報を記述します。

アトミック・トランザクションの進行の確認

CICS Web サービスがアトミック・トランザクションの一部として呼び出されると、トランザクションはいくつかの状態を移動します。これらの状態は、トランザクションが正常化、ロールバックしなければならないかを示します。

このタスクについて

この情報にアクセスしなければならない場合は、以下のいずれかを行うことができます。

- トレースを使用して <cicswsa:Action> エLEMENTの内容を調べます。
 1. コンポーネント PI を使用して補助トレースを実行し、トレース・レベルを 2 に設定します。
 2. トレース・ポイント PI 0A31 を探します。ここには、要求コンテナに関する情報が格納されています。特に、<cicswsa:Action> ELEMENTの直前に現れる PIIS EVENT - REQUEST_CNT を探します。
- DFHWSATR パイプラインと DFHWSATP パイプラインでユーザー作成のメッセージ・ハンドラー・プログラムを使用して、DFHWS-SOAPACTION コンテナの内容を表示します。この方法を選択する場合は、パイプライン構成ファイルで忘れずにメッセージ・ハンドラー・プログラムを定義します。

例

正常に完了して、コミットされるトランザクションの状態は、次のとおりです。

```
"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 "
```

ロールバックされるトランザクションの状態は、次のとおりです。

```
"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"
```

第 11 章 バイナリー・データの MTOM/XOP 最適化のサポート

標準的な SOAP メッセージでは、バイナリー・オブジェクトは Base64 エンコードされており、メッセージの本文に組み込まれています。これによりサイズが 33% 増加します。大規模なバイナリー・オブジェクトでは、このようなサイズの増加が伝送時間に重大な影響を与えることがあります。MTOM/XOP を実装すれば、この問題を解決できます。

SOAP Message Transmission Optimization Mechanism (MTOM) 仕様および XML-binary Optimized Packaging (XOP) 仕様 (MTOM/XOP と呼ばれることが多い) は、SOAP メッセージ内の大規模な base64Binary データ・オブジェクトの伝送を最適化するメソッドを定義します。

- MTOM 仕様は、バイナリー・データを分離し (そうしないと Base64 エンコードされる)、 MIME Multipart/Related メッセージを使用してそれを別のバイナリー添付ファイルに送信することによって、SOAP メッセージの最適化のメソッドを概念的に定義します。このタイプの MIME メッセージは *MTOM* メッセージと呼ばれます。データをバイナリー・フォーマットで送信するとサイズが著しく削減されるので、SOAP メッセージの伝送が最適化されます。
- XOP 仕様は、MIME メッセージを含むがこれに限定されるわけではない、パッケージ化フォーマットのバイナリー添付ファイルを使用して、XML メッセージの最適化についての実装を定義します。

トランスポート・プロトコルが WebSphere MQ、HTTP、または HTTPS の場合、CICS はリクエスター・パイプラインとプロバイダー・パイプラインの両方でこれらの仕様をサポートします。Web サービス・プロバイダーまたはリクエスターとして配置される CICS アプリケーションは、base64Binary データを SOAP メッセージに直接組み込む代わりに、このサポートを使用して、MTOM メッセージをバイナリー添付ファイルと一緒に送受信できます。

このサポートは、追加オプションをパイプライン構成ファイルに使用することで構成できます。

MTOM/XOP および SOAP

SOAP メッセージの最適化に MTOM/XOP が使用されるとき、SOAP メッセージは XOP 処理を使用して MIME Multipart/Related メッセージに直列化されます。base64Binary データが SOAP メッセージから抽出され、E メール添付ファイルのような方法で、MIME メッセージ内の別のバイナリー添付ファイルとしてパッケージされます。

添付ファイルがバイナリー・フォーマットでエンコードされるため、base64Binary データのサイズは著しく削減されます。次に SOAP メッセージの XML が XOP フォーマットに変換されます。これは、base64Binary データを、URL を使用して関連する MIME 添付ファイルを参照する特殊な <xop:Include> エレメントで置き換えることによって行います。

変更された SOAP メッセージは *XOP* 文書 と呼ばれ、メッセージ内にルート文書を形成します。XOP 文書とバイナリー添付ファイルが一緒になって *XOP* パッケージを形成します。SOAP MTOM 仕様に適用されるとき、XOP パッケージは MTOM フォーマットの MIME メッセージです。

ルート文書は、MIME メッセージ全体のコンテンツ・タイプ・ヘッダーで Content-ID を参照することで識別します。コンテンツ・タイプ・ヘッダーの例を示します。

```
Content-Type: Multipart/Related; boundary=MIME_boundary;  
type="application/soap+xml"; start="<claim@insurance.com>"
```

start パラメーターには、XOP 文書の Content-ID が含まれます。このパラメーターがコンテンツ・タイプ・ヘッダーに含まれていない場合は、MIME メッセージの最初の部分が XOP 文書であると想定されます。

MIME メッセージ内の添付ファイルの順序は重要ではありません。例えば一部のメッセージでは、バイナリー添付ファイルが XOP 文書の前に現れることがあります。MIME メッセージを処理するアプリケーションは、特定の順序で現れる添付ファイルに依存してはなりません。詳しくは、MTOM/XOP 仕様を参照してください。

以下の例は、JPEG イメージを含む単純な SOAP メッセージを XOP 処理を使用して最適化する方法を示しています。SOAP メッセージは次のとおりです。

```
<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>
```

SOAP メッセージの MTOM/XOP パージョンは以下のとおりです。

```
MIME-Version: 1.0  
Content-Type: Multipart/Related; boundary=MIME_boundary;  
type="application/soap+xml"; start="<claim@insurance.com>" 1
```

```
--MIME_boundary  
Content-Type: application/soap+xml; charset=UTF-8  
Content-Transfer-Encoding: 8bit  
Content-ID: <claim@insurance.com> 2
```

```
<soap:Envelope  
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"  
  xmlns:xop='http://www.w3.org/2004/08/xop/include'  
  xmlns:xop-mime='http://www.w3.org/2005/05/xmlmime'>  
  <soap:Body>  
    <submitClaim>  
      <accountNumber>5XJ45-3B2</accountNumber>  
      <eventType>accident</eventType>  
      <image xop-mime:content-type='image/jpeg'><xop:Include href="cid:image@insurance.com"/></image> 3  
    </submitClaim>  
  </soap:Body>  
</soap:Envelope>
```

```
--MIME_boundary  
Content-Type: image/jpeg  
Content-Transfer-Encoding: binary  
Content-ID: <image@insurance.com> 4
```

...binary JPG image...

--MIME_boundary--

1. **start** パラメーターは、MIME メッセージのどの部分がルート XOP 文書であるかを示します。
2. Content-ID 値は MIME メッセージの一部を識別します。このケースではルート XOP 文書です。
3. <xop:Include> エレメントは JPEG バイナリー添付ファイルを参照します。
4. Content-ID はバイナリー添付ファイル内でこの JPEG を識別します。

CICS における MTOM メッセージとバイナリー添付ファイル

CICS は、MTOM ハンドラー・プログラムと XOP 処理を使用して、Web サービス・プロバイダーと Web サービス・リクエスター両方のパイプラインで、MTOM メッセージの処理をサポートおよび制御します。

MTOM ハンドラーと XOP 処理は、パイプライン構成ファイルに定義されるオプションを使用して、使用可能に設定し、構成できます。MTOM ハンドラーは、使用可能になると、XOP 文書とバイナリー添付ファイルを含むインバウンド MTOM メッセージを受け入れてアンパックします。アウトバウンド MTOM メッセージはパックされて送信されます。MTOM ハンドラーがパイプラインで使用可能でない場合に CICS が MTOM メッセージを受け取ると、SOAP 障害で拒否されます。

プロバイダー・パイプラインは、次の目的で構成できます。

- MTOM メッセージを受け入れるが、MTOM 応答メッセージは送信しない。
- MTOM メッセージを受け入れ、同じタイプの応答メッセージを送信する。
- MTOM メッセージを受け入れるが、バイナリー添付ファイルが存在する場合にのみ MTOM メッセージを送信する。
- MTOM メッセージを受け入れ、常に MTOM 応答メッセージを送信する。
- XOP 文書およびバイナリー添付ファイルを直接モードまたは互換モードで処理する。

リクエスター・パイプラインは、次の目的で構成できます。

- MTOM メッセージを送信しないが、MTOM 応答メッセージを受け入れる。
- バイナリー添付ファイルがある場合にのみ MTOM メッセージを送信し、MTOM 応答メッセージを受け入れる。
- 常に MTOM メッセージを送信し、MTOM 応答メッセージを受け入れる。
- XOP 文書およびバイナリー添付ファイルを直接モードまたは互換モードで処理する。

サポートの方式

CICS がメッセージ内の XOP 文書フォーマットを直接サポートできない特定のシナリオがあります。例えば、Web Services Security サポートと Web サービスの検証では、XOP 文書内の <xop:Include> エレメントを解析できません。このため、パイプラインには 2 つの方式のサポートがあり、XOP 文書や任意の関連するバイナリー添付ファイルを処理します。

直接モード

直接モードでは、インバウンドまたはアウトバウンド MTOM メッセージに関連付けられるバイナリー添付ファイルは、パイプラインを通してコンテナ内に渡され、アプリケーションによって直接処理されます。データ変換の必要はありません。

互換モード

互換モードは、パイプライン処理で、メッセージが標準 XML フォーマットであり、メッセージ内に任意のバイナリー・データが base64Binary フィールドとして保管されている必要があるときに、使用されます。インバウンド・メッセージでは、XOP 文書とバイナリー添付ファイルが標準 XML メッセージに再構成されます。これは、Web Services Security が使用可能になるときのパイプラインの最初か、Web サービスの検証が使用可能になるときのパイプラインの最後に行われます。アウトバウンド・メッセージでは、標準 XML メッセージはパイプラインに従って作成され、渡されます。CICS が送信する直前に、MTOM ハンドラーによって XOP フォーマットに変換されます。

互換モードでは、バイナリー・データが base64 フォーマットに変換され、再度元に戻るため、直接モードよりかなり効率が悪くなります。しかし、アプリケーションを変更せずに、Web サービスとその他の MTOM/XOP Web サービス・リクエスターおよび Web サービス・プロバイダーとを相互運用できます。

インバウンド MTOM メッセージの処理

MTOM ハンドラーがパイプラインで使用可能な場合、DFHREQUEST または DFHRESPONSE コンテナ内のインバウンド・メッセージのヘッダーを検査して、トランスポートの処理実行中のメッセージのフォーマットを決定します。

MIME Multipart/Related メッセージを受け取ると、MTOM ハンドラーは次のようにこのメッセージをアンパックします。

1. 各バイナリー添付ファイルのヘッダーおよびバイナリー・データを、別のコンテナに入れる。
2. コンテナのリストを DFHWS-XOP-IN コンテナに入れる。
3. メッセージのルートを形成する XOP 文書を、DFHREQUEST または DFHRESPONSE コンテナに戻し、元のメッセージを置き換える。

バイナリー添付ファイルがない場合は、XOP 文書は通常の XML メッセージとして処理され、XOP 処理は必要ありません。バイナリー添付ファイルがある場合は、XOP 処理がメッセージについて使用可能になります。

XOP 処理が使用可能な場合は、MTOM ハンドラーがパイプラインのプロパティーを検査して、現行メッセージを直接モードで処理するか互換モードで処理するかを決定し、この情報を DFHWS-MTOM-IN コンテナに入れます。

プロバイダー・モードでは、MTOM ハンドラーは DFHWS-MTOM-OUT コンテナも作成して、アウトバウンド応答メッセージが処理される方法を決定します。

直接モード

CICS Web サービス・サポートを使用している場合 (つまり、サービス・プロバイダー・パイプラインが DFHPITP アプリケーション・ハンドラーを使用するか、サービス・リクエスター・パイプラインが INVOKE WEBSERVICE を使用して呼び出される場合)、パイプラインは直接モードで XOP 文書とバイナリー添付ファイルを処理できます。

この方式では、XOP 文書および関連するコンテナは、MTOM ハンドラーによって、処理のためにパイプライン内の次のメッセージ・ハンドラーに渡されます。

CICS Web サービス・サポートは <xop:Include> エレメントを解釈します。

base64Binary フィールドがアプリケーション・データ構造内でコンテナとして表される場合、添付ファイルのコンテナ名はこの構造に保管されます。このフィールドが可変または固定長のストリングとして表される場合、コンテナの内容は該当するアプリケーション・データ構造のフィールドにコピーされます。それからデータ構造がアプリケーション・プログラムに渡されます。

互換モード

パイプラインがカスタム・アプリケーション・ハンドラーを使用するように構成されている場合、または Web Services Security も使用可能な場合は、メッセージは互換モードで処理されます。この方式では、XOP 文書とバイナリー添付ファイルは XOP 処理を使用して即時に SOAP メッセージに再構成されるので、内容はパイプライン内で正常に処理されます。XOP 処理では以下を行います。

1. <xop:Include> エレメントについて XOP 文書をスキャンして、参照される添付ファイルから、バイナリー・データを持つ各オカレンスを base64 エンコードされたフォーマットに置き換える。
2. DFHWS-XOP-IN コンテナおよびすべての添付ファイルのコンテナを破棄する。

再構成された SOAP メッセージは、この後パイプライン内の次のハンドラーに渡され、通常どおりに処理されます。

Web サービスの検証が使用可能であれば、メッセージがアプリケーション・ハンドラーに到達するときに、パイプラインが互換モードに切り替えます。メッセージは、SOAP メッセージに再構成され、検証されてから、アプリケーションに渡されます。

アウトバウンド MTOM メッセージの処理

アウトバウンド MTOM メッセージを送信するようにパイプラインが構成されるとき、Web サービスとパイプラインのプロパティが検査されて、メッセージが処理および送信される方法が決定します。

これらのプロパティは、DFHWS-MTOM-OUT と DFHWS-XOP-OUT の 2 つのコンテナに保管されます。リクエスター・モードのパイプラインでは、これらのコンテナは、アプリケーションが EXEC CICS INVOKE WEBSERVICE コマンドを出すときに、CICS によって作成されます。プロバイダー・モードのパイプラインでは、DFHWS-MTOM-OUT コンテナは、インバウンド・メッセージが受信されたときに決定されたオプションで、すでに初期化されています。

アウトバウンド・メッセージを直接モードで処理できる場合、メッセージの最適化は即時に行われます。アウトバウンド・メッセージを互換モードで処理する必要がある場合は、最適化はパイプライン処理の最後に行われます。

Web サービス・プロバイダーまたは Web サービス・リクエスターのアプリケーションを CICS Web サービス・アシスタントを使用して配置していない場合、またはパイプラインで Web サービスの検証を使用可能にしているか、Web Services Security を使用可能にしている場合は、アウトバウンド・メッセージは互換モードで処理されます。

直接モード

直接モードでは、次の処理が行われます。

1. XOP 文書は、コンテナ DFHWS-DATA にあるアプリケーションのデータ構造から構成されます。サイズが 1,500 バイト以上のすべてのバイナリー・フィールドが識別され、バイナリー添付ファイルを記述するバイナリー・データおよび MIME ヘッダーが別のコンテナに入ります。バイナリー・データがすでにコンテナ内にある場合は、そのコンテナが添付ファイルとして直接使用されます。次に、生成された Content-ID を使用して、<xop:Include> エレメントが、通常の base64 エンコードされたバイナリー・データの代わりに XML に挿入されます。例を次に示します。

```
<xop:Include href="cid:generated-content-ID-value"
xmlns:xop="http://www.w3.org/2004/08/xop/include">
```
2. すべてのコンテナが DFHWS-XOP-OUT コンテナ内の添付ファイル・リストに追加されます。
3. SOAP ハンドラーが DFHWS-DATA を処理した場合は、XOP 文書および SOAP エンベロープが DFHREQUEST または DFHRESPONSE コンテナに保管され、パイプラインを介して処理されます。
4. 最後のメッセージ・ハンドラーが終了すると、MTOM ハンドラーが、XOP 文書およびバイナリー添付ファイルを MIME Multipart/Related メッセージにパックし、Web サービス・リクエスターまたは Web サービス・プロバイダーに送信します。その後で DFHWS-XOP-OUT コンテナおよび関連するコンテナがすべて破棄されます。

互換モード

パイプラインが XOP 文書を直接処理できない場合は、次の処理が行われます。

1. SOAP 本体がアプリケーション・データ構造から DFHWS-DATA に構成され、通常どおりパイプラインで処理されます。
2. 最終ハンドラーがメッセージの処理を終了すると、MTOM ハンドラーが DFHWS-MTOM-OUT コンテナのオプションを検査し、オプションでバイナリー添付ファイルが存在するかどうかを考慮に入れて、MTOM を使用するかどうかを決定します。MTOM ハンドラーが、MTOM は必要ないと判断する場合、XOP 処理は行われず、SOAP メッセージは CICS によって通常どおりに送信されます。
3. MTOM ハンドラーが、アウトバウンド・メッセージを MTOM フォーマットで送信すると決定すると、データをバイナリー添付ファイルに分割するのに適格なフィールドについて、XOP 処理がメッセージをスキャンします。適格なフィー

ルドとは、MIME contentType 属性がエレメントに指定されていて、関連するバイナリー値が正規形式で有効な base64Binary データからなるものです。データのサイズは 1,500 バイトより大きい必要があります。XOP 処理は、バイナリー添付ファイルと添付リストを作成してから、これらのフィールドを <xop:Include> エレメントで置き換えます。

4. MTOM ハンドラーが XOP 文書およびバイナリー添付ファイルを MIME Multipart/Related メッセージとしてパックし、CICS が Web サービス・リクエスターまたは Web サービス・プロバイダーに送信します。

MTOM/XOP 使用の際の制限

パイプラインで MTOM ハンドラーを使用可能にすると、MTOM/XOP 最適化を使用する Web サービスの実装をサポートできます。互換モードのオプションでは、Web サービス・アプリケーションを変更せずに、Web サービスと相互運用できます。ただし、ある状態では、MTOM/XOP を使用できないか、使用が制限されます。

CICS Web サービス・アシスタントの使用

MTOM/XOP の直接モード最適化を使用できるのは、少なくともマッピング・レベル 1.2 の DFHWS2LS を使用し、xsd:base64Binary タイプのフィールドが WSDL 文書に少なくとも 1 つ含まれる場合だけです。DFHLS2WS を使って有効化された Web サービスは、XOP 最適化を使用できません。

CHAR-VARYING=BINARY が指定された DFHLS2WS によって生成される Web サービスは、MTOM/XOP 最適化を使用できる可能性があります。DFHLS2WS を使って生成された他の Web サービスにはバイナリー・データが含まれず、MTOM/XOP 最適化を使用することはできませんが、MTOM/XOP をサポートする PIPELINE では正常に作動します。

プロバイダー・パイプライン

CICS は、プロバイダー・パイプラインで構成できる、DFHPITP と呼ばれるデフォルトのアプリケーション・ハンドラーを提供します。このアプリケーション・ハンドラーでは、XOP 文書を処理し、必要なコンテナを作成して、直接モードと互換モードの両方でのパイプライン処理をサポートできます。プロバイダー・パイプラインで独自のアプリケーション・ハンドラーを使用していて、MTOM/XOP を使用可能に設定したい場合は、パイプラインを構成して互換モードで実行します。

リクエスター・パイプライン

アプリケーションで INVOKE WEBSERVICE コマンドを使用する場合、CICS は SOAP メッセージの最適化を直接モードおよび互換モードで処理します。プログラム DFHPIRT を使用してパイプラインを開始する場合、互換モードで MIME Multipart/Related メッセージの送受信のみ行えます。

Web Services Security

パイプライン構成ファイルで MTOM ハンドラーを使用可能にして直接モードで実行する場合で、Web Services Security メッセージ・ハンドラーも使用可能にする場合、パイプラインは、互換モードで MTOM メッセージの処理のみサポートします。

バイナリー・データの処理

Web サービスに大容量のバイナリー・データ (例えば JPEG などのグラフ

ミック・ファイル) を組み込む場合、 MTOM/XOP を使用して、サービス・プロバイダーまたはサービス・リクエスターに送信されるメッセージのサイズを最適化できます。 MTOM/XOP を使用して最適化できるバイナリー・データの最小サイズは 1500 バイトです。フィールド内のバイナリー・データが 1500 バイトより小さいと、CICS はそのフィールドを最適化しません。

XOP 仕様に記載されるように、base64Binary データに空白文字を入れることはできません。 base64Binary データを作成するすべてのアプリケーション・プログラムで正規形式を使用する必要があります。アウトバウンド・メッセージの base64Binary データに空白文字が含まれていると、CICS はそのデータをバイナリー添付ファイルに変換しません。 base64Binary データが CICS によって生成される場合は、フィールドが正規形式で提供されるため、空白文字は含まれません。

アウトバウンド・メッセージで、互換モードで XOP 処理を行うには、contentType 属性が base64Binary フィールドに存在しなければなりません。 contentType 属性が hexBinary フィールドに存在してはなりません。

Web サービスの検証

Web サービスの検証のスイッチを入れると、次のパイプライン処理が起こります。

- インバウンド XOP 文書が直接モードでパイプラインを通過すると、CICS は自動的に互換モードに切り替わり、CICS Web サービス・サポートが文書を検証しようとするときに、元の標準 XML に変換します。
- アウトバウンド SOAP メッセージは標準 XML として生成され、互換モードで処理されます。

これは、検証処理で XOP 文書の内容を処理できないためです。

MTOM/XOP をサポートするための CICS の構成

CICS で MTOM メッセージのサポートを構成するには、パイプライン構成ファイルに MTOM ハンドラーを追加する必要があります。

始める前に

この作業を行う前に、MTOM/XOP の構成情報を追加するパイプライン構成ファイルを、識別するか、作成する必要があります。

このタスクについて

1. パイプライン構成ファイルに <cics_mtom_handler> エlementを追加する。 このElementが、<provider_pipeline> Elementの最初になり、<requester_pipeline> Element内の <service_parameter_list> の直前のElementになります。 次のElementをコーディングします。

```
<cics_mtom_handler>
  <dfhmtom_configuration version="1">
  </dfhmtom_configuration>
</cics_mtom_handler>
```

<dfhmtom_configuration> エLEMENTは、この構成内の他のELEMENTのコンテナです。MTOM/XOP 処理のデフォルト設定を受け入れる場合は、以下のように空のELEMENTを指定できます。

```
<cics_mtom_handler/>
```

2. オプション: <mtom_options> ELEMENTをコーディングする。このELEMENTは、サービス・プロバイダーおよびサービス・リクエスターの両方のパイプラインで、アウトバウンド・メッセージを MTOM メッセージとしてパックするかどうかを指定します。
 - a. send_mtom 属性をコーディングして、アウトバウンド・メッセージが MTOM メッセージとして送信されるかどうかを定義する。この属性については、94 ページの『<mtom_options>ELEMENT』を参照してください。
 - b. send_when_no_xop 属性をコーディングして、バイナリー添付ファイルが存在しないときに、アウトバウンド・メッセージが MTOM メッセージとして送信されるかどうかを定義する。この属性については、94 ページの『<mtom_options>ELEMENT』を参照してください。
3. オプション: <xop_options> ELEMENTを apphandler_supports_xop 属性と一緒にコーディングする。これで、アプリケーション・ハンドラーが XOP 文書を直接処理できるかどうかを指定します。この属性を組み込まない場合、デフォルトは、<apphandler> ELEMENTが DFHPITP を指定するか別のプログラムを指定するかに応じて異なります。この属性については詳しくは、95 ページの『<xop_options>ELEMENT』を参照してください。
4. オプション: <mime_options> ELEMENTを content_id_domain 属性と一緒にコーディングする。これで、バイナリー添付ファイルの識別に使用される MIME content-ID 値を生成する際に使用する、ドメイン・ネームを指定します。この属性については詳しくは、97 ページの『<mime_options>ELEMENT』を参照してください。

例

次の例は、オプションのELEMENTがすべて存在する、完全な <cics_mtom_handler> を示します。

```
<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>
```

第 12 章 Web Services Addressing のサポート

CICS は、Worldwide Web Consortium (W3C) の Web Services Addressing (WS-Addressing) 仕様を使用するサービスをサポートします。この仕様ファミリーは、Web サービスのアドレス指定とエンドツーエンドのアドレッシングを可能にする、トランスポートに依存しないメカニズムを提供します。

CICS は、WS-Addressing を使用する Web サービスからの要求を既存の Web サービス・アプリケーションで受け入れることができることを保証します。SOAP メッセージのエンドポイント参照とメッセージ・アドレッシング・プロパティを使用する Web サービスを新規作成することもできます。

WS-Addressing は、メッセージ・アドレッシング・プロパティ (MAP) の形式のアドレッシング情報を SOAP メッセージ・ヘッダーに追加します。MAP には、固有のメッセージ ID や、エンドポイント参照 (メッセージの送信元、メッセージの送信先、および応答メッセージや障害メッセージの送信先が詳しく説明されている) などのメッセージング情報が含まれます。エンドポイント参照 (EPR) は特定のタイプの MAP です。これには、メッセージの宛先アドレス、アプリケーションが使用するオプションの参照パラメーター、およびオプションのメタデータが含まれます。

WS-Addressing サポートの機能

CICS には、WS-Addressing をサポートする以下のフィーチャーが含まれます。

- Web サービス・リクエスター・アプリケーションと Web サービス・プロバイダー・アプリケーションは、再デプロイを行わずに、WS-Addressing を使用している他のサービスと対話することができます。パイプラインの新しいメッセージ・ハンドラーであるアドレッシング・メッセージ・ハンドラー DFHWSADH は、WS-Addressing 情報を含むメッセージを指定の Web サービスに経路指定します。
- WS-Addressing API コマンドを使用してエンドポイント参照を作成し、アドレッシング・コンテキストを作成、更新、削除、および照会するアプリケーションを作成することができます。
- 応答メッセージをリクエスター・エンドポイント以外のエンドポイントに経路指定することができます。例えば、障害メッセージを専用の障害ハンドラーに経路指定することができます。
- 参照パラメーターを SOAP ヘッダー内の MAP の一部としてアプリケーションに渡すことができます。

WS-Addressing 仕様のサポートと相互運用性

デフォルトで、CICS は以下の勧告仕様をサポートしています。

- W3C WS-Addressing 1.0 - Core
- W3C WS-Addressing 1.0 - SOAP Binding
- W3C WS-Addressing 1.0 - Metadata

これらの仕様は、<http://www.w3.org/2005/08/addressing> の名前空間によって識別されます。特に明記しない限り、本書に記載される WS-Addressing セマンティクスは勧告仕様のことです。

相互運用性に関しては、CICS は実行依頼仕様もサポートしています。

- W3C WS-Addressing- Submission

この仕様は、<http://schemas.xmlsoap.org/ws/2004/08/addressing> の名前空間によって識別されます。実行依頼仕様は、それをインプリメントするクライアントまたは Web サービス・プロバイダーと相互運用する必要がある場合のみ使用します。

Web Services Addressing の概要

Web Services Addressing (WS-Addressing) は、SOAP メッセージのエンドポイントを指定するための標準的なフレームワークです。このフレームワークはトランスポートに中立的であるため、さまざまなトランスポート機構を使用する Web サービスの相互運用性が改善されます。WS-Addressing 仕様は、メッセージ・アドレッシング・プロパティーおよびエンドポイント参照を導入しています。

Worldwide Web Consortium (W3C) の仕様の 1 つである Web Services Addressing (WS-Addressing) は、Web サービスをアドレス指定する標準的な方法を定義し、SOAP メッセージ内のアドレッシング情報を提供することにより、Web サービス間の相互運用性を改善します。HTTP や WMQ などのさまざまなトランスポート機構を介して SOAP メッセージを送ることができます。それぞれの機構は、他とは異なる方法でメッセージの宛先情報を保管します。

WS-Addressing を使用するように構成されたパイプラインにデプロイされている既存の CICS Web サービスでは、デフォルトの WS-Addressing 設定を変更せずに、そのまま使用できます。WS-Addressing 機能を十分に利用するためには、WS-Addressing API コマンドを使用します。

メッセージ・アドレッシング・プロパティー

メッセージ・アドレッシング・プロパティー (MAP) は、SOAP ヘッダー内のエレメントとして表すことができる、明確に定義された WS-Addressing プロパティーから成るセットです。MAP は、メッセージ応答の送信先となるエンドポイントや、そのメッセージと他のメッセージの関係を示す情報などを伝達する標準的な方法として機能します。以下の表は、WS-Addressing 仕様で定義されている MAP の要約です。

表 10. WS-Addressing 仕様で定義されているメッセージ・アドレッシング・プロパティー

抽象 WS-Addressing MAP 名	SOAP WS-Addressing MAP 名	MAP コンテンツ・タイプ	多重度	説明
[action]	<wsa:Action>	xs:anyURI	1..1	メッセージのセマンティクスを一意的に識別する絶対 URI。この値は必須です。

表 10. WS-Addressing 仕様で定義されているメッセージ・アドレッシング・プロパティ (続き)

抽象 WS-Addressing MAP 名	SOAP WS-Addressing MAP 名	MAP コンテンツ・タイプ	多重度	説明
[destination]	<wsa:To>	SOAP メッセージ内の xs:anyURI アドレッシング・コンテキスト での EndpointReference	0..1	期待されるメッセージ受信側のアドレスを 指定する絶対 URI。この値が指定されない 場合、デフォルトの、仕様で定義された匿名 URI http://www.w3.org/2005/08/ addressing/anonymous になります。 アドレッシング・コンテキストでは、 <wsa:To> MAP は EPR として表されま す。 <wsa:To> が SOAP メッセージの一 部として送信されると、それはスキーマで 定義されているようにアドレスおよびその 参照パラメーターに分割されます。
[reference parameters] *	[reference parameters]*	xs:any	0..制限なし	メッセージがアドレス指定されるエンドポ イント参照の <wsa:ReferenceParameters> プ ロパティに対応するパラメーター。この 値はオプションです。
[source endpoint]	<wsa:From>	EndpointReference	0..1	メッセージの発信元となったエンドポイン トの参照。この値はオプションです。
[reply endpoint]	<wsa:ReplyTo>	EndpointReference	0..1	このメッセージの応答を受け取ることが期 待されるエンドポイントの参照。この値は オプションです。 この値が指定されない場合、デフォルトと して http://www.w3.org/2005/08/addressing/ anonymous が使用されます。
[fault endpoint]	<wsa:FaultTo>	EndpointReference	0..1	このメッセージに関連した障害を受け取る ことが期待されるエンドポイントの参照。 この値はオプションで、デフォルトは <wsa:ReplyTo> MAP の値となります。
[relationship] *	<wsa:RelatesTo>	xs:anyURI に加えて、タイプ xs:anyURI のオプション属性	0..制限なし	このメッセージと別のメッセージの関係を 示す値のペア。このエレメントの内容は、 関連するメッセージの <wsa:MessageID> を 伝送します。オプション属性は関係のタイ プを伝送します。この値はオプションで す。 この値が指定されない場合、デフォルトと して http://www.w3.org/2005/08/addressing/ reply が使用されます。
[message id]	<wsa:MessageID>	xs:anyURI		メッセージを一意的に識別する絶対 URI。 この値はオプションで、提供されない場合 は CICS がアウトバウンド要求および応答 の値を生成します。

以下のサンプル SOAP メッセージには WS-Addressing MAP が含まれています。

```
<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>

```

エンドポイント参照

エンドポイント参照は、特定のエンドポイントについての情報をカプセル化する標準的な手段を提供する、特定のタイプの MAP です。エンドポイント参照を他のパーティーに送信して、それらが表す Web サービス・エンドポイントをターゲットにするために使用できます。以下の表は、エンドポイント参照の情報モデルの要約です。

表 11. エンドポイント参照の情報モデル

抽象プロパティ名	プロパティ・タイプ	多重度	説明
[address]	xs:anyURI	1..1	エンドポイントのアドレスを指定する絶対 URI。
[reference parameters] *	xs:any	0..制限なし	エンドポイントとインターフェースを取るために必要な、名前空間で修飾されたエレメント情報項目。
[metadata]	xs:any	0..制限なし	エンドポイントの動作、方針、および機能に関する説明。

次の XML 断片は、エンドポイント参照を例示しています。

<wsa:EndpointReference> エレメントは、URI `http://example.ibm.com/enquiry` でエンドポイントを参照し、エンドポイント参照先のインターフェースを指定するメタデータおよびアプリケーション固有のいくつかの参照パラメーターを含んでいます。

```

<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>ABCDEFG</example:DiscountId>
  </wsa:ReferenceParameters>
</wsa:EndpointReference>

```

タイプ `wsa:EndpointReferenceType` の WS-Addressing MAP は、<wsa:From>、<wsa:ReplyTo>、および <wsa:FaultTo> です。ただし <wsa:To> MAP は、WS-Addressing 1.0 標準ではタイプ `xs:anyURI` として定義されています。分か

りやすくするために、CICS はアドレッシング・コンテキストでは <wsa:To> MAP を EPR として扱います。 <wsa:To> MAP が SOAP メッセージの一部として送信されると、CICS は標準の必要に応じて、そのアドレスおよび参照パラメーターに分割します。

デフォルトの名前空間

以下の接頭部およびそれに対応する名前空間は、WS-Addressing の文書全体で参照されます。

表 12. 接頭部および対応する名前空間

デフォルトの接頭部	名前空間
xs	http://www.w3.org/2001/XMLSchema
wsa	http://www.w3.org/2005/08/addressing (勧告スキーマ) http://schemas.xmlsoap.org/ws/2004/08/addressing (サブミッション・スキーマ)
wsam	http://www.w3.org/2007/05/addressing/metadata

関連資料

138 ページの『DFHWS-URI コンテナー』

DFHWS-URI は、サービスの URI を格納する、DATATYPE(CHAR) のコンテナーです。

Web Services Addressing に合わせたパイプラインの構成

Web Services Addressing (WS-Addressing) をサポートするようにパイプラインを構成するには、パイプライン構成ファイルにアドレッシング・ハンドラーを追加する必要があります。

始める前に

この作業を行う前に、WS-Addressing の構成情報を追加するパイプライン構成ファイルを、識別するか、作成する必要があります。

どの WS-Addressing 仕様を使用するかも決定する必要があります。 W3C *WS-Addressing 1.0 Core* の仕様を使用することをお勧めします。

このタスクについて

アドレッシング・ハンドラーをパイプライン構成ファイルに追加するには、以下の手順に従います。

1. CICS アドレッシング・ハンドラー・エレメント (<cics_soap_1.1_handler> エレメントまたは <cics_soap_1.2_handler> エレメント) をリクエスター・パイプラインに追加します。 リクエスター・パイプラインは、以下のパイプライン定義のようになります。

```
<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>

```

ここに示されているとおりに、<program_name>、<localname>、および <mandatory> エlementを正確にコーディングします。 W3C *WS-Addressing 1.0 Core* 仕様を使用するには、<namespace> を <http://www.w3.org/2005/08/addressing> に設定します。 W3C *WS-Addressing Submission* 仕様を使用するには、<namespace> を <http://schemas.xmlsoap.org/ws/2004/08/addressing> に設定します。

注: <service_handler_list> Elementに複数の CICS SOAP ハンドラー・Elementを定義する場合、定義する最初の SOAP ハンドラー・Elementに DFHWSADH ヘッダー・ハンドラーを定義する必要があります。

2. CICS アドレッシング・ハンドラー・Element (<cics_soap_1.1_handler> Elementまたは <cics_soap_1.2_handler> Element) をプロバイダー・パイプラインに追加します。 アドレッシング・ハンドラーを既存のハンドラー・Elementに追加することも、新しいハンドラー・Elementを作成してアドレッシング・ハンドラーを含めることもできます。以下のパイプライン定義は、<terminal_handler> Elementの下に <cics_soap_1.1_handler> Elementをリストしたプロバイダー・パイプラインを示しています。

```

<provider_pipeline>
  <service>
    <service_handler_list/>
    <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>>false</mandatory>
        </headerprogram>
      </cics_soap_1.1_handler>
    </terminal_handler>
  </service>
</provider_pipeline>

```

注: <terminal_handler> Elementに定義されている CICS SOAP ハンドラーに DFHWSADH ハンドラーを定義する必要があります。 <terminal_handler> Elementに CICS SOAP ハンドラーが含まれていない場合、<service_handler_list> Elementの最後の CICS SOAP ハンドラーに DFHWSADH を定義します。

プロバイダー・パイプラインでは、W3C *WS-Addressing Core* 仕様レベルまたは W3C *WS-Addressing Submission* 仕様レベル (あるいはその両方) を使用するようにアドレッシング・ハンドラーを設定できます。

W3C *WS-Addressing 1.0 Core* 仕様を使用する要求をサポートするには、アドレッシング・ハンドラーの <namespace> Elementは <namespace><http://www.w3.org/2005/08/addressing></namespace> のようになります。

W3C *WS-Addressing Submission* 仕様を使用する要求をサポートするには、アドレッシング・ハンドラーの `<namespace>` エレメントは `<namespace>http://schemas.xmlsoap.org/ws/2004/08/addressing</namespace>` のようになります。

どちらの仕様でも使用できる要求をサポートするには、次に示すように DFHWSADH ヘッダー・ハンドラーを 2 回 (各名前空間ごとに 1 回) 定義します。

```
<cics_soap_1.1_handler>
  <headerprogram>
    <program_name>DFHWSADH</program_name>
    <namespace>http://www.w3.org/2005/08/addressing</namespace>
    <localname>*</localname>
    <mandatory>false</mandatory>
  </headerprogram>
  <headerprogram>
    <program_name>DFHWSADH</program_name>
    <namespace>http://schemas.xmlsoap.org/ws/2004/08/addressing</namespace>
    <localname>*</localname>
    <mandatory>false</mandatory>
  </headerprogram>
</cics_soap_1.1_handler>
```

タスクの結果

使用するパイプラインは Web Services Addressing をサポートするように構成されました。

Web Services Addressing 用に WSDL を構成する

WSDL を Web Services Addressing 用に構成する場合、考慮すべき点があります。Web サービス・アシスタントに設定するパラメーター、WSDL にデフォルトの EPR を設定するかどうか、そして WSDL 文書に `<wsa:Action>` プロパティを明示的に定義するか CICS がデフォルトの入力アクション、出力アクション、および障害アクションを構築するように設定するのか、という点です。

DFHWS2LS が WS-Addressing をサポートするために必要とするパラメーター

WSDL を Web Services Addressing 用に構成する場合、Web サービス・アシスタントである DFHWS2LS の **MINIMUM-RUNTIME** パラメーターおよび **MAPPING-LEVEL** パラメーターの値を 3.0 以上に設定する必要があります。**WSADDR-EPR-ANY** パラメーターを TRUE に設定することを検討することもできます。

Web サービス・アシスタント DFHWS2LS の **MINIMUM-RUNTIME** パラメーターを 3.0 以上に設定します。3.0 以上のランタイム・レベルでは、アシスタントが生成する WSBind ファイルはすべて Web Services Addressing を完全にサポートし、他の Web サービス・プラットフォームとも確実に相互運用できます。

Web サービス・アシスタント DFHWS2LS の **MAPPING-LEVEL** パラメーターを 3.0 以上に設定します。

WSDL 文書で定義されている要求メッセージまたは応答メッセージにタイプ `wsa:EndpointReferenceType` のエレメントがある場合、またそれらのエレメントを

WSACONTEXT BUILD API コマンドへの入力として実行時に使用する場合、**WSADDR-EPR-ANY** パラメーターを TRUE に設定します。**WSADDR-EPR-ANY** パラメーターを TRUE に設定する場合、CICS は EPR を言語構造に実行時に変換することはしません。代わりに CICS は、EPR データを <xsd:any> エレメントとして扱い、指定されたコンテナに格納する必要があります。

以下の例の WSDL フラグメントは、タイプ `wsa:EndpointReferenceType` のエレメントとして渡される <wsa:To> MAP を示しています。

```
<?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>
```

WSADDR-EPR-ANY パラメーターを TRUE に設定した DFHWS2LS によってエレメント `<xs:element name="myEpr" type="wsa:EndpointReferenceType"/>` **1** が処理されると、myEpr エレメント・データは指定されたコンテナに実行時には <xsd:any> エレメントとして格納され、生成される言語構造にそのコンテナへのポインターとして追加されます。

例えば、myEpr エレメントに関して DFHWS2LS によって生成される COBOL 言語構造を以下に示します。

```
09 myEpr.
   12 myEpr-xml-cont          PIC X(16).
   12 myEpr-xmlns-cont        PIC X(16).
```

myEpr-xml-cont コンテナは、myEpr データを格納するコンテナの名前を格納します。myEpr-xmlns-cont は、スコープ内にある XML 名前空間宣言を取り込むオプションのコンテナです。

関連概念

304 ページの『Web Services Addressing の例』

この例は、メッセージの送信に Web Services Addressing を使用する企業に対して顧客がオーダーを出すときに生じるプロセスに関する概要を示します。

関連資料

221 ページの『<xsd:any> および xsd:anyType のサポート』
DFHWS2LS および DFHSC2LS は XML スキーマにおける <xsd:any> および
xsd:anyType の使用をサポートしています。 <xsd:any> XML スキーマ・エレメ
ントを使用して未定義の内容を持つ XML 文書のセクションを記述できます。
xsd:anyType とは、すべての単純および複合データ・タイプの派生元となる基本
のデータ・タイプです。データ内容に制限や制約はありません。

デフォルト EPR の設定

WS-Addressing の使用時に、デフォルトの EPR を WSDL 文書に設定できます。デ
フォルトの EPR を使用する利点は、EPR に定義された任意の静的参照パラメータ
ー <wsa:ReferenceParameters> を要求メッセージの一部として送信できることです。

以下の WSDL 1.1 フラグメントには、デフォルト EPR <soap:address
location="http://example.ibm.com:12345/exampleTest" /> が含まれています。
<port> エレメントには子エレメント <wsa:EndpointReference> が含まれており、
子エレメントによって指定されるアドレス **2** は親エレメントによって指定される
アドレス **1** と一致する必要があります。

```
<service name="exampleService">
  <port name="examplePort" binding="s0:createBinding">
    <soap:address location="http://example.ibm.com:12345/exampleTest" /> 1

    <wsa:EndpointReference
      xmlns:example="http://example.ibm.com/namespace"
      xmlns:wsdl="http://www.w3.org/2006/01/wsdl-instance"
      wsdl:wsdlLocation="http://example.ibm.com/location "
      title="http://example.ibm.com/example/example_wsdl"
      class="http://example.ibm.com/example/example_wsdl">
        <wsa:Address>http://example.ibm.com:12345/exampleTest</wsa:Address> 2
        <wsa:Metadata>
          <wsam:InterfaceName>example:Inventory</wsam:InterfaceName>
        </wsa:Metadata>
        <wsa:ReferenceParameters>
          <example:AccountCode>123456789</example:AccountCode>
          <example:DiscountId>ABCDEFG</example:DiscountId>
        </wsa:ReferenceParameters>
      </wsa:EndpointReference>
    </port>
  </service>
```

WSDL 文書での <wsa:Action> プロパティの定義

WSDL 文書に <wsa:Action> プロパティを明示的に定義するか、CICS がデフォル
トの入力アクション、出力アクション、および障害アクションを構成するように設
定することができます。

明示的アクション

<wsa:Action> プロパティの値を WSDL 文書内に明示的に定義できます。

WSDL 1.1

以下の WSDL 1.1 フラグメントは、明示的に定義された <wsa:Action> プロパティ
ーが含まれる予約システムを表しています。

```
<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>

```

この例で、makeBooking 操作の入力アクションは明示的に `http://example.ibm.com/namespace/makeBooking` に定義され、出力アクションは明示的に `http://example.ibm.com/namespace/bookingResponse` に定義されています。

WSDL 2.0

以下の WSDL 2.0 フラグメントは、明示的に定義された `<wsa:Action>` プロパティが含まれる予約システムを表しています。

```

<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>

```

この例で、makeBooking 操作の入力アクションは明示的に `http://example.ibm.com/namespace/makeBooking` に定義され、出力アクションは `http://example.ibm.com/namespace/makeBookingResponse` に定義されています。

`<wsa:Action>` プロパティを WSDL 文書に明示的に定義していない場合、WSDL が DFHWS2LS によって処理されるときに、CICS はデフォルトのアクションを作成します。

詳しくは、以下の *W3C WS-Addressing 1.0 Metadata* 仕様を参照してください:
<http://www.w3.org/TR/ws-addr-metadata>.

WSDL 1.1 のデフォルト・アクション

`<wsa:Action>` プロパティを WSDL 1.1 文書に明示的に指定していない場合、WSDL が DFHWS2LS によって処理されるときに、CICS はデフォルトの入力アクション、出力アクション、および障害アクションを作成します。

WSDL 1.1 のデフォルト入出力アクション

勧告スキーマまたはサブミッション・スキーマに従う WSDL 1.1 文書では、デフォルトの入力または出力アクションを構成するために以下のようなパターンが CICS によって使われます。

```
[target namespace]/[port type name]/[input|output name]
```


WSDL 1.1 のデフォルト障害アクション

勧告スキーマに従う場合、CICS がデフォルトの障害アクションを構築する方法は、スキーマに記述された動作とは異なります。勧告スキーマに従う WSDL 1.1 文書では、デフォルトの障害メッセージを構成するために以下のようなパターンが CICS によって使われます。障害の名前が省略されていることに注意してください。

```
[target namespace]/[port type name]/[operation name]/Fault/
```

サブミッション・スキーマに従う場合、CICS がデフォルトの障害アクションを構築する方法は、スキーマに記述された動作に準拠します。サブミッション・スキーマに従う WSDL 1.1 文書では、デフォルトの障害メッセージを構成するために以下のようなパターンが CICS によって使われます。

```
[target namespace]/[port type name]/[operation name]/Fault/[fault name]
```

WSDL 1.1 文書用に CICS によって生成されたデフォルト・アクションの例

この予約システムの例は、CICS が WSDL 1.1 文書からデフォルト・アクションを構成する方法を示しています。

```
<description targetNamespace="http://example.ibm.com/namespace" ...>
  ...
  <portType name="bookingInterface">
    <operation name="makeBooking">
      <input element="tns:makeBooking" name="MakeBooking"/>
      <output element="tns:bookingResponse" name="BookingResponse"/>
      <fault message="tns:InvalidBooking" name="InvalidBooking"/>
    </operation>
  </interface>
  ...
</definitions>
```

この WSDL フラグメントのアドレッシング・プロパティは、以下のとおりです。

プロパティ名	値
[targetNamespace]	http://example.ibm.com/namespace
[portType name]	bookingInterface
[operation name]	makeBooking
[input name]	MakeBooking
[output name]	BookingResponse
[fault name]	InvalidBooking

以下のアクションは、これらの値から作成されます。

アクション	値
入力アクション	http://example.ibm.com/namespace/bookingInterface/MakeBooking
[input name] が指定されない場合、代わりに "Request" が付加された [operation name] の値が使用されます。例えば、この場合の入力アクションは http://example.ibm.com/namespace/bookingInterface/makeBookingRequest です。	
出力アクション	http://example.ibm.com/namespace/bookingInterface/BookingResponse

アクション	値
[output name] が指定されない場合、代わりに "Response" が付加された [operation name] の値が使用されます。例えば、この場合の出力アクションは <code>http://example.ibm.com/namespace/bookingInterface/makeBookingResponse</code> です。	
障害アクション (勧告スキーマ)	<code>http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/</code> [fault name] が省略されていることに注意してください。
障害アクション (サブミッション・スキーマ)	<code>http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/InvalidBooking</code>

詳しくは、以下の *W3C WS-Addressing 1.0 Metadata* 仕様を参照してください:
<http://www.w3.org/TR/ws-addr-metadata>。

WSDL 2.0 のデフォルト・アクション

<wsa:Action> プロパティを WSDL 2.0 文書に明示的に指定していない場合、CICS はデフォルトの入力アクション、出力アクション、および障害アクションを作成します。

WSDL 2.0 のデフォルト入出力アクション

勧告スキーマに従う WSDL 2.0 文書では、入出力のデフォルト・アクションを構成するために以下のようなパターンが CICS によって使われます。

```
[target namespace]/[interface name]/[operation name][direction token]
```

WSDL 2.0 のデフォルト障害アクション

勧告スキーマに従う場合、CICS が WS-Addressing 障害のデフォルト・アクションを構築する方法は、スキーマに記述された動作とは異なります。サブミッション・スキーマに従う場合、CICS が WS-Addressing 障害のデフォルト・アクションを構築する方法は、スキーマに記述された動作に準拠します。

勧告スキーマに従う WSDL 2.0 文書では、障害のデフォルト・アクションを構成するために以下のようなパターンが CICS によって使われます。障害の名前が省略されていることに注意してください。

```
[target namespace]/[interface name]/
```

サブミッション・スキーマに従う WSDL 2.0 文書では、障害のデフォルト・アクションを構成するために以下のようなパターンが CICS によって使われます。

```
[target namespace]/[interface name]/[fault name]
```

WSDL 2.0 文書用に CICS によって生成されたデフォルト・アクションの例

この例は、CICS が勧告スキーマに従って WSDL 2.0 文書用のデフォルト・アクションを構成する方法を示しています。

```
<description targetNamespace="http://example.ibm.com/namespace" ...>
...
<interface name="bookingInterface">
  <operation name="makeBooking" pattern="http://www.w3.org/ns/wsd1/in-out">
    <input element="tns:makeBooking" messageLabel="In"/>
  </operation>
</interface>
</description>
```

```

        <output element="tns:bookingResponse" messageLabel="Out"/>
    </operation>
</interface>
...
</definitions>

```

この WSDL フラグメントのアドレッシング・プロパティは、以下のとおりです。

プロパティ名	値
[targetNamespace]	http://example.ibm.com/namespace
[interface name]	bookingInterface
[operation name]	makeBooking
[direction token]	Request または Response のどちらか。

以下の入出力アクションは、これらの値から作成されます。

アクション	値
入力アクション	http://example.ibm.com/namespace/bookingInterface/makeBookingRequest
出力アクション	http://example.ibm.com/namespace/bookingInterface/makeBookingResponse

詳しくは、以下の *W3C WS-Addressing 1.0 Metadata* 仕様を参照してください:
<http://www.w3.org/TR/ws-addr-metadata>。

メッセージ交換

Web Services Addressing (WS-Addressing) は、片方向、両方向要求/応答、同期要求/応答、および非同期要求/応答のメッセージ交換をサポートしています。

Web Services Addressing のメッセージ交換には、メッセージ・アドレッシング・プロパティ (MAP) とエンドポイント参照 (EPR) が関係します。

実行時に CICS は、要求メッセージの SOAP ヘッダーに、関連する WS-Addressing メッセージ情報が含まれるようにします。リクエスター・アプリケーションが WS-Addressing ヘッダーを設定する必要はなく、WS-Addressing を使用していることを意識する必要さえないかもしれません。

片方向

この単純な片方向メッセージは、入力のみ操作として定義されています。この操作を表す Web サービス記述言語 (WSDL) は、次のような形式になります。

```

<operation name="myOperation">
  <input message="tns:myInputMessage"/>
</operation>

```

WS-Addressing を使用している場合、CICS は、実行時に <wsa:Action> MAP および <wsa:MessageID> MAP を WS-Addressing 要求メッセージの SOAP メッセージ・ヘッダーに追加して、確実に WS-Addressing 仕様に準拠するようにします。

<wsa:MessageID> MAP は固有の ID で、指定されない場合は CICS がこの ID を自動的に生成します。

<wsa:Action> MAP は WSDL から派生して WSBind ファイルに保管されます。

これら MAP の値は、CICS WS-Addressing API コマンドを使って指定変更することができます。

両方向要求/応答

この両方向の交換には、要求メッセージと応答メッセージが関係します。この操作の応答部分は、出力メッセージ、障害メッセージ、またはその両方として定義可能です。要求/応答操作を表す WSDL 定義は、次の形式になります。

```
<operation name="myOperation">
  <input message="tns:myInputMessage"/>
  <output message="tns:myOutputMessage"/>
  <fault="tns:myFaultMessage"/>
</operation>
```

エンドポイントに送られる要求に対する応答（または要求から生成される障害）の宛先は、応答タイプが正常か障害かに応じて、<wsa:ReplyTo> MAP または <wsa:FaultTo> MAP です。

応答の送信先を示すには、<wsa:ReplyTo> または <wsa:FaultTo> MAP を要求メッセージで指定します。

勧告仕様を使用していて、<wsa:ReplyTo> MAP に値を指定しない場合には、<wsa:ReplyTo> MAP はデフォルトの、匿名 URI (<http://www.w3.org/2005/08/addressing/anonymous>) を含むエンドポイント参照になります。これにより、CICS は応答を要求側に送り戻します。

勧告仕様を使用していて、<wsa:FaultTo> MAP に値を指定しない場合には、<wsa:FaultTo> MAP はデフォルトの、<wsa:ReplyTo> MAP の値になります。

要求側が妥当性検査の障害を発生させる不正な MAP を作成する場合、CICS は障害メッセージを <wsa:FaultTo> MAP で指定されたアドレスではなく要求側に送り戻します。

同期要求/応答

デフォルトでは、使用されている基礎プロトコルに従って両方向メッセージの応答部分が戻されます。HTTP 要求の場合、応答は HTTP 応答で同期的に戻されます。

非同期要求/応答

非同期応答の宛先は別の Web サービスであり、元のリクエスター・アプリケーションに戻されることはありません。HTTP 要求の場合、要求側のクライアントとの接続は HTTP 202 応答とともに閉じられます。Web サービス・プロバイダーが CICS システム上で実行されている場合、リクエスター・アプリケーションは空の応答メッセージを受信します。Web サービス・プロバイダーが WebSphere MQ システム上で実行されている場合、リクエスター・アプリケーションは応答を受信しません。

両方向メッセージの応答部分の宛先を変更するには、適切なアドレスを <wsa:ReplyTo> MAP、または <wsa:ReplyTo> MAP および <wsa:FaultTo> MAP に指定する必要があります。

WSDL 1.1 および WSDL 2.0 で必須の MAP の完全なリストについては、『WS-Addressing の必須のメッセージ・アドレッシング・プロパティ』を参照してください。

関連概念

32 ページの『WSDL およびメッセージ交換パターン』
WSDL 2.0 文書には、SOAP 1.2 メッセージを Web サービス・リクエスターと Web サービス・プロバイダーの間で交換する方法を定義する、メッセージ交換パターン (MEP) が含まれます。

関連資料

『WS-Addressing の必須のメッセージ・アドレッシング・プロパティ』
WS-Addressing 1.0 メタデータ仕様には、WSDL 1.1 資料および WSDL 2.0 資料に含める必要があるメッセージ・アドレッシング・プロパティ (MAP) が示されています。WS-Addressing の CICS 実装は、それら必須の MAP に値を自動的に指定することで、WS-Addressing 仕様に準拠する上で役立ちます。

WS-Addressing の必須のメッセージ・アドレッシング・プロパティ

WS-Addressing 1.0 メタデータ仕様には、WSDL 1.1 資料および WSDL 2.0 資料に含める必要があるメッセージ・アドレッシング・プロパティ (MAP) が示されています。WS-Addressing の CICS 実装は、それら必須の MAP に値を自動的に指定することで、WS-Addressing 仕様に準拠する上で役立ちます。

提供する WSDL の MAP に独自の値を指定したり、アドレッシング・コンテキストのそれらの値を CICS WS-Addressing API コマンドを使って更新したりすることができます。必須の MAP に値を指定しない場合、CICS によって値が自動的に生成されます。

以下の表は、サポートされるさまざまな WSDL 1.1 および WSDL 2.0 メッセージ交換パターン (MEP) の必須の MAP をリストしています。

表 13. WS-Addressing の必須のメッセージ・アドレッシング・プロパティ。

WS-Addressing MAP の名前	説明	WSDL 1.1 で必須	WSDL 2.0 で必須
<wsa:To>	予期されるメッセージ受信側のアドレス。	いいえ	いいえ
	値 <wsa:To> MAP が指定されない場合、CICS はアドレスを匿名 URI http://www.w3.org/2005/08/addressing/anonymous に設定します。匿名 URI は、DFHWS-URI コンテナで指定されるアドレスに要求が送信されることを示します。詳しくは、138 ページの『DFHWS-URI コンテナ』を参照してください。		

表 13. WS-Addressing の必須のメッセージ・アドレッシング・プロパティ。 (続き)

WS-Addressing MAP の名前	説明	WSDL 1.1 で必須	WSDL 2.0 で必須
<wsa:Action>	WS-Addressing アクション: 入力、出力、または障害。	以下の MEP で必須: 片方向 両方向 (要求) 両方向 (応答)	以下の MEP で必須: In-only Robust In-only (In) Robust In-only (Fault) In-out (In) In-out (Out) In-optional-out (In) In-optional-out (Out)
	WSDL 文書に <wsa:Action> MAP を明示的に定義するか、CICS が自動的に生成するように設定 することができます。詳しくは、295 ページの『WSDL 文書での <wsa:Action> プロパティの 定義』を参照してください。		
<wsa:From>	メッセージの発信元となっ たエンドポイント。	いいえ	いいえ
	この値は必須ではありません。		
<wsa:ReplyTo>	メッセージの応答を受け取 ることが予期される受信側 のエンドポイント。	いいえ	いいえ
	値が <wsa:ReplyTo> MAP のアドレス・エレメントに設定されない場合、アドレスは匿名 URI http://www.w3.org/2005/08/addressing/anonymous に設定されます。匿名 URI は、応答が要求側に 送り戻されることを示します。		
<wsa:FaultTo>	メッセージに関連した障害 を受け取ることが予期され る受信側のエンドポイン ト。	いいえ	いいえ
	値が <wsa:FaultTo> MAP のアドレス・エレメントに指定されない場合、CICS はこのアドレス を <wsa:ReplyTo> MAP のアドレス・エレメントと同じ値に設定します。 要求側が妥当性検査の障害を発生させる不正な MAP を作成する場合、CICS は障害メッセージ を <wsa:FaultTo> MAP で指定されたアドレスではなく要求側に送り戻します。		
<wsa:MessageID>	固有のメッセージ ID。	以下の MEP で必須: 両方向 (要求)	以下の MEP で必須: Robust In-only (In) In-out (In) In-optional-out (In)
	応答を予期する要求メッセージ、および応答メッセージに関して、CICS は実行時に、 <wsa:MessageID> MAP に対して自動的に固有値を設定します。		

表 13. WS-Addressing の必須のメッセージ・アドレッシング・プロパティー。 (続き)

WS-Addressing MAP の名前	説明	WSDL 1.1 で必須	WSDL 2.0 で必須
<wsa:RelatesTo>	このメッセージと別のメッセージの関係を示す値のペア。このエレメントには関連するメッセージの <wsa:MessageID> を組み込み、オプションの属性は関係のタイプを伝送します。	以下の MEP で必須: 両方向 (応答)	以下の MEP で必須: Robust In-only (Fault) In-out (Out) In-optional-out (Out)
<wsa:RelatesTo> MAP は応答メッセージで必須です。メッセージの関係のタイプはオプションであり、デフォルトは http://www.w3.org/2005/08/addressing/reply になります。			

詳しくは、以下の W3C WS-Addressing 1.0 Metadata 仕様を参照してください:
<http://www.w3.org/TR/ws-addr-metadata>。

関連概念

299 ページの『メッセージ交換』

Web Services Addressing (WS-Addressing) は、片方向、両方向要求/応答、同期要求/応答、および非同期要求/応答のメッセージ交換をサポートしています。

関連資料

138 ページの『DFHWS-URI コンテナ』

DFHWS-URI は、サービスの URI を格納する、DATATYPE(CHAR) のコンテナです。

Web Services Addressing のセキュリティ

Web Services Addressing (WS-Addressing) を使って公衆網で伝送される通信を適切に保護し、通信のパーティー間で十分なレベルの信用を確立する必要があります。通信を保護するために、トランスポート・レベルのセキュリティ (例えば SSL や HTTPS) を使用することをお勧めします。

(SSL や HTTPS などの) トランスポート・レベルのセキュリティは、WS-Addressing 通信を確実に保護するための最も単純な方法です。トランスポート・レベルのセキュリティを使用できない場合、WS-Addressing メッセージ・アドレッシング・プロパティーに署名し、エンドポイント参照を暗号化することによって、メッセージを保護できます。

CICS では WS-Addressing メッセージ・アドレッシング・プロパティーを含むヘッダーを署名することはできず、エンドポイント参照の暗号化も不可能です。ただし、CICS では着信メッセージの署名を検証したり、暗号化されたヘッダーを暗号化解除することは可能です。通信を保護するために署名と暗号化を使用したい場合には、外部のセキュリティ・ゲートウェイ (例えば IBM Websphere DataPower[®] XML Security Gateway) を使用する必要があります。

IBM Websphere DataPower XML Security Gateway の詳細情報については、<http://www-01.ibm.com/software/integration/datapower/xs40/> を参照してください。

Web Services Addressing の例

この例は、メッセージの送信に Web Services Addressing を使用する企業に対して顧客がオーダーを出すときに生じるプロセスに関する概要を示します。

電子部品を販売するある国際企業は、業務に Web Services Addressing を使用しています。この企業のインフラストラクチャーは、Ordering Client、Distribution Service のグループ、Fulfilment Service、および Configuration Service から構成されます。

WS-Addressing を使用することは、企業にとって以下の利点があります。

- WS-Addressing には、メッセージ転送のためのトランスポートに依存しないメカニズムが備えられています。これにより、異なるプラットフォーム上で実行される Web サービス間の相互運用性が実現されます。この例では、企業が所有する配布サービスがさまざまなプラットフォームで実行されます。WS-Addressing を使用することで、異なるプラットフォーム間の相互運用性が簡単になります。Web サービスのリクエスターとプロバイダーは、メッセージを交換するサービスが実行されるプラットフォームを意識する必要がないからです。
- WS-Addressing を使用して、<wsa:ReplyTo> MAP 内の EPR を更新することにより、応答メッセージの宛先を変更できます。この例では、Fulfilment Service がメッセージの転送先の Distribution Service を選択したときに、応答メッセージの宛先を変更します。

この例の企業にはいくつかの異なる国々に複数の配布センターがあり、各配布センターが Distribution Service によって表されて、Configuration Service に登録されています。

Fulfilment Service は、例えば要求されたアイテムの在庫や顧客から Distribution Center の距離などのさまざまな要素に基づいて、オーダーを処理するために最適な Distribution Service を選択します。

アドレッシング情報は Configuration Service との間で受け渡されます。

Configuration Service は、使用可能なサービスのアドレスをエンドポイント参照の形式で保管します。新規のサービスは、WSAEPR CREATE コマンドを使用して EPR を作成し、その EPR を Configuration Service に送信することにより、Configuration Service に登録します。Configuration Service は EPR を XML のブロックの形式で必要とするため、DFHWS2LS 上の **WSADDR-EPR-ANY** パラメーターを **TRUE** に設定する必要があります。 **WSADDR-EPR-ANY=TRUE** オプションを使用して、CICS が EPR を <xsd:any> エレメントとして扱うように指示します。CICS は、実行時にそれを言語構造に変換するのではなく、コンテナに入れる必要があります。

これらのサービスが対話する方法は、以下の図に示されています。図は、このタスクからは除外されているがビジネス・アプリケーションに関係する可能性がある他のサービスを示しています。これには以下のものがあります。

- 他のそれぞれのサービスによってオーダーの状況が更新されるトラッキング・サービス。
- 発生する障害メッセージを処理するための問題解決サービス。

- Ordering Client に送信される応答メッセージを扱うための Ordering Client コールバック・サービス。

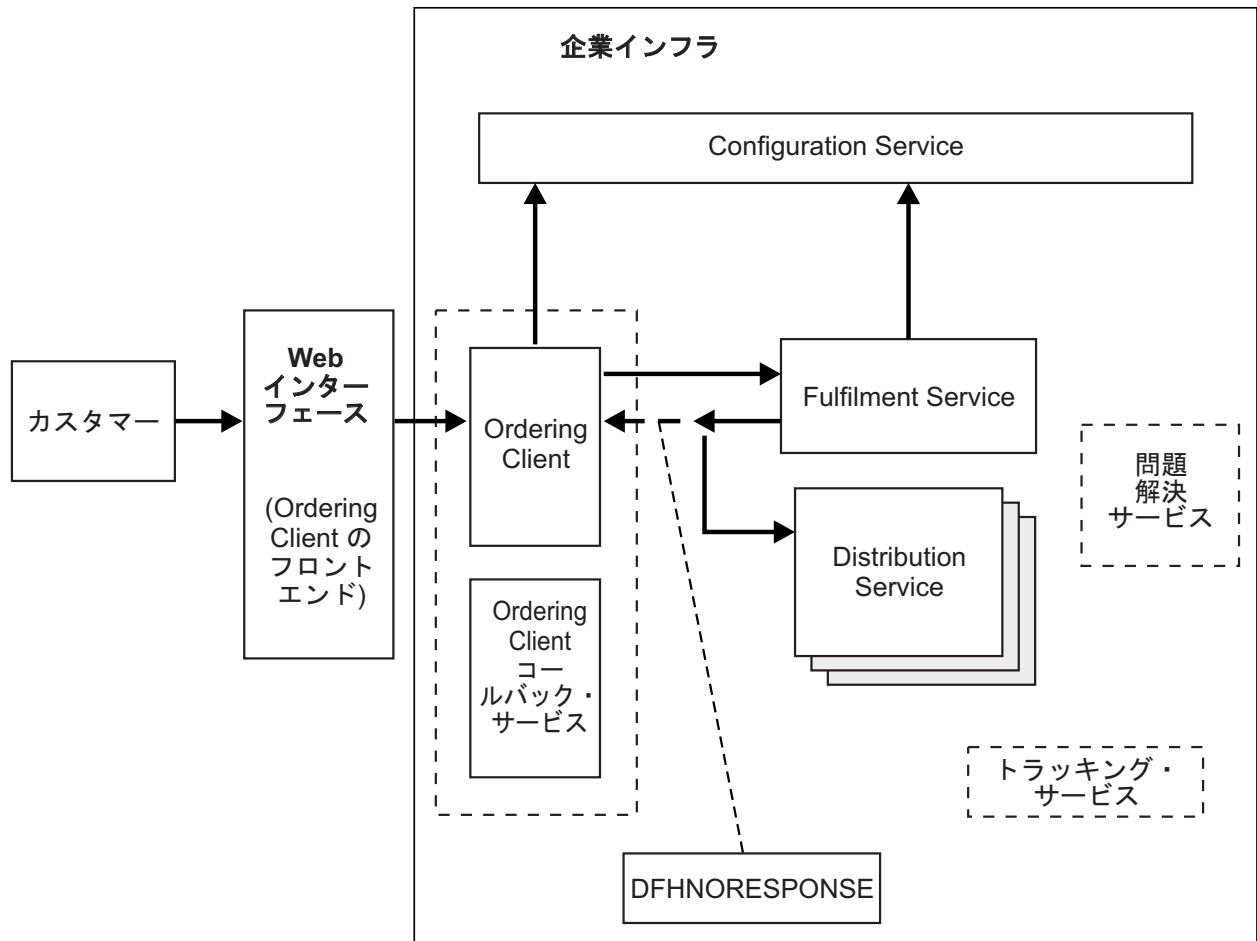


図 27. 企業のインフラストラクチャー

以下のステップは、顧客がオーダーを出す時点からそのオーダーが処理される時点までのプロセスを説明しています。

- 顧客が企業にオーダーを出します。
 - 顧客は Ordering Client のフロントエンドである企業の Web サイトにオーダーを出します。
 - Ordering Client は、顧客の連絡先の詳細をオーダーの一部として取得します。
 - Ordering Client は Web インターフェイスを介して確認メッセージおよび固有のオーダー参照を顧客に戻します。
- Ordering Client はオーダー要求を Fulfilment Service に送信します。
 - Ordering Client が Fulfilment Service の EPR をまだ認識していない場合、Configuration Service からそれを要求します。Ordering Client が Configuration Service から Fulfilment Service の EPR を要求する際に関係するプロセスについては、<wsa:To>の例のセクションで詳しく説明されます。

- b. Ordering Client は INVOKE SERVICE コマンドを Fulfilment Service に対して発行します。WS-Addressing はメッセージを、要求アドレッシング・コンテキストで EPR によって指定されたアドレスに経路指定します。
3. Fulfilment Service は、オーダーを処理するための Distribution Service を選択し、応答メッセージをそのサービスに転送します。
 - a. Fulfilment Service は WSACONTEXT GET コマンドを使用して、オーダー参照および他のアドレッシング・プロパティをアドレッシング・コンテキストから抽出します。
 - b. Fulfilment Service は最適な Distribution Service を Configuration Service から選択します。
 - c. <wsa:ReplyTo> EPR がアドレッシング・コンテキストに追加されます。


```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Address>http://www.example.ibm.com/DistributionService</wsa:Address>
</wsa:EndpointReference>
```

Fulfilment Service は WSACONTEXT BUILD コマンドを使用して、以下のよう
に選択された Distribution Service の ReplyTo EPR を要求アドレッシング
・コンテキストに追加します。
 - d. Fulfilment Service は WSACONTEXT BUILD コマンドを繰り返し使用して、
オーダー参照や他の情報を要求アドレッシング・コンテキストに追加しま
す。
 - e. DFHNORESPONSE コンテナが Ordering Client パイプラインに追加され
て、応答を受け取らないことを Ordering Client に知らせます。応答メッセ
ージは、要求メッセージの形式で Distribution Service に転送されます。
4. Distribution Service は、転送された応答メッセージを受け取り、オーダーを処理
します。
 - a. Distribution Service は WSACONTEXT GET コマンドを使用して、オーダー
参照およびアドレッシングの詳細を要求アドレッシング・コンテキストから
抽出します。
 - b. Distribution Service はオーダーを処理します。

<wsa:To> の例

1. Ordering Client は、メッセージの送信先とするサービスの EPR を Configuration
Service から要求します。この例では、Ordering Client は Fulfilment Service の
EPR を要求します。
2. Configuration Service は、次のように応答メッセージの作成および送信を行いま
す。
 - a. Configuration Service は、WSAEPR CREATE API コマンドを使用して、
Fulfilment Service の要求された <wsa:To> EPR を作成します (EXEC CICS
WSAEPR CREATE)。
 - b. Configuration Service は WSAEPR CREATE コマンドの出力をコンテナに
書き込みます (EXEC CICS PUT CONTAINER(work-cont))。
 - c. Configuration Service はコンテナ名を myEpr-xml-contエレメントにコピー
します (MOVE work-cont TO myEpr-xml-cont)。

- d. Configuration Service は、応答メッセージを Ordering Client に送信します。このメッセージには、myEpr-xml-cont コンテナによって指定されたコンテナの内容が含まれています。この例では、work-cont コンテナの内容は、<wsa:myEpr> エレメント内の Ordering Client に送信されます。

```
<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>
```

図 28 は、Ordering Client と Configuration Service の間の要求/応答メッセージの交換が示されています。このメッセージ交換には、2 つの標準的な Web サービス・パイプラインが関係します。

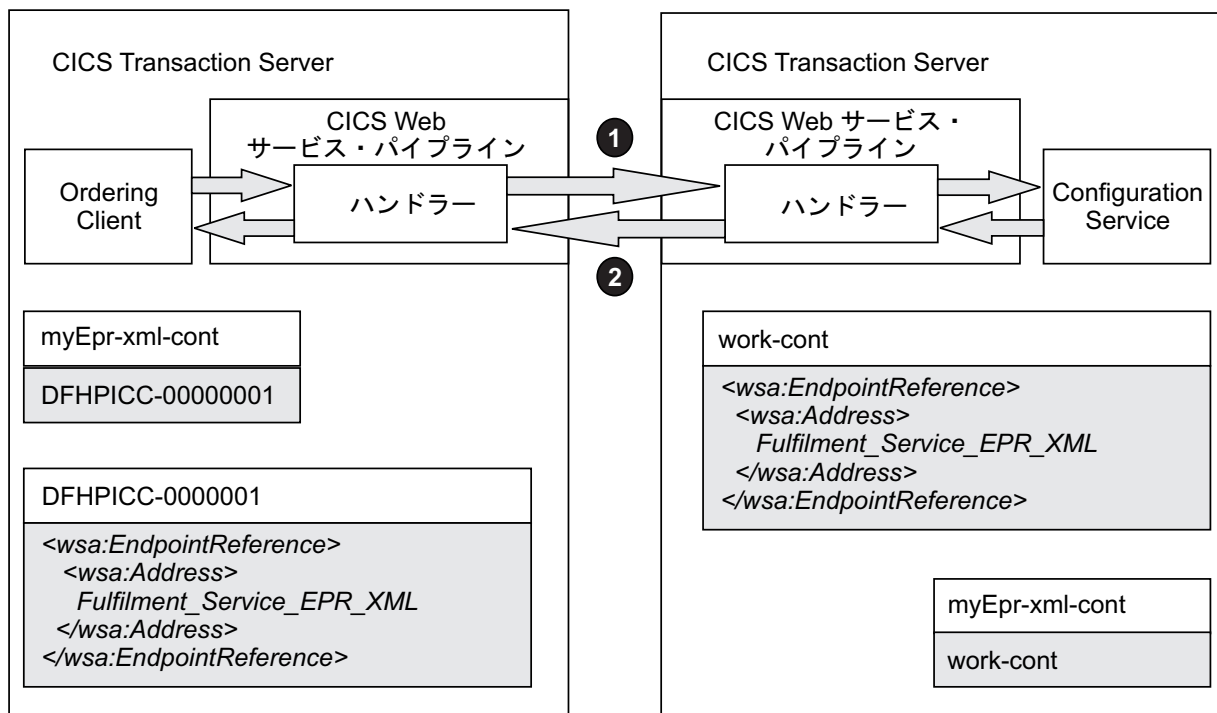


図 28. Ordering Client と Configuration Service の間の要求/応答メッセージの交換

3. Ordering Client は、応答メッセージを受け取り、<wsa:To> EPR を作成し、要求を Fulfilment Service に送信します。
 - a. Ordering Client は <wsa:To> EPR データを応答メッセージから抽出します。
 - b. CICS は固有のコンテナ（この例では DFHPICC-00000001 コンテナ）に <wsa:To> EPR データを取り込みます。

- c. CICS はコンテナの名前 (この例では DFHPICC-00000001) を myEpr-xml-cont エlement にコピーします。
- d. Ordering Client は myEpr-xml-cont Element によって指定されるコンテナの内容を読み取り、それを WSACONTEXT BUILD API コマンドに入力として提供します。WSACONTEXT BUILD コマンドはこの入力を使用して、Fulfilment Service の <wsa:To> EPR を作成します。
- e. Ordering Client は、パイプライン処理を開始する INVOKE SERVICE コマンドを発行します。
- f. アウトバウンド・パイプライン上の CICS Web サービスのアドレッシング・ハンドラー DFHWSADH は <wsa:To> EPR をアドレスおよびオプションの一連の参照パラメーターに変換し、Fulfilment Service に送信される SOAP 要求メッセージのヘッダーにそれを置きます。

```
<env:Header>
  <wsa:To>http://example.ibm.com/Fulfilment_Service</wsa:To>
</env:Header>
```

図 29 は、Ordering Client から Fulfilment Service への要求を示します。この要求には、CICS Web サービスのアドレッシング・ハンドラー DFHWSADH を組み込む Web サービス・パイプラインが関係します。

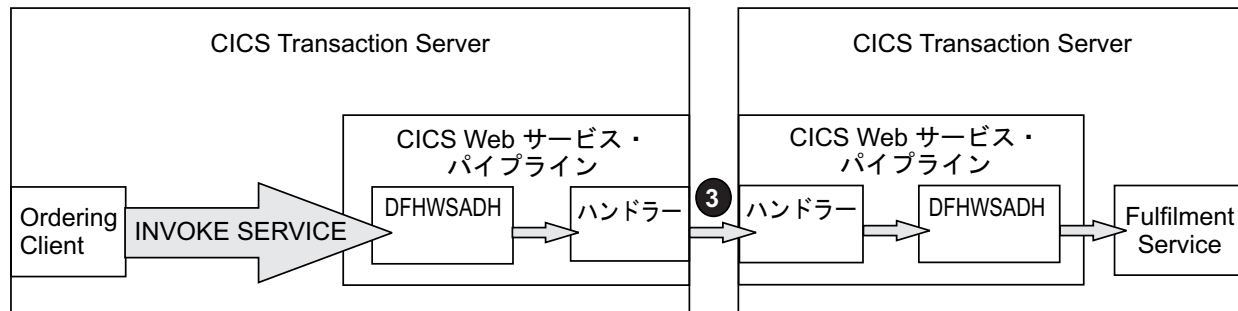


図 29. Ordering Client から Fulfilment Service への要求

関連資料

『Web Services Addressing の用語』

Web Services Addressing (WS-Addressing) のサポートについて説明する際に使用される用語。

Web Services Addressing の用語

Web Services Addressing (WS-Addressing) のサポートについて説明する際に使用される用語。

アドレッシング・コンテキスト (addressing context)

WS-Addressing メッセージ・アドレッシング・プロパティ (MAP) が SOAP 要求メッセージに送信される前と SOAP 要求メッセージおよび応答メッセージから受信された後にそれを保管するアドレッシング・コンテキスト。

エンドポイント参照 (EPR) (endpoint reference (EPR))

メッセージを Web サービスに経路指定するのに使用されるアドレッシング情報を含む XML 構造。このアドレッシング情報には、メッセージの宛先アドレス、アプリケーションが使用するオプションの参照パラメーター、およびオプションのメタデータが含まれます。

メッセージ・アドレッシング・プロパティ (MAP) (message addressing property (MAP))

固有のメッセージ ID、メッセージの宛先、およびメッセージのエンドポイント参照など、特定の Web サービス・メッセージに関するアドレッシング情報を伝達する XML エlement。

第 13 章 Web サービスを保護するためのサポート

CICS Transaction Server for z/OS は、SOAP メッセージを保護できるいくつかの関連仕様に対するサポートを提供します。

Web Services Security (WSS): SOAP Message Security 1.0 仕様には、セキュリティー・トークン とデジタル署名 を使用した SOAP メッセージの保護と認証について説明されています。WSS: Soap Message Security 1.0 仕様を参照してください。

Web Services Security は、メッセージが不正に開示されたり、不正または気付かれずに変更されたりしないようにすることによって、SOAP メッセージのプライバシー と保全性 を保護します。WSS は、メッセージ内の XML エlement をデジタル署名および暗号化することでこのような保護を提供します。保護できる Element は、本体または本体やヘッダー内の Element です。SOAP メッセージ内の異なる Element に対して異なるレベルの保護を適用することができます。

Web Services Trust Language 仕様は、セキュリティー・トークンを要求して発行するためのフレームワークを提供し、Web サービス・リクエストと Web サービス・プロバイダー間の信頼関係を管理することによって、Web Services Security をさらに拡張します。SOAP メッセージの認証をこのように拡張することによって、Web サービスは、信頼における第三者機関を使用して、さまざまなタイプのセキュリティー・トークンを検証および交換することができます。この第三者機関は、*Security Token Service (STS)* と呼ばれます。Web Services Trust Language の詳細については、WS-Trust Language 仕様を参照してください。

CICS Transaction Server for z/OS は、CICS 提供のパイプラインのセキュリティー・ハンドラーを使用することによって、これらの仕様をサポートします。

- アウトバウンド・メッセージでは、CICS は SOAP 本体全体にデジタル署名して暗号化するためのサポートを提供します。また CICS は、STS を使用して、さまざまなタイプのセキュリティー・トークンでユーザー名トークンを交換することができます。
- インバウンド・メッセージでは、CICS は、本体または本体やヘッダーの Element が暗号化されているかデジタル署名されているメッセージをサポートします。また CICS は、STS を使用してセキュリティー・トークンを交換して検証することもできます。

CICS は、個別の Trust クライアント・インターフェースを提供して、CICS セキュリティー・ハンドラーを使用せずに STS とデータをやり取りすることもできます。

Web Services Security を実装するための前提条件

Web Services Security を実装するには、次のような更新を CICS 領域に適用する必要があります。IBM XML Toolkit for z/OS v1.10 をインストールし、APAR OA14956 を適用し、3 つのライブラリーを DFHRPL 連結に追加します。

このタスクについて

Web Services Security を実装する前に、以下の手順を完了してください。

1. 無料の IBM XML Toolkit for z/OS v1.10 をインストールします。これは、サイト <http://www.ibm.com/servers/eserver/zseries/software/xml/> からダウンロードできます。バージョン 1.10 をインストールする必要があります。それより後のバージョンは、CICS の Web Services Security サポートでは機能しません。
2. ICSF APAR OA14956 が CICS 領域にまだインストールされていない場合は、これを適用します。
3. 次のライブラリーを DFHRPL 連結に追加します。
 - *hlq.SIXMLOD1*。 *hlq* は、XML ツールキットの高位修飾子です。
 - *hlq.SCEERUN*。 *hlq* は、言語環境の高位修飾子です。
 - *hlq.SDFHWSLD*。 *hlq* は、CICS 領域の高位修飾子です。

最初の 2 つのライブラリーには、実行時にセキュリティー・ハンドラーに必要な DLL が含まれています。IXM4C57 は XML ツールキットによって提供され、*hlq.SIXMLOD1* にあります。C128N は 言語環境ランタイムによって提供され、*hlq.SCEERUN* にあります。

hlq.SDFHWSLD ライブラリーを使用すると、CICS は DFHWSSE1 と DFHWSXXX の Web Services Security モジュールを検索できるようになります。

4. **EDSALIM** システム初期化パラメーターの値を増やさなければならないことがあります。ロードされる 3 つの DLL は、約 15 MB の EDSA ストレージを必要とします。

タスクの結果

ライブラリーを指定していない場合は、次のメッセージが表示されます。

```
CEE3501S The module module_name was not found.  
(CEE3501S モジュール module_name が見つかりませんでした。)
```

module_name は、欠落しているライブラリーによって異なります。

Web サービスを保護するための計画

ご使用の Web サービスを保護するための最良の方法を判別します。CICS は、構成可能なセキュリティー・メッセージ・ハンドラーや、個別の Trust クライアント・インターフェースなど、多くのオプションをサポートしています。

このタスクについて

CICS は、Web サービスごとではなくパイプライン・レベルで、Web Services Security を実装します。以下の質問に答えて、セキュリティーを実装する最良の方法を判別してください。

1. パイプライン処理のパフォーマンスは重要ですか? WSS を使って Web サービスを保護する場合、パフォーマンスがかなり影響を受けます。

WSS を実装する主な利点は、SOAP メッセージの一部を暗号化することによって、一連の中間ノードを介してメッセージを送信できることです。これらの中間ノードはすべて、ルーティングまたは処理の決定を行うために SOAP ヘッダーを調べることができますが、メッセージの内容を表示することはできません。機密にすべきセクションだけを暗号化することには、次のような利点があります。

- 一連の中間プロセス内のすべてのノードで暗号化および暗号化解除が行われることによるオーバーヘッドが生じません。
- データの最終的な受信側からの理解を得られる場合に限り、信頼できないノードの公衆網を介して機密メッセージを送付することができます。

Web Services Security の使用に代わる方法として、SSL を使用してデータ・ストリーム全体を暗号化することができます。

2. Web Services Security を使用する場合、どのレベルのセキュリティが必要ですか？ このオプションは、メッセージ・ヘッダーがユーザー名およびパスワードを含む基本認証から、メッセージでのデジタル署名と暗号化の組み合わせまで、多岐にわたります。CICS セキュリティー・ハンドラーがサポートするオプションについては、『SOAP メッセージを保護するためのオプション』で説明しています。
3. CICS 提供のセキュリティ・ハンドラーは、要件を満たしていますか？ より高度なセキュリティ処理を実行する場合は、独自にカスタムのセキュリティ・ハンドラーを作成する必要があります。このハンドラーは、RACF によって直接か、または Security Token Service を使用して、メッセージの必要な認証を実行し、デジタル証明書および暗号化されたエレメントの処理を行う必要があります。詳しくは、328 ページの『カスタムのセキュリティ・ハンドラーの作成』を参照してください。
4. パイプラインに MTOM ハンドラーが含まれていますか？ パイプライン構成ファイルで、MTOM ハンドラーとセキュリティ・ハンドラーの両方を使用できるようにする計画の場合、MIME Multipart または Related メッセージはすべて、互換モードで処理されるため、セキュリティ・ハンドラーはメッセージ本文の XOP エレメントを構文解析できません。この処理は、パイプライン処理のパフォーマンスに、さらに影響を与える恐れがあります。

例

次のタスク

SOAP メッセージを保護するためのオプション

CICS では、SOAP メッセージの署名と暗号化の両方がサポートされるため、SOAP メッセージで送受信するデータに最適なセキュリティ・レベルを選択することができます。

以下のオプションの中から選択できます。

トラステッド認証

サービス・プロバイダー・パイプラインでは、CICS は、SOAP メッセージ・ヘッダーのユーザー名トークンを、信頼できるものとして受け入れることができます。この通常ユーザー名とパスワードを含むタイプのセキュリティ・トークンですが、この場合、パスワードは不要です。CICS は提

供されたユーザー名を信頼し、それを DFHWS-USERID コンテナに置きます。メッセージはパイプラインで処理されます。

サービス・リクエスターのパイプラインでは、CICS は、SOAP メッセージ・ヘッダーにパスワードがないユーザー名トークンを、サービス・プロバイダーに送信することができます。

基本認証

サービス・プロバイダー・モードでは、CICS は、インバウンド SOAP メッセージでの認証のために、SOAP メッセージ・ヘッダーのユーザー名トークンを受け入れることができます。このユーザー名とパスワードを含むタイプのセキュリティ・トークンです。CICS は、RACF などの外部セキュリティ・マネージャーを使用して、ユーザー名トークンを検証します。成功すると、ユーザー名はコンテナ DFHWS-USERID に置かれ、SOAP メッセージがパイプラインで処理されます。CICS がユーザー名トークンを検証できない場合は、SOAP 障害メッセージがサービス・リクエスターに戻されます。

パスワードを含むユーザー名トークンは、サービス・リクエスター・モードや、アウトバウンド SOAP メッセージではサポートされません。

HTTP 基本認証

サービス・プロバイダー・モードでは、CICS は、HTTP プロトコルでの基本認証情報を受け入れることができます。サービス要求元は URIMAP 定義を使用して資格情報 (ユーザー識別情報) がグローバル・ユーザー出口 XWBAUTH によって収集されることを指定します。XWBAUTH は要求に応じてこの情報を CICS に受け渡し、CICS は HTTP 許可ヘッダーの情報をサービス・プロバイダーに送信します。

拡張認証

サービス・プロバイダーおよびリクエスター・パイプラインでは、認証の目的で、Security Token Service (STS) によってセキュリティ・トークンを検証または交換できます。この認証により、CICS は、メッセージ・ヘッダーにセキュリティ・トークンのある、通常はサポートされないメッセージ (Kerberos トークンや SAML アサーションなど) の送受信を行うことができます。

インバウンド・メッセージの場合は、セキュリティ・トークンの検証または交換を選択できます。要求が、セキュリティ・トークンの交換である場合、CICS は、STS からユーザー名トークンを受け取る必要があります。アウトバウンド・メッセージの場合は、セキュリティ・トークンに関するユーザー名トークンの交換だけが可能です。

X.509 証明書による署名

サービス・プロバイダー・モードとサービス・リクエスター・モードでは、SOAP メッセージ・ヘッダーで X.509 証明書を提供して、認証のために SOAP メッセージの本文に署名することができます。このバイナリー・セキュリティ・トークンとして知られるタイプのセキュリティ・トークンです。インバウンド SOAP メッセージからのバイナリー・セキュリティ・トークンを受け入れるには、証明書に関連付けられた公開鍵を RACF などの外部セキュリティ・マネージャーにインポートして、**KEYRING** システム初期化パラメーターで指定された鍵リングに関連付ける必要があります。アウトバウンド SOAP メッセージでは、公開鍵を生成して、意図し

た受信側に発行されます。公開鍵の生成には、Integrated Cryptographic Service Facility (ICSF) が使用されます。

X.509 デジタル証明書に関連付けられたラベルを指定する場合は、次の文字は使用しないでください。

< > : ! =

また、ヘッダーに 2 番目の X.509 証明書を含めて、最初の証明書を使用して署名することができます。この 2 番目の証明書により、2 番目の X.509 証明書に関連付けられたユーザー ID を使用して CICS で作業を実行できるようになります。SOAP メッセージに署名するために使用する証明書は、トラステッド・ユーザー ID に関連付けられている必要があります。また、異なる ID (宣言 ID) に関連付けられたパスワードをトラステッド・ユーザー ID が持たなくても、この ID で作業を実行することを表明するために代理権限も必要です。

暗号化 サービス・プロバイダー・モードおよびサービス・リクエスター・モードでは、Triple-DES や AES などの対称アルゴリズムを使用して、SOAP メッセージの本文を暗号化することができます。対称アルゴリズムでは、データの暗号化と暗号化解除に同じ鍵が使用されます。この鍵は、対称鍵として知られています。この鍵はメッセージに含められ、意図した受信側の公開鍵と非対称鍵暗号化アルゴリズム RSA 1.5 との組み合わせを使用して暗号化されます。非対称アルゴリズムは複雑で、対称鍵を暗号化解除するのは困難であるため、この暗号化によってセキュリティが強化されます。また一方、SOAP メッセージの大部分は、より迅速に暗号化解除できる対称アルゴリズムで暗号化されるため、パフォーマンスが向上します。

インバウンド SOAP メッセージでは、SOAP 本体のエレメントを暗号化してから、SOAP 本体を全体として暗号化することができます。暗号化の種類は特に、機密データを含むエレメントに適していることがあります。CICS が 2 つのレベルで暗号化された SOAP メッセージを受信すると、CICS は両方のレベルを自動的に暗号化解除します。暗号化の種類は、アウトバウンド SOAP メッセージではサポートされていません。

CICS では、メッセージ・ヘッダーのみに暗号化されたエレメントを含み、SOAP 本体のエレメントは暗号化されていないインバウンド SOAP メッセージはサポートされません。

署名および暗号化

サービス・プロバイダー・モードおよびサービス・リクエスター・モードでは、SOAP メッセージの署名と暗号化の両方を選択することができます。CICS は必ず、最初に SOAP メッセージの本文に署名してから、暗号化します。この方法の利点は、メッセージの機密性と保全性の両方を確保できる点です。

Security Token Service を使用した認証

CICS は、Tivoli Federated Identity Manager などの Security Token Service (STS) と相互運用して、Web サービスのより高度な認証を提供することができます。

STS は、信頼のおける第三者機関として働き、Web サービス・リクエスターと Web サービス・プロバイダー間の信頼関係を仲介する Web サービスです。SSL

ハンドシェークでの認証局と同様の方法で、STS は、メッセージが示すクレデンシャルをリクエスターとプロバイダーが「信頼」できることを保証します。この信頼関係は、セキュリティー・トークンの交換によって表されます。STS は、これらのセキュリティー・トークンを発行、交換、および検証して信頼関係を確立し、さまざまな信頼ドメインからの Web サービス同士が正常に対話できるようにします。詳しくは、Web Services Trust Language 仕様の説明を参照してください。

CICS は Trust クライアントとして働き、2 つのタイプの Web サービス要求を STS に送信できます。要求の 1 つ目のタイプは、WS-Security メッセージ・ヘッダーのセキュリティー・トークンを検証することであり、要求の 2 つ目のタイプは、セキュリティー・トークンを別のタイプに交換することです。これらの要求により、多岐にわたる信頼ドメインからのさまざまなセキュリティー・トークン (SAML アサーションや Kerberos トークンなど) を含むメッセージを、CICS で送受信できるようになります。

CICS セキュリティー・ハンドラーを構成して、CICS が STS とデータをやり取りする方法を定義するか、独自のメッセージ・ハンドラーを作成して、個別に提供される Trust クライアント・インターフェースを使用することができます。選択したメソッドに関わらず、SSL を使用して CICS と STS の間の接続を保護することをお勧めします。

セキュリティー・ハンドラーで STS を起動する方法

CICS セキュリティー・ハンドラーは、パイプライン構成ファイルの情報を使用して、Web サービス要求を Security Token Service (STS) に送信します。送信される要求のタイプは、STS で実行するアクションによって異なります。

サービス・プロバイダー・パイプラインの場合

サービス・プロバイダー・パイプラインでは、セキュリティー・ハンドラーの構成方法に応じて、次のような 2 種類のアクションがセキュリティー・ハンドラーによってサポートされます。

- インバウンド・メッセージの WS-Security ヘッダーにある、セキュリティー・トークンの最初のインスタンスまたは特定タイプの最初のセキュリティー・トークンを検証するために、STS へ要求を送信します。
- インバウンド・メッセージの WS-Security ヘッダーにある、セキュリティー・トークンの最初のインスタンスまたは特定タイプの最初のセキュリティー・トークンを、CICS が理解できるセキュリティー・トークンに交換するために、STS へ要求を送信します。

セキュリティー・ハンドラーは、動的にパイプラインを作成し、Web サービス要求を STS に送信します。このパイプラインは、STS からの応答が受信されるまで存在し、その後削除されます。要求が成功した場合、STS は、ID トークンまたはトークンの妥当性の状況に戻します。セキュリティー・ハンドラーは、トークンを DFHWS-USERID コンテナに配置します。

STS でエラーが発生した場合、STS は SOAP 障害をセキュリティー・ハンドラーに戻します。その後セキュリティー・ハンドラーは、Web サービス・リクエスターに障害に戻します。

サービス・リクエスター・パイプラインの場合

サービス・リクエスター・パイプラインでは、セキュリティ・ハンドラーが要求できるのは、STS によってトークンを交換することのみです。パイプライン構成ファイルは、STS がセキュリティ・ハンドラーに対して発行する必要があるトークンのタイプを定義します。

要求が成功した場合、トークンは DFHWS-USERID に配置され、その後アウトバウンド・メッセージ・ヘッダーに組み込まれます。STS でエラーが発生した場合、STS は SOAP 障害をセキュリティ・ハンドラーに戻します。その後セキュリティ・ハンドラーは、パイプラインを介して、障害を Web サービス・リクエスター・アプリケーションに戻します。

セキュリティ・ハンドラーは、パイプラインに関する 1 つのタイプのアクションだけを STS に要求できます。また、アウトバウンド要求メッセージに関する 1 つのタイプのトークンだけを交換でき、WS-Security メッセージ・ヘッダー内の最初のトークン (最初のインスタンス、または特定のタイプの最初のインスタンス) だけを限定的に処理します。これらのオプションは、STS の使用に関する一般的なシナリオのほとんどをカバーしますが、インバウンドおよびアウトバウンド・メッセージを扱う際に必要となる処理を提供しない場合もあります。

より限定的な処理を準備してインバウンド・メッセージ・ヘッダーで多くのトークンを扱うようにする場合、または複数のタイプのトークンをアウトバウンド・メッセージと交換する場合は、Trust クライアント・インターフェースを使用することをお勧めします。このインターフェースを使用すると、カスタムのメッセージ・ハンドラーを作成して、独自の Web サービス要求を STS に送信できます。

Trust クライアント・インターフェース

Trust クライアント・インターフェースによって、セキュリティ・ハンドラーを使用せずに、直接 Security Token Service (STS) と対話することができます。この方法を使用すると、セキュリティ・ハンドラーを使った処理よりも高度な処理をトークンに対して柔軟に実行できます。

Trust クライアント・インターフェースは、CICS 提供のプログラム DFHPIRT を拡張したものです。このプログラムは通常、CICS Web サービス・アシスタントを使用して Web サービス・リクエスター・アプリケーションが導入されていない場合に、パイプラインを開始するために使用されます。ただし、STS に対する Trust クライアント・インターフェースとして機能することもできます。

Trust クライアント・インターフェースを起動するには、メッセージ・ハンドラーまたはヘッダー処理プログラムから DFHPIRT にリンクして、DFHWSTC-V1 と呼ばれるチャンネルおよびセキュリティ・コンテナ一式を渡します。これらのコンテナを使用することで、柔軟性が高められ、STS の検証または実行アクションのいずれかを要求し、交換するトークンのタイプを選択し、メッセージ・ヘッダーの該当するトークンを渡すことができます。DFHPIRT は動的にパイプラインを作成し、セキュリティ・コンテナからの Web サービス要求を構成し、それを STS に送信します。

DFHPIRT は、STS から応答が戻るのを待機し、それを DFHWS-RESTOKEN コンテナに入れてメッセージ・ハンドラーに渡します。STS でエラーが発生した場

合、STS は SOAP 障害を戻します。 DFHPIRT は、その障害を DFHWS-STSFault コンテナに入れ、パイプライン内のリンクしているプログラムに戻します。

サービス・プロバイダーおよびサービス・リクエスター・パイプラインでセキュリティ・ハンドラーを使用できるようにしなくても、Trust クライアント・インターフェースを使用できます。あるいは、セキュリティ・ハンドラーに追加して、Trust クライアント・インターフェースを使用することもできます。

SOAP メッセージへの署名

インバウンド・メッセージでは、CICS は SOAP 本体内のエレメントおよび SOAP ヘッダー・ブロックのデジタル署名をサポートしています。アウトバウンド・メッセージでは、CICS は SOAP 本体内のすべてのエレメントに署名します。

SOAP メッセージは、<Envelope> エレメントから構成される XML 文書です。この <Envelope> エレメントには、オプションの <Header> エレメントと必須の <Body> エレメントが格納されています。

WSS: SOAP Message Security 仕様では、<Header> と <Body> の内容にエレメント・レベルで署名することができます。つまり、あるメッセージでは、署名するエレメントと署名しないエレメントがあり、別の署名または別のアルゴリズムを使用して署名することができます。例えば、オンライン購入アプリケーションで 사용되는 SOAP メッセージでは、受注を確認するエレメントは法的状況を持つことがあるため、これらのエレメントには署名するのが適しています。ただし、メッセージ全体への署名にかかるオーバーヘッドを避けるために、他の情報は署名されていないままでも支障はありません。

インバウンド・メッセージでは、セキュリティ・メッセージ・ハンドラーが、SOAP <Header> および <Body> 内の個々のエレメントのデジタル署名を検証することができます。

- <Header> で検出される署名済みエレメント。
- SOAP <Body> 内の署名済みエレメント。署名済みの本体を受け入れるようハンドラーが構成されている場合、CICS は本体が署名されていない SOAP メッセージをすべて拒否して、SOAP 障害を発行します。

アウトバウンド・メッセージでは、セキュリティ・メッセージ・ハンドラーが署名できるのは、<Body> だけで、<Header> には署名しません。アルゴリズム、および本体への署名に使用される鍵は、ハンドラーの構成情報で指定されます。

署名アルゴリズム

CICS は、XML Signature 仕様で必要な署名アルゴリズムをサポートします。それぞれのアルゴリズムは、汎用リソース ID (URI) で識別されます。

3. メッセージ・ダイジェストの値。
4. その後、ダイジェスト値はユーザーの秘密鍵で暗号化され、署名値としてここに含められます。
5. 署名を検証するために使用される公開鍵を含むバイナリー・セキュリティー・トークンを参照します。

暗号化された SOAP メッセージの CICS サポート

インバウンド・メッセージでは、CICS は、SOAP 本体内の暗号化済みエレメント、および本体も暗号化された暗号化済みの SOAP ヘッダー・ブロックを暗号化解除することができます。アウトバウンド・メッセージでは、CICS は SOAP 本体全体を暗号化します。

SOAP メッセージは、<Envelope> エレメントから構成される XML 文書です。この <Envelope> エレメントには、オプションの <Header> エレメントと必須の <Body> エレメントが格納されています。

WSS: *SOAP Message Security* 仕様では、<Header> エレメントの内容の一部と <Body> エレメントのすべての内容をエレメント・レベルで暗号化することができます。つまり、あるメッセージでは、個々のエレメントを異なるレベルで暗号化したり、別のアルゴリズムを使用して暗号化したりすることができます。例えば、オンライン購入アプリケーションで使用される SOAP メッセージでは、個々のクレジット・カードの詳細が機密のままになるように、詳細を暗号化するのが適しています。ただし、メッセージ全体の暗号化にかかるオーバーヘッドを避けるために、一部の情報は安全性の低い（しかし高速な）アルゴリズムを使用して暗号化して、他の情報は暗号化されていないままでも支障はありません。

インバウンド・メッセージでは、CICS 提供のセキュリティー・メッセージ・ハンドラーが、SOAP <Body> 内の個々のエレメントを暗号化解除して、SOAP 本体も暗号化されている場合は SOAP <Header> 内のエレメントを暗号化解除することができます。セキュリティー・メッセージ・ハンドラーは、以下のエレメントを常に暗号化解除します。

- <Header> エレメント内に検出される各エレメント（見つかった順序で）。
- SOAP <Body> エレメント内のエレメント。暗号化された <Body> を含まない SOAP メッセージを拒否する場合は、<expect_encrypted_body> エレメントを使用して暗号化された本体を要求するようハンドラーを構成します。

アウトバウンド・メッセージでは、セキュリティー・メッセージ・ハンドラーがサポートするのは、SOAP <Body> の内容の暗号化だけです。<Header> エレメント内のエレメントは暗号化されません。セキュリティー・メッセージ・ハンドラーが <Body> エレメントを暗号化すると、本体内のすべてのエレメントが、同じアルゴリズムと同じ鍵を使用して暗号化されます。アルゴリズム、および鍵に関する情報は、ハンドラーの構成情報で指定されます。

暗号化アルゴリズム

CICS は、XML 暗号化仕様で必要な暗号化アルゴリズムをサポートします。それぞれのアルゴリズムは、汎用リソース ID (URI) で識別されます。

1. バイナリー・セキュリティー・トークンには、base64binary エンコードの X.509 証明書が含まれています。このエンコードには、対称鍵の暗号化に使用された公開鍵が格納されています。
2. 対称鍵の暗号化に使用されたアルゴリズムを提示します。
3. 対称鍵の暗号化に使用された公開鍵を含むバイナリー・セキュリティー・トークンを参照します。
4. メッセージの暗号化に使用された暗号化済みの対称鍵。
5. メッセージの暗号化に使用された暗号化アルゴリズム。
6. 暗号化されたメッセージ。

Web Services Security に合わせた RACF の構成

アウトバウンド SOAP メッセージに署名して暗号化するための公開鍵と秘密鍵のペアおよび X.509 証明書を作成して、署名および暗号化されたインバウンド SOAP メッセージを認証して暗号化解除するには、RACF などの外部セキュリティー・マネージャーを構成する必要があります。

始める前に

この作業を実行するには、その前に、CICS で作業するよう RACF をセットアップしておく必要があります。 **DFLTUSER**、**KEYRING**、および **SEC=YES** の各システム初期化パラメーターを、Web サービス・パイプラインが格納された CICS 領域を指定します。

1. 署名されたインバウンド SOAP メッセージを認証するには、次のようにします。
 - a. X.509 証明書を ICSF 鍵として RACF にインポートします。
 - b. RACDCERT コマンドを使用して、**KEYRING** システム初期化パラメーターで指定した鍵リングに証明書を添付します。

```
RACDCERT ID(userid1)
CONNECT(ID(userid2) LABEL('label-name') RING(ring-name))
```

ここで、

- *userid1* は、鍵リングのデフォルトのユーザー ID であるか、他のユーザー ID の鍵リングに証明書を添付する権限を持っています。
 - *userid2* は、証明書に関連付けるユーザー ID です。
 - *label-name* は、証明書の名前です。
 - *ring-name* は、**KEYRING** システム初期化パラメーターで指定された鍵リングの名前です。
- c. オプション: 宣言 ID を使用する場合は、証明書に関連付けられたユーザー ID が、作業を他のユーザー ID の下で実行できる代理権限を持っていることを確認します。また、SOAP メッセージ・ヘッダーに含まれる追加の証明書も忘れずに RACF にインポートします。

SOAP メッセージのヘッダーには、証明書または証明書への参照のいずれかが入ったバイナリー・セキュリティー・トークンを含めることができます。この参照は、KEYNAME (RACF での証明書ラベル)、ISSUER と SERIAL 番号の組み合

わせ、または SubjectKeyIdentifier です。 SubjectKeyIdentifier が RACF における証明書の定義で属性として指定された場合、CICS が認識できるのは SubjectKeyIdentifier だけです。

2. アウトバウンド SOAP メッセージに署名するには、次のようにします。
 - a. 次の RACDCERT コマンドを使用して、X.509 証明書および公開鍵と秘密鍵のペアを作成します。

```
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')
```

ここで、*userid2* は、証明書に関連付けるユーザー ID です。証明書の *label-name* 値を指定する場合は、次の文字は使用しないでください。

< > : ! =

- b. **KEYRING** システム初期化パラメーターで指定した鍵リングに証明書を添付します。RACDCERT コマンドを使用します。
 - c. 証明書をエクスポートして、SOAP メッセージの意図した受信側に発行します。

CICS が、署名を検証するために、意図した受信側の SOAP メッセージ・ヘッダーのバイナリー・セキュリティ・トークンに X.509 証明書を自動的に含めるように、パイプライン構成ファイルを編集することができます。

3. 暗号化されたインバウンド SOAP メッセージを暗号化解除するには、SOAP メッセージに、鍵ペアの一部である公開鍵が含まれている必要があります。この場合、秘密鍵は CICS で定義されます。
 - a. 暗号化のために、RACF で公開鍵と秘密鍵のペアおよび証明書を生成します。鍵ペアと証明書は、ICSF を使用して生成する必要があります。
 - b. **KEYRING** システム初期化パラメーターで指定した鍵リングに証明書を添付します。RACDCERT コマンドを使用します。
 - c. 証明書をエクスポートして、暗号化解除する SOAP メッセージの生成プログラムに発行します。

その後、SOAP メッセージの生成プログラムは、公開鍵を含む証明書をインポートして、これを使用して SOAP メッセージを暗号化することができます。

SOAP メッセージのヘッダーには、公開鍵またはこの公開鍵への参照のいずれかが入ったバイナリー・セキュリティ・トークンを含めることができます。この参照は、KEYNAME、ISSUER と SERIAL 番号の組み合わせ、または SubjectKeyIdentifier です。SubjectKeyIdentifier が RACF における公開鍵の定義で属性として指定された場合、CICS が認識できるのは SubjectKeyIdentifier だけです。

4. アウトバウンド SOAP メッセージを暗号化するには、次のようにします。
 - a. 暗号化に使用する公開鍵を含む証明書を ICSF 鍵として RACF にインポートします。意図した受信側が SOAP メッセージを暗号化解除するには、公開鍵に関連付けられた秘密鍵が必要です。

- b. **KEYRING** システム初期化パラメーターで指定した鍵リングに、公開鍵を含む証明書を添付します。 **RACDCERT** コマンドを使用します。

CICS は、証明書の公開鍵を使用して、SOAP 本体を暗号化し、SOAP メッセージ・ヘッダー内にバイナリー・セキュリティ・トークンとして公開鍵を含む証明書を送信します。公開鍵は、パイプライン構成ファイルで定義されます。

次のタスク

アウトバウンド・メッセージに署名して暗号化する上記の構成では、使用される証明書が CICS 領域のユーザー ID によって所有されている必要があります。 RACF では、証明書の所有者だけが秘密鍵 (署名または暗号化のプロセスで使用される) を抽出できるため、証明書は CICS 領域ユーザー ID によって所有される必要があります。

CICS が所有していない証明書を使用してメッセージの署名または暗号化を行う必要がある場合 (例えば、各システムが異なる領域ユーザー ID を持つ複数の CICS システムによって単一の証明書が共有される場合)、以下の条件が当てはまらなければなりません。

1. 以下の z/OS リリースのうちのいずれかを使用している必要があります。
 - z/OS 1.9 以降
 - z/OS 1.8 (PTF UA37039 を適用済み)
 - z/OS 1.7 (PTF UA37038 を適用済み)
2. 証明書は、PERSONAL 使用オプションを指定してその鍵リングに接続されている必要があります。
3. 証明書が USER 証明書である場合、証明書を使用する CICS 領域ユーザー ID に、RDATA LIB クラスの <ringOwner>.<ringName>.LST リソースに対する READ または UPDATE 権限が必要です。
4. RACLIST オプションを使用して RDATA LIB クラスを活動化しておく必要があります。

CICS は RACF R_data lib 呼び出し可能サービスを使用して、証明書から秘密鍵を抽出します。詳しくは、「z/OS Security Server RACF 呼び出し可能サービス」ガイドを参照してください。

Web Services Security に合わせたパイプラインの構成

Web Services Security (WSS) をサポートするようにパイプラインを構成するには、パイプライン構成ファイルにセキュリティ・ハンドラーを追加する必要があります。説明されているように、CICS 提供のセキュリティ・ハンドラーを使用するか、独自に作成することができます。

始める前に

CICS 提供のセキュリティ・ハンドラーを定義する前に、WSS に関する構成情報の追加先となるパイプライン構成ファイルを指定または作成する必要があります。

1. <wsse_handler> エレメントをパイプラインに追加します。 サービス・プロバイダー・パイプラインまたはサービス・リクエスター・パイプライン内の

<service_handler_list> エlementにハンドラーを含める必要があります。
次のElementをコーディングします。

```
<wsse_handler>
  <dfhwsse_configuration version="1">

    </dfhwsse_configuration>
</wsse_handler>
```

<dfhwsse_configuration> Elementは、構成内の他のElementのコンテナです。

2. オプション: <authentication> Elementをコーディングします。

- サービス・リクエスター・パイプラインでは、<authentication> Elementが、アウトバウンド SOAP メッセージのセキュリティ・ヘッダーで使用する必要がある認証のタイプを指定します。
- サービス・プロバイダー・パイプラインでは、このElementが、CICS がインバウンド SOAP メッセージでセキュリティ・トークンを使用して、処理が行われるユーザー ID を決定するかどうかを指定します。
 - a. **trust** 属性をコーディングして、宣言 ID を使用するかどうか、およびサービス・プロバイダーとサービス・リクエスター間の信頼関係の性質を指定します。 **trust** 属性について詳しくは、100 ページの『<authentication>Element』を参照してください。
 - b. オプション: **trust=none** を指定した場合は、**mode** 属性をコーディングして、メッセージで見つかったクレデンシャルの処理方法を指定します。**mode** 属性について詳しくは、100 ページの『<authentication>Element』を参照してください。
 - c. 以下のElementを <authentication> Element内にコーディングします。
 - 1) オプションの、空の <suppress/> Element。

このElementがサービス・プロバイダー・パイプラインに指定される場合、ハンドラーは、作業が行われるユーザー ID を決定するメッセージ内のどのセキュリティ・トークンの使用も試みません。

このElementがサービス・リクエスター・パイプラインに指定される場合、ハンドラーは、アウトバウンド SOAP メッセージに、認証に必要な、どのセキュリティ・トークンの追加も試みません。

- 2) リクエスター・パイプラインでは、SOAP メッセージの本文の署名に使用されるアルゴリズムの URI を指定する、オプションの <algorithm> Element。**trust** 属性と **mode** 属性の値の組み合わせが、メッセージが署名されていることを示している場合は、このElementを指定する必要があります。このElementでは、SHA1 を使用する RSA アルゴリズムだけを指定できます。URI は <http://www.w3.org/2000/09/xml#sig#rsa-sha1> です。
- 3) RACF にインストールされる X.509 デジタル証明書に関連したラベルを指定する、オプションの <certificate_label> Element。このElementがサービス・リクエスター・パイプラインに指定され、<suppress> Elementが指定されない場合は、証明書が SOAP メッセージのセキュ

リティー・ヘッダーに追加されます。 <certificate_label> エlementを指定しない場合は、 CICS が RACF 鍵リングでデフォルトの証明書を使用します。

このElementはサービス・プロバイダー・パイプラインでは無視されます。

3. オプション: <authentication> Elementの代わりに <sts_authentication> Elementをコーディングします。 両方をパイプライン構成ファイルにコーディングすることはできません。このElementは、Security Token Service (STS) が認証に使用されることを指定し、送信される要求のタイプを決定します。
 - a. オプション: サービス・プロバイダー・モードの場合のみ、 **action** 属性をコーディングして、STS がセキュリティ・トークンを検証または交換するかどうかを指定します。 **action** 属性について詳しくは、104 ページの『<sts_authentication>Element』を参照してください。
 - b. 以下のElementを <sts_authentication> Element内にコーディングします。
 - 1) <auth_token_type> Element。このElementは、サービス・リクエスター・パイプラインで <sts_authentication> Elementを指定する場合は必須で、サービス・プロバイダー・パイプラインではオプションです。詳しくは、<auth_token_type> を参照してください。
 - サービス・リクエスター・パイプラインでは、<auth_token_type> Elementは、CICS が DFHWS-USERID コンテナに含まれるユーザー ID を STS に送信したときに、STS が発行するトークンのタイプを示します。CICS が STS から受け取るトークンは、アウトバウンド・メッセージのヘッダーに置かれます。
 - サービス・プロバイダー・パイプラインでは、<auth_token_type> Elementは、CICS がメッセージ・ヘッダーから取得して、交換または妥当性検査のために STS に送信する識別トークンを判別するために使用されます。CICS は最初に、メッセージ・ヘッダーで指定されたタイプの識別トークンを使用します。このElementを指定しない場合、CICS は、メッセージ・ヘッダーで見つけた最初の識別トークンを使用します。CICS は、次のものは ID トークンと見なしません。
 - wsu:Timestamp
 - xenc:ReferenceList
 - xenc:EncryptedKey
 - ds:Signature
 - 2) サービス・プロバイダー・パイプラインの場合に限り、オプションの空の <suppress/> Element。このElementが指定される場合、ハンドラーは、作業が行われるユーザー ID を決定するメッセージ内のどのセキュリティ・トークンの使用も試みません。<suppress/> Elementは、STS によって戻される ID トークンが含まれます。
4. オプション: <sts_endpoint> Elementをコーディングします。 このElementは、<sts_authentication> Elementを指定した場合にのみ使用してください。 <sts_endpoint> Elementの中に、以下のElementをコーディングします。

- <endpoint> エlement。このElementには、ネットワーク上の Security Token Service (STS) の場所を指し示す URI が含まれます。STS への接続を安全に保つには、HTTP ではなく、SSL または TLS を使用することをお勧めします。

また、WebSphere MQ エndポイントは、JMS フォーマットの URI を使用して指定することもできます。

5. オプション: インバウンド SOAP メッセージにデジタル署名する必要がある場合は、空の <expect_signed_body/> Elementをコーディングします。

<expect_signed_body/> Elementは、インバウンド・メッセージの <body> に署名が必要であることを示します。インバウンド・メッセージの本文が正しく署名されていない場合、CICS はセキュリティ障害でメッセージを拒否します。

6. オプション: デジタル署名されたインバウンド SOAP メッセージを拒否する場合は、空の <reject_signature/> Elementをコーディングします。
7. オプション: インバウンド SOAP メッセージを暗号化する必要がある場合は、空の <expect_encrypted_body/> Elementをコーディングします。

<expect_encrypted_body/> > Elementは、インバウンド・メッセージの <body> を暗号化する必要があることを示します。インバウンド・メッセージの本文が正しく暗号化されていない場合、CICS はセキュリティ障害でメッセージを拒否します。

8. 部分的に、または完全に暗号化されたインバウンド SOAP メッセージを拒否する場合は、空の <reject_encryption/> Elementをコーディングします。
9. オプション: アウトバウンド SOAP メッセージに署名する必要がある場合は、<sign_body> Elementをコーディングします。
 - a. <sign_body> Element内にある <algorithm> Elementをコーディングします。
 - b. <algorithm> Elementの後にある <certificate_label> Elementをコーディングします。

これは、完成した <sign_body> Elementの例です。

```
<sign_body>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>SIGCERT01</certificate_label>
</sign_body>
```

10. オプション: アウトバウンド SOAP メッセージを暗号化する必要がある場合は、<encrypt_body> Elementをコーディングします。
 - a. <encrypt_body> Element内にある <algorithm> Elementをコーディングします。
 - b. <algorithm> Elementの後にある <certificate_label> Elementをコーディングします。

これは、完成した <encrypt_body> Elementの例です。

```
<encrypt_body>
  <algorithm>http://www.w3.org/2001/04/xmenc#tripledes-cbc</algorithm>
  <certificate_label>ENCCERT02</certificate_label>
</encrypt_body>
```

例

次の例は、ほとんどのオプション・エレメントが存在する、完成したセキュリティー・ハンドラーを示しています。

```
<wsse_handler>
  <dfhwsse_configuration version="1">
    <authentication trust="signature" mode="basic">
      <suppress/>
      <certificate_label>AUTHCERT03</certificate_label>
    </authentication>
    <expect_signed_body/>
    <expect_encrypted_body/>
    <sign_body>
      <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
      <certificate_label>SIGCERT01</certificate_label>
    </sign_body>
    <encrypt_body>
      <algorithm>http://www.w3.org/2001/04/xmlenc#tripledes-cbc</algorithm>
      <certificate_label>ENCCERT02</certificate_label>
    </encrypt_body>
  </dfhwsse_configuration>
</wsse_handler>
```

カスタムのセキュリティー・ハンドラーの作成

独自のセキュリティー手順および処理を使用する場合は、カスタムのメッセージ・ハンドラーを作成して、セキュアな SOAP メッセージをパイプラインで処理することができます。

始める前に

ご使用のセキュリティー・ハンドラーでサポートするセキュリティーのレベルを決定し、サポートされないセキュリティーをメッセージが含んでいる場合には、必ず該当する SOAP 障害を戻すようにする必要があります。

このタスクについて

メッセージ・ハンドラーは、インバウンドおよびアウトバウンド・メッセージでのセキュリティーも処理できる必要があります。

1. EXEC CICS GET CONTAINER コマンドを使用して、DFHREQUEST または DFHRESPONSE コンテナを取り出します。
2. XML を構文解析して、WS-Security メッセージ・ヘッダーにあるセキュリティー・トークンを検出します。ヘッダーの先頭は、<wsse:Security> エレメントです。セキュリティー・トークンは、ユーザー名およびパスワード、デジタル証明書、または暗号鍵である可能性があります。メッセージは、セキュリティー・ヘッダーに多くのトークンを持つことがあるので、ハンドラーは、処理対象であるトークンを正しく識別する必要があります。
3. メッセージに実装されているセキュリティーに応じて、適切な処理を実行します。
 - a. 基本認証を実行する場合は、EXEC CICS VERIFY PASSWORD コマンドを実行します。このコマンドは、メッセージのセキュリティー・ヘッダーにあるユーザー名およびパスワードを検査します。このコマンドが成功した場合

は、DFHWS-USERID コンテナを EXEC CICS PUT CONTAINER で更新します。その他の場合は、EXEC CICS SOAPFAULT CREATE コマンドを実行します。

- b. Security Token Service によってトークンの範囲を交換するか検証することにより、拡張認証を実行する場合は、Trust クライアント・インターフェースを使用します。詳しくは、『メッセージ・ハンドラーからの Trust クライアントの起動』を参照してください。
- c. メッセージが署名済みの場合は、デジタル証明書のクレデンシャルを検証します。
- d. メッセージの部分が暗号化されている場合は、セキュリティー・ヘッダーの情報を使用して、メッセージを暗号化解除します。Web Services Security: SOAP Message Security 仕様には、これを行う方法が記述されています。

タスクの結果

CICS でセキュリティー・ハンドラー・プログラムを定義し、パイプライン構成ファイルを更新して、そのプログラムが XML に正しく配置されるようにします。サービス・リクエストのパイプライン構成ファイルでは、パイプラインの最後で実行されるように、セキュリティー・ハンドラーを構成する必要があります。サービス・プロバイダーのパイプライン構成ファイルでは、パイプラインの最初で実行されるように、セキュリティー・ハンドラーを構成する必要があります。

次のタスク

カスタムのメッセージ・ハンドラーの作成方法に関する一般情報については、<http://www.redbooks.ibm.com/abstracts/sg247126.html> からアクセスできる「*Application Development for CICS Web Services*」の Redbook 資料を参照してください。

メッセージ・ハンドラーからの Trust クライアントの起動

CICS は、独自のメッセージ・ハンドラーを作成して Security Token Service (STS) を起動できるようにするインターフェースを備えています。このインターフェースを使用すると、CICS 提供のセキュリティー・ハンドラーよりも高度な処理を実行することができます。

始める前に

このタスクについて

セキュリティー・ハンドラーの代わりに、またはそれに追加して Trust クライアントを使用できます。Trust クライアント・インターフェースを使用するには、次のようにします。

1. 正しいトークンを、インバウンドまたはアウトバウンド・メッセージのセキュリティー・メッセージ・ヘッダーから取り出します。
2. チャンネル DFHWSTC-V1 および以下の必要なコンテナを渡す、プログラム DFHPIRT にリンクします。
 - DFHWS-STURI。ネットワーク上の STS の位置を格納しています。

- DFHWS-STSACTION。STS が実行する必要がある要求のタイプの URI を格納しています。サポートされている 2 つのアクションが実行され、検証されます。
- DFHWS-IDTOKEN。STS によって検証または交換される必要があるトークンを格納しています。
- DFHWS-TOKENTYPE。STS が応答で戻す必要があるトークンのタイプを格納しています。
- DFHWS-SERVICEURI。呼び出されている Web サービス操作の URI を格納しています。

オプションで DFHWS-XMLNS コンテナを組み込んで、セキュリティー・トークンを格納する SOAP メッセージのネームスペースを提供することができます。このコンテナについては、121 ページの『ヘッダー処理プログラム・インターフェース』で詳しく説明しています。

3. DFHPIRT は、STS からの応答によって戻ります。成功した応答は、DFHWS-RESTOKEN コンテナに格納されます。

STS で要求に関する問題が発生した場合、STS は SOAP 障害を戻します。DFHPIRT は、その SOAP 障害を DFHWS-STSFault コンテナに入れます。STS が、SOAP 障害を発行した理由を提供した場合、理由は DFHWS-STReason コンテナに入れられます。

異常終了が発生した場合、処理エラーの詳細を格納している DFHERROR コンテナが戻されます。

メッセージ・ハンドラーは、これらの応答を処理し、エラーが発生した際には適切な処理を実行する必要があります。例えば、メッセージ・ハンドラーは、適切な SOAP 障害を Web サービス・リクエスターに戻す場合があります。

4. 必要に応じて、応答を処理します。プロバイダー・モードでは、パイプライン処理は、メッセージがアプリケーション・ハンドラーに到達するまでに、CICS が理解できるユーザー名およびパスワードを DFHWS-USERID コンテナに配置する必要があります。リクエスター・モードでは、メッセージ・ハンドラーは、正しいトークンをアウトバウンド・メッセージのセキュリティー・ヘッダーに追加する必要があります。

次のタスク

メッセージ・ハンドラーを作成した場合、CICS でそのプログラムを定義してインストールし、該当するパイプライン構成ファイルを更新します。サービス・リクエスターのパイプラインで、パイプライン処理の最後 (ただし CICS 提供のセキュリティー・ハンドラーの前) に発生するように、メッセージ・ハンドラーを定義します。サービス・プロバイダーのパイプラインで、パイプラインの最初 (ただし CICS 提供のセキュリティー・ハンドラーの後) に発生するように、メッセージ・ハンドラーを定義します。

関連資料

147 ページの『DFHWS-STsURI コンテナ』

DFHWS-STsURI は DATATYPE(CHAR) のコンテナです。SOAP メッセージについて ID トークンを検証または発行するために使用する、Security Token Service (STS) の絶対 URI を格納します。

147 ページの『DFHWS-STSACTION コンテナ』

DFHWS-STSACTION は DATATYPE(CHAR) のコンテナです。セキュリティ・トークンを検証または発行するために、Security Token Service (STS) がとる必要があるアクションの URI を格納します。

146 ページの『DFHWS-IDTOKEN コンテナ』

DFHWS-IDTOKEN は DATATYPE(CHAR) のコンテナです。これには、メッセージの ID トークンを発行するために Security Token Service (STS) によって検証または使用されるトークンが入ります。

148 ページの『DFHWS-TOKENTYPE コンテナ』

DFHWS-TOKENTYPE は DATATYPE(CHAR) のコンテナです。Security Token Service (STS) が SOAP メッセージについて ID トークンとして発行する、要求されたトークン・タイプの URI を格納します。

146 ページの『DFHWS-SERVICEURI コンテナ』

DFHWS-SERVICEURI は DATATYPE(CHAR) のコンテナです。Security Token Service (STS) が AppliesTo の有効範囲として使用する URI を格納します。

146 ページの『DFHWS-RESTOKEN コンテナ』

DFHWS-RESTOKEN は DATATYPE(CHAR) のコンテナです。Security Token Service (STS) からの応答を格納します。

147 ページの『DFHWS-STSFault コンテナ』

DFHWS-STSFault は DATATYPE(CHAR) のコンテナです。Security Token Service (STS) によって戻されたエラーを格納します。

147 ページの『DFHWS-STReason コンテナ』

DFHWS-STReason は DATATYPE(CHAR) のコンテナです。このエレメントが Security Token Service (STS) からの応答メッセージに存在する場合は、<wst:Reason> エレメントの内容を格納します。

126 ページの『DFHERROR コンテナ』

DFHERROR は、パイプラインのエラーに関する情報を他のメッセージ・ハンドラーに 伝達する、DATATYPE(BIT) のコンテナです。

第 14 章 Web サービス・アシスタントと WSRR の相互運用性

CICS Web サービス・アシスタントは、IBM WebSphere Service Registry and Repository (WSRR) との相互運用が可能です。WSRR を使用すると、必要な Web サービスを素早く見つけたり、提供する Web サービスのバージョン管理を実施することができます。

DFHLS2WS および DFHWS2LS には、WSRR と相互運用するためのパラメーターが含まれています。また、DFHLS2WS には、WSRR の WSDL 文書に独自のカスタマイズ・メタデータを追加するためのオプション・パラメーターも含まれています。

Web サービス・アシスタントと WSRR との通信を保護するには、セキュア・ソケット・レベル (SSL) 暗号化を使用できます。DFHLS2WS および DFHWS2LS には、SSL 暗号化を使用するためのパラメーターが含まれています。

Web サービス・アシスタントと WSRR で SSL を使用するには、『Web サービス・アシスタントと WSRR で SSL を使用方法の例』を参照してください。

Web サービス・アシスタントと WSRR で SSL を使用方法の例

Secure Sockets Layer (SSL) 暗号化を使用することにより、Web サービス・アシスタントと IBM WebSphere Service Registry and Repository (WSRR) サーバーの間で安全に相互運用することができます。SSL 暗号化を使用するには鍵ストアとトラストストアが必要です。さらに、Web サービス・アシスタントでいくつかのパラメーターを指定する必要があります。

このタスクについて

Web サービス・アシスタントと WSRR の対話のために SSL 暗号化を使用するには、以下の手順を完了します。

1. 秘密鍵および公開鍵証明書 (PKC) のための鍵ストアを作成します。
 - a. IBM 鍵管理ユーティリティ (iKeyman) などの鍵構成プログラムを使って鍵ストアを作成できます。
 - b. 作成した鍵ストアの完全修飾名を、DFHWS2LS または DFHLS2WS の **SSL-KEYSTORE** パラメーターで指定します。
 - c. オプション: 作成した鍵ストアのパスワードを、DFHWS2LS または DFHLS2WS の **SSL-KEYPWD** パラメーターで指定します。
2. すべてのトラステッド・ルート認証局 (CA) 証明書のためのトラストストアを作成します。これらの証明書は、インバウンド公開鍵証明書の信用を確立するために使用されます。
 - a. IBM 鍵管理ユーティリティ (iKeyman) などの鍵構成プログラムを使ってトラストストアを作成できます。
 - b. 作成したトラストストアの完全修飾名を、DFHWS2LS または DFHLS2WS の **SSL-TRUSTSTORE** パラメーターで指定します。

- c. オプション: 作成したトラストストアのパスワードを、DFHWS2LS または DFHLS2WS の **SSL-TRUSTPWD** パラメーターで指定します。
3. Web サービス・アシスタントが SSL 暗号化を使って WSRR と通信できることをテストします。
- a. Web サービス・アシスタントと WSRR の通信をテストするには、IBM WebSphere Application Server (WAS) に付属のサンプル・ファイルを使用できます。
 - WAS に付属のサンプル鍵ストアは DummyClientKeyFile.jks および DummyServerKeyFile.jks です。
 - WAS に付属のサンプル・トラストストアは DummyClientTrustFile.jks および DummyServerTrustFile.jks です。
 - b. 鍵ストアおよびトラストストアのサンプル・ファイル内の鍵を置き換えます。これらは WAS に付属している鍵であり、セキュリティのために置き換える必要があります。

タスクの結果

これで、Web サービス・アシスタントは SSL 暗号化を使用してネットワークで WSRR と安全に通信できます。

第 15 章 問題の診断

CICS での Web サービスの実装時に起こる可能性がある問題は、CICS が SOAP メッセージを変換している間の配置プロセス中または実行時に発生することがあります。

このタスクについて

配置エラーの診断

配置エラーは、CICS Web サービス・アシスタントのバッチ・ジョブまたは CICS XML アシスタントのバッチ・ジョブを実行したり、CICS に PIPELINE リソースまたは WEBSERVICE リソースをインストールしたりしようとするときに発生することがあります。ここでは、問題の症状、原因、および解決策を含む、最も一般的な配置エラーについて説明します。

このタスクについて

配置エラーが発生した場合、通常 PIPELINE リソースは DISABLED 状態でインストールされ、WEBSERVICE リソースは UNUSABLE 状態でインストールされます。CICS Web サービス・アシスタントのバッチ・ジョブおよび CICS XML アシスタントのバッチ・ジョブに関連する情報とエラー・メッセージは、ジョブ・ログにあります。リソースのインストールに関連するエラー・メッセージは、システム・ログにあります。

コード 0、4、8、または 12 はアシスタントによって発行され、それ以外のコードは通常、BPXBATCH、JVM、または IEBGENER によって発行されます。

BPXBATCH によって発行されるコードは 2 つの主要カテゴリーに分類されます。つまり、128 未満のコードはコマンドの失敗を示し、128 以上のコードはシグナルによって処理が終了されたことを示します。BPXBATCH とその戻りコードについて詳しくは、*z/OS UNIX システム・サービス コマンド解説書*を参照してください。

- CICS Web サービス・アシスタントのバッチ・ジョブまたは CICS XML アシスタントのバッチ・ジョブの実行時に、戻りコード 0、4、8、または 12 を受け取ります。戻りコードの意味は次のとおりです。
 - 0 - ジョブは正常に完了しました。
 - 4 - 警告。ジョブは正常に完了しましたが、1 つ以上の警告メッセージが発行されました。
 - 8 - 入力エラー。ジョブが正常に完了しませんでした。入力パラメーターの検証中に 1 つ以上のエラー・メッセージが発行されました。
 - 12 - エラー。ジョブが正常に完了しませんでした。実行中に 1 つ以上のエラー・メッセージが発行されました。

1. ジョブ・ログで警告メッセージまたはエラー・メッセージがないか確認します。メッセージの詳細な説明を調べてください。通常は、問題を修正するために実行できる処置についての説明があります。
 2. ジョブで各パラメーターに合った値を入力したことを確認します。Web サービス記述のファイル名やエレメントなどのパラメーター値は、大/小文字が区別されます。
 3. 正しいパラメーターの組み合わせを指定したことを確認します。例えば、サービス・リクエスターの Web サービス・バインディング・ファイルの生成時に、DFHWS2LS に **PGMNAME** パラメーターを含めると、エラーが発生し、ジョブは正常に完了しません。
- CICS Web サービス・アシスタントのバッチ・ジョブまたは CICS XML アシスタントのバッチ・ジョブの実行時に、戻りコード 1、136、または 139 を受け取ります。これらの戻りコードは、多くの場合使用可能なストレージが不十分なために JVM で障害が発生したことを意味します。CICS アシスタントでは、少なくとも 200 MB の JCL 領域サイズが必要です。
 1. 領域サイズを増やすか、領域サイズを 0M に設定することを検討します。
 2. 領域サイズを制限する可能性があるアクティブな IEFUSI 出口があるかを調べます。
 - CICS Web サービス・アシスタントのバッチ・ジョブ DFHLS2WS または CICS XML アシスタントのバッチ・ジョブ DFHLS2SC の実行時に、戻りコード 137 を受け取ります。この戻りコードは、ジョブがタイムアウトになったことを意味します。
 1. ジョブの EXEC ステートメントの **TIME** パラメーターを **TIME=1440** にコーディングして時間を増やすか、SYS1.PARMLIB(BPXPRMxx) メンバーの MAXCPU TIME 値を増やします。
 - WEBSERVICE リソースをインストールしようすると、DFHPI0914 エラー・メッセージを受け取ります。このメッセージには、インストール障害の原因に関する情報が含まれています。
 1. z/OS UNIX での Web サービス・バインディング・ファイルの読み取りを CICS に許可したことを確認します。
 2. Web サービス・バインディング・ファイルが破損していないことを確認します。これは、FTP を使用して、バイナリー・モードではなくテキスト・モードでファイルを z/OS UNIX に転送する場合に発生することがあります。
 3. 同じ名前の 2 つの Web サービス・バインディング・ファイルが異なるピックアップ・ディレクトリーにないことを確認します。
 4. Web サービス・リクエスター・アプリケーションのリソースをインストールする場合は、SOAP バインディングのバージョンがパイプラインでサポートされるレベルと一致することを確認します。SOAP 1.2 をサポートするサービス・リクエスター・パイプラインに SOAP 1.1 WEBSERVICE をインストールすることはできません。
 5. プロバイダー・モード WEBSERVICE リソースをリクエスター・モード・パイプラインにインストールしていないことを確認します。プロバイダー・モードの Web サービス・バインディング・ファイルでは **PROGRAM** 値が指定されるのに対して、リクエスター・モードのバインディング・ファイルではこの値は指定されません。

6. DFHWS2LS または DFHLS2WS を使用する場合は、Web サービス・バインディング・ファイルの生成時に正しいパラメーターを指定したことを確認します。 **PGMNAME** など、パラメーターの中には Web サービス・プロバイダーでしか使用できないものがあり、Web サービス・リクエスターを作成する場合には除外する必要があります。
7. DFHWS2LS または DFHLS2WS を使用する場合は、ジョブによって発行されたメッセージを調べて、**WEBSERVICE** リソースを作成する前に解決すべき問題があるかどうかを確認します。
- **PIPELINE** リソースがインストールに失敗し、DFHPI0700、DFHPI0712、または DFHPI0714 などのエラー・メッセージを受け取ります。
 1. DFHPI0700 エラー・メッセージを受け取った場合は、CICS 領域で PL/I 言語サポートを使用可能にする必要があります。これは、**PIPELINE** リソースをインストールする前に行う必要があります。詳しくは、「*CICS Transaction Server for z/OS インストール・ガイド*」を参照してください。
 2. パイプライン構成ファイルを読み取るために z/OS UNIX ディレクトリーへのアクセスを CICS に許可したことを確認します。
 3. **WSDIR** パラメーターで指定したディレクトリーが有効なことを確認します。z/OS UNIX ではディレクトリーとファイル名は大/小文字が区別されるため、特に大/小文字を確認します。
 4. CICS 領域に **ENABLED** 状態の同じ名前の **PIPELINE** リソースがないことを確認します。
- **PIPELINE** リソースが **DISABLED** 状態でインストールされます。DFHPI0702 から DFHPI0711 までの範囲のエラー・メッセージを受け取ります。
 1. パイプライン構成ファイルにエラーがないことを確認します。パイプライン構成ファイルの要素は、特定の場所にしか現れません。これらの要素を誤って指定すると、DFHPI0702 エラー・メッセージを受け取ります。このメッセージには、問題の原因となっている要素の名前が含まれています。要素記述を調べて、正しい場所でコーディングしたことを確認します。
 2. タブなどの印刷不能文字がパイプライン構成ファイルにないことを確認します。
 3. XML が有効なことを確認します。XML が無効な場合、これにより **PIPELINE** リソースをインストールしようすると構文解析エラーが発生することがあります。
 4. パイプライン構成ファイルが US EBCDIC でエンコードされていることを確認します。別の EBCDIC エンコードを使用すると、CICS がファイルを処理できません。

サービス・プロバイダーのランタイム・エラーの診断

プロバイダー・モード・パイプラインでのインバウンド・メッセージの受信または処理中に問題が発生する場合は、トランスポートまたは特定の SOAP メッセージに問題がある可能性があります。

始める前に

このタスクについて

- HTTP または WMQ トランスポート・エラーが発生したことを示す、DFHPI0401 や DFHPI0502 のようなメッセージを受け取ります。 トランスポートが HTTP の場合、クライアントは「500 Server Internal Error (500 サーバーの内部エラーが発生しました)」メッセージを受け取ります。トランスポートが WMQ の場合、メッセージは送達不能キュー (DLQ) に書き込まれます。 CICS は受け取ったメッセージのタイプを判別できないため、SOAP 障害は Web サービス・リクエスターに戻されません。
- DFHxx メッセージおよび「404 Not Found (404 見つかりません)」エラー・メッセージを受け取ります。
 1. Web サービス・アシスタントを使用していない場合は、URIMAP リソースを作成する必要があります。 Web サービス・アシスタントを使用している場合は、PIPELINE SCAN コマンドの実行時に URIMAP が自動的に作成されます。システム・ログには、このコマンドを実行した結果発生したエラーに関する情報が記載されています。
 2. WEBSERVICE リソースを使用できること、およびこのリソースが関連付けられている URIMAP が予期した URIMAP であることを確認します。 WEBSERVICE リソースを UNUSABLE 状態でインストールした場合は、335 ページの『配置エラーの診断』を参照してください。
 3. URI およびポート番号を正しく指定したことを確認します。 URIMAP リソース上の属性 PATH には大/小文字の区別があるため、特に大/小文字を確認します。
- 予期せぬエラーが報告されている場合は、CEDX を使用して Web サービス・アプリケーションをデバッグすることを検討します。
 1. システム・ログを調べて、CICS によって報告されているエラー・メッセージを確認します。 これは、発生しているエラーのタイプを示しています。 CICS がエラーを報告していない場合は、要求がネットワーク経由で CICS に到達していることを確認します。
 2. HTTP トランスポートの場合は CPIH、WMQ トランスポートの場合は CPIQ、またはこれが異なる場合はユーザーが URIMAP で指定したトランザクションに対して CEDX を実行します。

パイプライン処理中にアプリケーション・ハンドラーの前にタスク切り替えが行われると、DFHWS-TRANID コンテナが取り込まれない限り、新規のタスクが最初のタスクと同じトランザクション ID の下で実行されます。最初のタスクの CEDX セッションにロックがある場合は、これによって CEDX の実行が妨害されることがあります。この問題は、DFHWS-TRANID を使用してタスク切り替え時にトランザクション ID を変更し、パイプラインとアプリケーションの両方のタスクで別個に CEDX を使用できるようにすることによって回避できます。CEDX について詳しくは、「*CICS Supplied Transactions*」の『Using the CEDX transaction』を参照してください。

3. CEDX が活動化されないか、問題を解決できない場合は、PI、SO、AP、EI、および XS ドメインをアクティブにして補助トレースを実行することを検討してください。 これは、CICS 領域にセキュリティの問題、TCP/IP の問

題、アプリケーション・プログラムの問題、またはパイプラインの問題があることを示している可能性があります。例外トレース・ポイントまたは異常終了がないか調べてください。

- 変換エラーを受け取る場合は、343 ページの『データ変換エラーの診断』を参照してください。
- 問題が MTOM メッセージに関連していると思う場合は、341 ページの『MTOM/XOP エラーの診断』を参照してください。

例

次のタスク

サービス・リクエスターのランタイム・エラーの診断

サービス・リクエスター・アプリケーションから Web サービス要求を送信する際に問題が発生するか、Web サービス・プロバイダーから SOAP 障害メッセージを受け取る場合は、このセクションを参照してください。

このタスクについて

発生する問題は、個々の Web サービスのエラーまたはトランスポート・レベルでの問題が原因になっている可能性があります。

- アプリケーション・プログラムで `INVOKE SERVICE` コマンドを使用している場合、問題があると `RESP` コードと `RESP2` コードが戻されます。
 1. 何が問題なのかを示す、`INVOKE SERVICE` コマンドの `RESP` および `RESP2` コードの意味を調べます。
 2. CICS システム・ログを調べて、問題の原因を判別する際に役立つメッセージがあるかどうかを確認します。
- SOAP 要求メッセージを送信できず、パイプラインが `DFHERROR` コンテナを戻す場合は、パイプラインが SOAP メッセージを処理しようとしたときに問題が発生しました。
 1. `DFHERROR` コンテナの内容を調べてください。ここには、エラー・メッセージおよび発生した問題についての説明データが含まれているはずです。
 2. パイプラインに新規のメッセージ・ハンドラーまたはヘッダー処理プログラムを導入しましたか？ 導入した場合は、その新規のプログラムを除去して、Web サービスを再実行することによって、問題が解決するかどうかを確認します。メッセージ・ハンドラーが、パイプラインに存在しないコンテナを使用して処理を実行しようとしたり、読み取り専用のコンテナを更新しようとしたりと、パイプラインは処理を停止して、`DFHERROR` コンテナでエラーを戻します。ヘッダー処理プログラムが更新できるのは、パイプラインの限定されたコンテナ群のみです。詳しくは、121 ページの『ヘッダー処理プログラム・インターフェース』を参照してください。
 3. Web サービス・リクエスター・アプリケーションが、Web サービス要求を送信するために `INVOKE SERVICE` コマンドを使用していない場合、必要な制御コンテナがすべて作成されていること、およびこれらのコンテナのデータ・タイプが正しいことを確認します。特に、`DFHREQUEST` コンテナのデータ・タイプが `BIT` ではなく `CHAR` であることを確認します。

4. Web サービス・リクエスター・アプリケーションが INVOKE SERVICE コマンドを使用しており、INVREQ および RESP2 コード 14 が戻される場合は、データ変換エラーが発生したことを示しています。343 ページの『データ変換エラーの診断』を参照してください。
 5. パイプライン処理中に SOAP メッセージ内の XML がカスタムのメッセージ・ハンドラーによって無効にされていなかったことを確認します。CICS は、パイプライン内のアウトバウンド・メッセージで検証を実行しません。アプリケーションが INVOKE SERVICE コマンドを使用する場合、SOAP メッセージの本文が DFHREQUEST コンテナ内に置かれると、XML が CICS によって生成され、適切な形式になります。ただし、SOAP メッセージの内容を変更する追加のメッセージ・ハンドラーがある場合、これはパイプラインでは検証されません。
- SOAP メッセージを送信できるのに、タイムアウト・エラーまたはトランスポート・エラーが発生する場合は、通常、これは SOAP 障害として戻されます。プログラムが INVOKE SERVICE コマンドを使用している場合、CICS は、タイムアウト・エラーでは TIMEDOUT の RESP 値および RESP2 コード 2 を戻し、トランスポート・エラーでは INVREQ の RESP 値および RESP2 コード 17 を戻します。
 1. ネットワーク・エンドポイントが存在することを確認します。
 2. PIPELINE リソース上の RESPWAIT 属性が、ご使用のアプリケーションの要件を満たすよう正しく構成されていることを確認します。RESPWAIT 属性は、CICS がアプリケーションに戻るまでに Web サービス・プロバイダーからの応答を待機する期間を定義します。値が指定されていない場合、CICS は、HTTP ではデフォルトの 10 秒、WMQ ではデフォルトの 60 秒を使用します。ただし、CICS は、ディスパッチャーでもトランザクションごとにタイムアウトを持ちます。これが使用されているプロトコルのデフォルトより短い場合は、CICS は代わりにディスパッチャーのタイムアウトを使用します。
 - SOAP メッセージを送信できるのに、Web サービス・プロバイダーから予期していなかった SOAP 障害の応答が戻される場合は、SOAP 障害の詳細について DFHWS-BODY コンテナの内容を調べてください。
 1. DFHPIRT インターフェースを使用して、DFHREQUEST 内の完全な SOAP エンベロープを送信する場合は、アウトバウンド・メッセージに重複した SOAP ヘッダーが含まれていないか確認してください。これは、リクエスターのパイプラインで SOAP 1.1 または SOAP 1.2 メッセージ・ハンドラーが使用された場合に発生する可能性があります。SOAP メッセージ・ハンドラーは、サービス・リクエスター・アプリケーションによって SOAP ヘッダーが SOAP エンベロープ内にすでに指定されている場合にも、SOAP ヘッダーを追加します。このシナリオでは、次のいずれかを行うことができます。
 - パイプラインから SOAP 1.1 または SOAP 1.2 メッセージ・ハンドラーを除去する。これは、そのパイプラインを使用している他のサービス・リクエスター・アプリケーションに影響を与えます。
 - アプリケーションによって DFHREQUEST に置かれた SOAP エンベロープから SOAP ヘッダーを除去する。必要な SOAP ヘッダーが CICS により追加されます。ヘッダーで追加の処理を実行する場合は、ヘッダー処理プログラム・インターフェースを使用できます。

- 代わりにアプリケーションで WEB SEND コマンドを使用し、Web サポートから抜ける。
- 問題が MTOM メッセージの送受信に関連していると思う場合は、『MTOM/XOP エラーの診断』を参照してください。

MTOM/XOP エラーの診断

MTOM/XOP エラーは、リクエスター・モード・パイプラインとプロバイダー・モード・パイプラインの両方で、実行時に発生する可能性があります。

始める前に

MTOM/XOP をサポートするようパイプラインを構成する際に問題が発生した場合は、335 ページの『配置エラーの診断』を参照してください。

このタスクについて

- Web サービス要求メッセージを MTOM 形式で送信できるのに、Web サービス・プロバイダーから SOAP 障害メッセージを受け取る場合は、SOAP 障害の詳細について DFHWS-BODY コンテナの内容を調べてください。
 1. Web サービス・プロバイダーは MIME Multipart/Related メッセージを受信できますか? Web サービス・プロバイダーが MTOM 形式をサポートしていない場合、戻される障害は実装によって異なることがあります。Web サービス・プロバイダーが別の CICS アプリケーションである場合、SOAP 障害は、MIME メッセージが有効なコンテンツ・タイプではないことを示しています。Web サービスが MTOM/XOP をサポートするかどうかを調べるには、CEMT INQUIRE WEBSERVICE コマンドを使用します。
 2. Web サービス・プロバイダーが MIME メッセージを受信できる場合は、パイプラインがメッセージを直接モードで送信しているか、または互換モードで送信しているかを確認します。パイプラインの状況を取得するには、INQUIRE PIPELINE コマンドを使用します。MTOM メッセージを直接モードで送信すると、XML の問題が発生する可能性があります。
 3. 問題が XML に関連しているかどうかを調べるには、Web サービスの検証をオンにします。これによって、パイプライン全体で MTOM メッセージが互換モードで処理されるようになります。この処理の一環として、MTOM ハンドラーは base64binary データを最適化するためにメッセージの内容を解析します。XML にエラーがある場合は、CICS はエラーを DFHERORR コンテナ内に入れて、パイプラインで MTOM トランスポート障害を発行します。
 4. DFHERORR コンテナの内容を調べて、これが発生した問題を示しているかどうかを確認します。この情報が問題の原因を診断するのに十分ではない場合は、レベル 2 トレースの PI ドメインを実行します。
 5. トレース・ポイント PI 0C16 を調べます。ここには、検出した問題が詳細に説明されており、リクエスター・アプリケーションによって提供された XML の問題を修正する際に役立ちます。
- アウトバウンド MTOM メッセージに必要なバイナリー添付ファイルがない場合は、バイナリー・データが小さすぎて、バイナリー添付ファイルとして最適化できないと見なされたことを示している可能性があります。CICS は、パイプラインでデータを最適化する処理オーバーヘッドを許容できるほど十分な大きさのデ

ータにのみバイナリー添付ファイルを作成します。サイズが 1,500 バイトを下回るバイナリー・データは最適化されません。

- アウトバウンド MTOM メッセージを互換モードで送信できず、パイプラインが DFHERROR コンテナを戻す場合は、パイプラインが MTOM メッセージを処理しようとしたときに問題が発生しました。

1. DFHERROR コンテナの内容を調べてください。ここには、エラー・メッセージおよび発生した問題についての説明データが含まれているはずです。
2. アウトバウンド MTOM メッセージ内の XML が有効なことを確認します。CICS は、パイプライン内のアウトバウンド・メッセージで検証を実行しません。

- DFHPI1100E メッセージを受信する場合は、CICS が受信した MTOM メッセージの MIME ヘッダーに問題が発生しました。CICS メッセージには、発生した MIME エラーの汎用クラスが含まれています。発生した正確な問題を見つけるには、次のようにします。

1. CICS 領域で補助トレースがアクティブになっている場合は、例外トレース・エントリーがないかを確認します。
2. トレース・ポイント PI 1305 を調べます。ここには、MIME ヘッダー・エラーの性質、ヘッダー内のエラーの場所、およびエラーの前後にある 80 バイトまでのテキストが記述されているため、エラーが発生した場所のコンテキストを理解することができます。

例えば、次のトレースの抜粋は、MIME コンテンツ・タイプの開始パラメーターが、引用符で囲まれていないが、引用符付きストリングの外に無効な文字が含まれているため無効であることを示しています。

PI 1305 PIMM *EXC* - MIME_PARSE_ERROR -

```
TASK-01151 KE_NUM-0214 TCB-QR /009C7B68 RET-9C42790A TIME-10:33:41.3667303015 INTERVAL-00.0000053281 =000599=
1-0000 C5A79785 83A38584 40978199 819485A3 859940A5 8193A485 40A39692 85954096 *Expected parameter value token o*
0020 994098A4 96A38584 40A2A399 899587 *r quoted string *
2-0000 D4C9D4C5 40A2A895 A381A740 85999996 994081A3 404EF0F0 F0F0F1F1 F2408995 *MIME syntax error at +0000112 in*
0020 40C39695 A38595A3 60A3A897 85408885 81848599 * Content-type header *
3-0000 5F626F75 6E646172 793B2074 7970653D 22617070 6C696361 74696F6E 2F786F70 *_boundary; type="application/xop*
0020 2B786D6C 223B2073 74617274 2D696E66 6F3D2261 70706C69 63617469 6F6E2F73 **xml"; start-info="application/s*
0040 6F61702B 786D6C22 3B207374 6172743D 68757273 6C65792E 69626D2E 636F6D3E *oap+xml"; start=
4-0000 3C736F61 70736C75 6E674074 6573742E 68757273 6C65792E 69626D2E 636F6D3E *soap:lung@test.hursley.ibm.com*>
0020 3B206368 61727365 743D7574 662D38 *; charset=utf-8 *
```

- パイプライン処理はインバウンド MTOM メッセージの解析に失敗し、Web サービス・リクエスターは SOAP 障害メッセージを受け取ります。これは、MTOM メッセージ内の XOP 文書に問題があったことを示しています。直接モードでは、SOAP 障害はアプリケーション・ハンドラーによって生成されます。パイプラインが互換モードで実行されている場合、メッセージは、SOAP メッセージの作成時に MTOM ハンドラーによって解析されます。この場合、CICS は接頭部が DFHPI のエラー・メッセージと SOAP 障害を発行します。

1. 接頭部が DFHPI のエラー・メッセージは、XOP 文書の何に問題があるかを示しています。例えば、MIME ヘッダーが無効か、バイナリー添付ファイルがない可能性があります。
2. 問題の正確な原因を見つけるには、例外トレース・ポイントがないか調べます。特に、先頭文字が PI 13xx のトレース・ポイントを調べます。ここには、発生した例外の詳細が記述されています。

また、PI レベル 2 トレースを実行して、エラーをもたらした一連のイベントを設定することもできますが、これによってパフォーマンスが大きな影響を受ける可能性があるため、実動領域ではお勧めできません。

データ変換エラーの診断

データ変換エラーは、SOAP メッセージを CICS COMMAREA またはコンテナに変換したり、COMMAREA またはコンテナから SOAP メッセージに変換すると、実行時に発生することがあります。

始める前に

SOAP 障害メッセージおよび障害が発生したことを示す CICS メッセージが生成されるなどの症状があります。

このタスクについて

データ変換の問題が発生した場合は、以下のステップを実行する必要があります。

1. WEBSERVICE リソースが最新の状態になっていることを確認します。Web サービスの Web サービス・バインディング・ファイルを再生成して、CICS に再配置します。
2. リモート Web サービスが、CICS によって使用または生成されたものと同じバージョンの Web サービス文書 (WSDL) を使用して生成されたことを確認します。
3. WEBSERVICE リソースが現行の Web サービス・バインディング・ファイルを使用していることが確かな場合は、次のようにします。
 - a. コマンド SET WEBSERVICE(*name*) VALIDATION を使用して、WEBSERVICE リソースのランタイム検証を使用可能にします。ここで、*name* は WEBSERVICE リソース名です。
 - b. メッセージ・ログに CICS メッセージ DFHPI1001 または DFHPI1002 がないかどうかを確認します。DFHPI1001 には、データ変換の問題に関する正確な性質が記載されており、変換エラーの原因を特定するのに役立ちます。DFHPI1002 は、問題が見つからなかったことを示しています。
 - c. Web サービスの検証が必要なくなったら、コマンド SET WEBSERVICE(*name*) NOVALIDATION を使用して検証をオフにします。
4. それでも変換エラーの理由を特定できない場合は、障害が発生している Web サービスの CICS トレースを使用します。次の PI ドメインの例外トレース・エントリーを探します。

```
PI 0F39 - PICC *EXC* - CONVERSION_ERROR  
PI 0F08 - PIII *EXC* - CONVERSION_ERROR
```

PICC 変換エラーは、CICS が受信した SOAP メッセージを COMMAREA またはコンテナに変換する際に問題が発生したことを示しています。PIII 変換エラーは、アプリケーション・プログラムが提供した COMMAREA またはコンテナから SOAP メッセージを生成する際に問題が発生したことを示しています。どちらの場合も、トレース・ポイントは、変換エラーに関連したフィールドの名前を示しており、また問題の原因となる値も示していることがあります。これらのいずれかのトレース・ポイントがある場合は、この後に変換エラーがあります。これらの変換エラーの考えられる解釈については、345 ページの『トレース・ポイントにおける変換エラー』を参照してください。

次のタスク

データ変換エラーが起こる理由

CICS による SOAP メッセージの検証は、メッセージが適切な形式の XML を含むことを確認し、メッセージを変換するために必要な範囲に限定されます。これは、WSDL を使用して SOAP メッセージを正常に検証できるが、ランタイム環境では失敗することを意味します。また、その逆も同様です。

WEBSERVICE リソースは、マッピング指示をカプセル化して、CICS が実行時にデータ変換を実行できるようにします。WEBSERVICE リソースに記述されるように、入力と期待されるデータが一致しないと、変換エラーが起こります。

この不一致は、以下のどの理由でも起こります。

- CICS が受け取る SOAP メッセージが、WEBSERVICE リソースに関連付けられた Web サービス記述 (WSDL) について検査される際に、このメッセージが適切な形式でなく、有効でない。
- CICS が受け取る SOAP メッセージが適切な形式であり有効だが、WEBSERVICE リソースの範囲外の値が含まれている。
- COMMAREA またはコンテナの内容と、WEBSERVICE リソースおよび Web サービスが生成された言語構造に整合性がない。

例えば、WSDL 文書で、10 から 20 の間の値しか持てない `unsignedInt` のような範囲制限をフィールドに指定する場合があります。SOAP メッセージに値 25 が含まれている場合に、SOAP メッセージを検証すると、このメッセージは無効として拒否されます。値 25 は整数については有効な値として受け入れられるため、アプリケーションに渡されます。

2 番目の例は、WSDL 文書が最大長を指定しないでストリングを指定する場合です。DFHWS2LS は、Web サービス・バインディング・ファイルを生成する際に、デフォルトで最大長を 255 文字と仮定します。SOAP メッセージに 300 文字が含まれている場合、最大長が設定されていないので WSDL に対する検査ではメッセージは有効とされますが、CICS によって割り振られる 255 文字のバッファに値が合わないので、メッセージを変換しようとするエラーが報告されます。

コード・ページの問題

CICS は、**LOCALCCSID** システム初期設定パラメーターの値を使用して、アプリケーション・プログラム・データをエンコードします。しかし、Web サービス・バインディング・ファイルは US EBCDIC (Cp037) でエンコードされます。このため、アプリケーション・プログラムで使用するコード・ページが、US EBCDIC コード・ページと異なる文字をエンコードすると、データ変換の問題が発生することがあります。この問題を避けるために、Web サービス・アシスタントのバッチ・ジョブで **CCSID** パラメーターを使用して、アプリケーション・プログラムと Web サービス・バインディング・ファイルの間でデータをエンコードするのに異なるコード・ページを指定できます。このパラメーターの値は、特定の WEBSERVICE リソースについて、**LOCALCCSID** システム初期設定パラメーターを指定変更します。**CCSID** の *value* は EBCDIC CCSID になります。

トレース・ポイントにおける変換エラー

障害が起こった Web サービスについてトレースを実行し、PI ドメイン例外トレース・ポイント PI 0F39 または PI 0F08 を検索する際に、CICS によって変換エラーが出されます。この変換エラーについて可能な解釈を示します。変換エラーの原因の診断に役立ちます。該当する場合は、次の手順についても示します。

次の変換エラーは COMMAREA を参照しますが、同様にコンテナに適用できません。

INPUT_TOO_LONG

この変換エラーは次の場合に起こります。

- 数値として宣言される SOAP エlement に 31 を超える桁が含まれる。
- COMMAREA 内の数値フィールドに、長さが 31 桁を超える値が含まれる。

OUTPUT_OVERFLOW

この変換エラーは次の場合に起こります。

- SOAP エlement に、長すぎて COMMAREA の関連フィールドに合わない値が含まれる。
- SOAP エlement に、COMMAREA 内の関連フィールドに許可された範囲外の数値が含まれる。

Web サービス記述 (WSDL) を変更して、このフィールドに 『maxLength』 ファセットを明示的に指定してみてください。WSDL に 『maxLength』 が指定される場合は、CICS によってこれだけのスペースがこのフィールドについて COMMAREA 内で確実に確保されます。『maxLength』 ファセットが指定されない場合は、CICS はデフォルトの 255 文字を使用します。これは、このフィールドについて不適切な値である可能性があります。

また、文字ベースのフィールドに 『whitespace』 ファセットを追加して、これを 『collapse』 に設定できます。これで、空白文字がこのフィールドから除去されます。デフォルトでは、空白文字は保持されます。

NEGATIVE_UNSIGNED

この変換エラーは次の場合に起こります。

- 符号なしとして宣言される SOAP エlement に負の数値が検出された。
- 符号なしとして宣言される COMMAREA フィールドに負の数値が検出された。

NO_FRACTION_DIGITS

この変換エラーは、小数点はあるが、その後に有効な小数桁が続かない数値が、SOAP エlement にある場合に起こります。

FRACTION_TOO_LONG

この変換エラーは、ゼロ以外の小数部の桁数が WSDL が許可する桁数より多い数値が、SOAP エlement にある場合に起こります。

INVALID_CHARACTER

この変換エラーは次の場合に起こります。

- ブール値として宣言される SOAP エlement に、0、1、true、または false 以外の値が含まれる。

- hexBinary として宣言される SOAP エlementに、0-9、a-f、A-F の範囲以外の値が含まれる。
- 数値として宣言される SOAP エlementに非数字が含まれる。
- SOAP メッセージが適切な形式でない。

ODD_HEX_DIGITS

この変換エラーは、hexBinary として宣言される SOAP エlementに、奇数の 16 進文字がある場合に起こります。

INVALID_PACKED_DEC

この変換エラーは、COMMAREA 内のパック 10 進数フィールドに、XML に変換できない無効な値が含まれる場合に起こります。

INVALID_ZONED_DEC

この変換エラーは、COMMAREA 内のゾーン 10 進数フィールドに、XML に変換できない無効な値が含まれる場合に起こります。

INCOMPLETE_DBCS

この変換エラーは、COMMAREA 内の DBCS シーケンスにシフトイン (SI) 文字がない場合に起こります。

変換エラーについての SOAP 障害メッセージ

実行時に変換エラーが起こり、CICS が Web サービス・プロバイダーのように動作する場合は、サービス・リクエスターに SOAP 障害メッセージが発行されます。この SOAP 障害メッセージには、CICS が発行したメッセージが含まれます。

サービス・リクエスターは、以下の SOAP 障害メッセージのいずれかを受け取ることがあります。

- Cannot convert SOAP message (SOAP メッセージを変換できません)

この障害メッセージは、SOAP メッセージが適切な形式でなく有効でないか、値が範囲外であることを、暗黙に示しています。

- Outbound data cannot be converted (アウトバウンド・データを変換できません)

この障害メッセージは、COMMAREA またはコンテナの内容に一貫性がないことを暗黙に示しています。

- Operation not part of web service (Web サービスの操作ではありません)

この障害メッセージは、CICS が無効な SOAP メッセージを受け取ったときに出力される、特殊な形です。

CICS が Web サービス・リクエスターである場合は、INVOKE SERVICE コマンドが、INVREQ の RESP コードと 14 の RESP2 値を返します。

第 16 章 CICS カタログ・マネージャーの実例アプリケーション

CICS カタログ実例アプリケーションとは、CICS アプリケーションを外部のクライアントおよびサーバーに接続するときの最良事例を示すために設計された実用的な COBOL アプリケーションのことです。

実例アプリケーションは、簡単な販売カタログと注文処理のアプリケーションで構成されており、ここでは、エンド・ユーザーが以下の機能を実行できます。

- カタログ内の品目をリストする。
- カタログ内の個々の品目について問い合わせる。
- カタログを基に品目を注文する。

カタログは VSAM ファイルとして実装されています。

ベース・アプリケーションは 3270 ユーザー・インターフェースを備えています。が、明確に定義されたコンポーネント間インターフェースを備えたモジュラー構造により、コンポーネントを追加できます。特に、このアプリケーションは Web サービス・サポートに付属しており、既存のアプリケーションを Web サービス環境に拡張する方法を示す目的で設計されています。

ベース・アプリケーション

ベース・アプリケーションと、その 3270 ユーザー・インターフェースによって提供される機能を使用すると、保管カタログの内容をリスト表示し、このリストから品目を選択して、注文数量を入力できます。アプリケーションは、モジュラー設計になっています。これにより、アプリケーションを拡張して Web サービスなどの新技術をサポートすることが簡単になります。

348 ページの図 30 は、ベース・アプリケーションの構造を示しています。

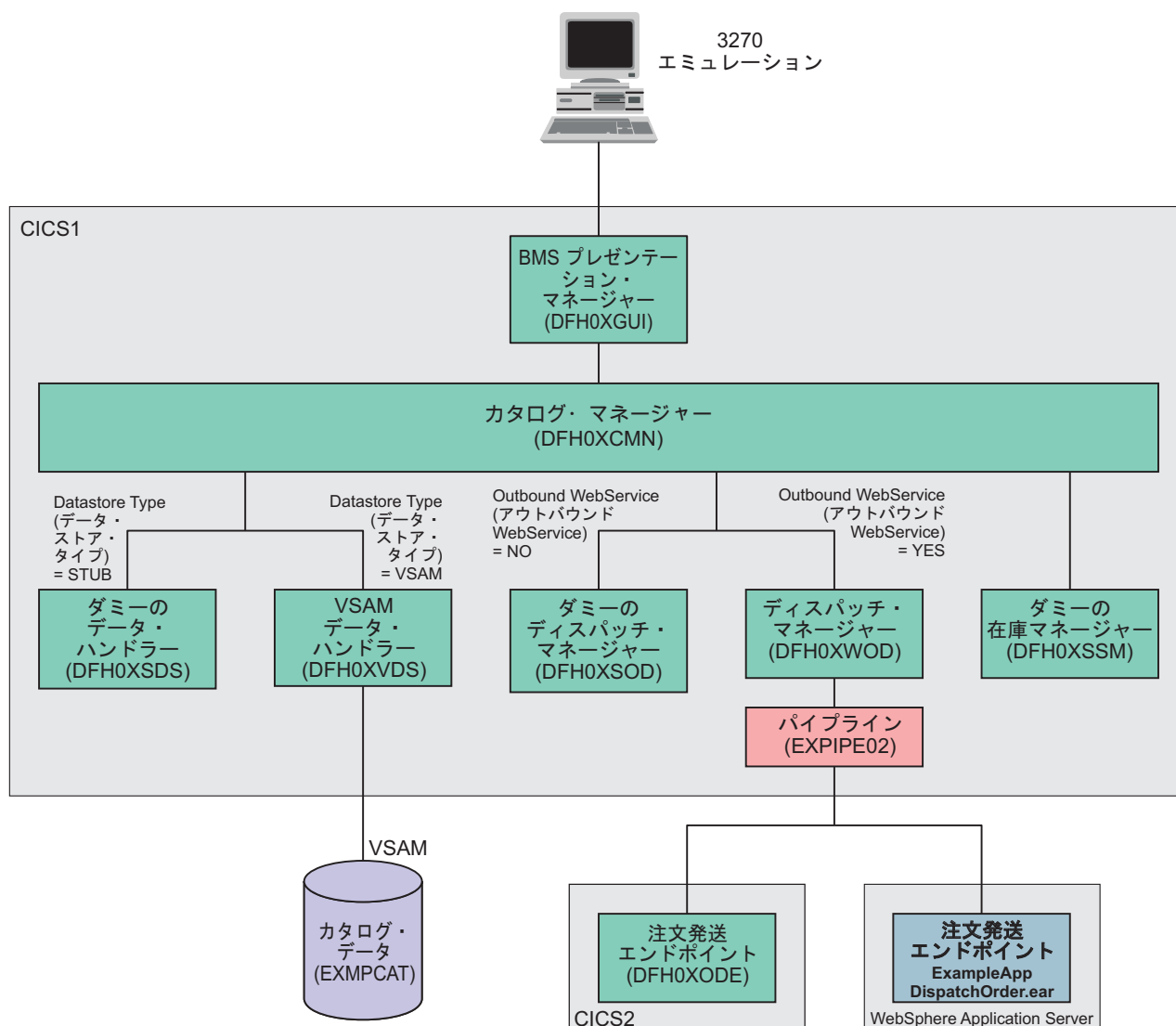


図 30. ベース・アプリケーションの構造

ベース・アプリケーションのコンポーネントは次のとおりです。

1. BMS プレゼンテーション・マネージャー (DFH0XGUI)。 3270 端末またはエミュレータをサポートし、メインのカタログ・マネージャー・プログラムと対話します。
2. カatalog・マネージャー・プログラム (DFH0XCMN)。実例アプリケーションのコアとなるもので、いくつかのバックエンド・コンポーネントと対話します。
3. バックエンド・コンポーネントは次のとおりです。
 - ・ データ・ハンドラー・プログラム。カタログ・マネージャー・プログラムとデータ・ストア間のインターフェースを提供します。ベース・アプリケーションは、このプログラムを 2 種類提供します。これらのプログラムは、VSAM データ・セットにデータを保管する VSAM データ・ハンドラー・プログラム (DFH0XVDS) と、データを保管せず、その呼び出し元に有効な応答を戻すダミーのデータ・ハンドラー (DFH0XSDS) です。構成オプションでは、2 つのプログラムのいずれかを選択できます。

- ディスパッチ・マネージャー・プログラム。注文品を顧客へ発送するためのインターフェースを提供します。この場合も、構成オプションでは、このプログラムの 2 種類のいずれかを選択することができます。DFHX0WOD は、リモートの注文発送エンドポイントを呼び出す Web サービス・リクエスターで、DFHX0SOD は、その呼び出し元に有効な応答を戻すダミー・プログラムです。

同等の注文発送エンドポイントは 2 つあります。DFHX0XODE は CICS サービス・プロバイダー・プログラムで、ExampleAppDispatchOrder.ear は、WebSphere Application Server などの類似した環境に配置できるエンタープライズ・アーカイブです。

- ダミーの在庫マネージャー・プログラム (DFHX0SSM)。呼び出し元に有効な応答を戻すが、その他の動作は行いません。

BMS プレゼンテーション・マネージャー

プレゼンテーション・マネージャーは、3270 BMS パネルを介したエンド・ユーザーとの対話をすべて担っています。このプログラムでは、ビジネス上の判断は行われません。

BMS プレゼンテーション・マネージャーは、次の 2 つの方法で使用できます。

- ベース・アプリケーションの一部として。
- SOAP メッセージを使用してベース・アプリケーションと通信する CICS Web サービス・クライアントとして。

データ・ハンドラー

データ・ハンドラーは、カタログ・マネージャーとデータ・ストア間のインターフェースを提供します。

実例アプリケーションには、以下の 2 種類のデータ・ハンドラーがあります。

- 最初のタイプでは、VSAM ファイルをデータ・ストアとして使用します。
- 2 番目のタイプは、問い合わせに対して常に同じデータを返し、更新要求の結果は保管しないダミー・プログラムです。

ディスパッチ・マネージャー

発送マネージャーは、注文が確認された場合に注文品を顧客へ発送する処理を担当します。

実例アプリケーションには、以下の 2 種類の発送マネージャー・プログラムがあります。

- 最初のタイプは、呼び出し元に正しい応答を戻すが、その他の動作はしないダミー・プログラムです。
- 2 番目のタイプは、構成ファイルに定義されたエンドポイント・アドレスに要求を行う Web サービス・リクエスター・プログラムです。

注文発送エンドポイント

注文発送プログラムとは、品目の顧客への発送を担当する Web サービス・プロバイダー・プログラムのことです。

この実例アプリケーションでは、注文発送プログラムは、呼び出し元に正しい応答を返すが、その他の動作はしないダミー・プログラムです。このプログラムにより、実例 Web サービスのすべての構成が動作可能になります。

在庫マネージャー

在庫マネージャーは、在庫補充の管理を担当します。

この実例プログラムでは、在庫マネージャーは、呼び出し元に正しい応答を返すが、その他の動作はしないダミー・プログラムです。

アプリケーションの構成

実例アプリケーションには、ベース・アプリケーションを構成できるプログラムが組み込まれています。

BMS インターフェースによる実例アプリケーションの実行

ベース・アプリケーションは、その BMS インターフェースを使用して起動できます。

1. CICS 端末からトランザクション EGUI を入力します。 実例アプリケーションにより、次のメニューが表示されます。

```
CICS EXAMPLE CATALOG APPLICATION - Main Menu
```

```
Select an action, then press ENTER
```

```
Action . . . . 1. List Items  
                2. Order Item Number ____  
                3. Exit
```

```
F3=EXIT      F12=CANCEL
```

メニューのオプションを選択すると、カタログ内の品目のリスト表示、品目の注文、アプリケーションの終了のいずれかを実行できます。

2. 「1」と入力し、ENTER キーを押して、「LIST ITEMS」オプションを選択します。 アプリケーションにより、カタログ内の品目リストが表示されます。

CICS EXAMPLE CATALOG APPLICATION - Inquire Catalog

Select a single item to order with /, then press ENTER

Item	Description	Cost	Order
0010	Ball Pens Black 24pk	2.90	/
0020	Ball Pens Blue 24pk	2.90	-
0030	Ball Pens Red 24pk	2.90	-
0040	Ball Pens Green 24pk	2.90	-
0050	Pencil with eraser 12pk	1.78	-
0060	Highlighters Assorted 5pk	3.89	-
0070	Laser Paper 28-lb 108 Bright 500/ream	7.44	-
0080	Laser Paper 28-lb 108 Bright 2500/case	33.54	-
0090	Blue Laser Paper 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. 「**ORDER**」列に「/」と入力し、ENTER キーを押して、品目を注文します。アプリケーションにより、注文した品目の詳細が表示されます。

CICS EXAMPLE CATALOG APPLICATION - Details of your order

Enter order details, then press ENTER

Item	Description	Cost	Stock	On Order
0010	Ball Pens Black 24pk	2.90	0011	000

Order Quantity: 5
User Name: CHRISB
Charge Dept: CICSDEV1

F3=EXIT F12=CANCEL

4. 注文を受けられるだけの十分な在庫がある場合は、以下の情報を入力します。
 - a. 「ORDER QUANTITY (注文数量)」フィールドに値を入力します。注文する品目の数を指定します。
 - b. 「USERID (ユーザー ID)」フィールドに値を入力します。1 から 8 文字のストリングを入力します。ベース・アプリケーションは、ここに入力された値を検査しません。
 - c. 「CHARGE DEPT (課金部門)」フィールドに値を入力します。1 から 8 文字のストリングを入力します。ベース・アプリケーションは、ここに入力された値を検査しません。
5. ENTER キーを押して注文をサブミットし、メインメニューに戻ります。

6. 「EXIT」オプションを選択して、アプリケーションを終了します。

ベース・アプリケーションのインストールおよびセットアップ

ベース・アプリケーションを実行するには、その前に 2 つの VSAM データ・セットを定義して取り込み、2 つのトランザクション定義をインストールする必要があります。

VSAM データ・セットの作成および定義

実例アプリケーションでは、定義とデータ取り込みの対象となる 2 つの KSDS VSAM データ・セットが使用されます。一方のデータ・セットには、実例アプリケーションの構成情報が格納されています。もう一方には、販売カタログが格納されています。

1. JCL を検索して、VSAM データ・セットを作成します。CICS のインストール時に、JCL は、*hlq*.SDFHINST ライブラリーに置かれます。
 - メンバー DFH\$ECNF には、構成データ・セットを生成するための JCL が記述されています。
 - メンバー DFH\$ECAT には、カタログ・データ・セットを生成するための JCL が記述されています。
2. JCL とアクセス方式サービス・プログラム・コマンドを変更します。
 - a. 有効な JOB カードを入力します。
 - b. アクセス方式サービス・プログラム・コマンドのデータ・セット名に、適切な高位修飾子を入力します。JCL は、高位修飾子の HLQ の部分に、入力に応じた内容を使用します。

以下のコマンドでは、構成ファイルが定義されます。

```
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) -
)
```

ここで、*hlq* は、選択した高位修飾子を表します。

以下のコマンドでは、カタログ・ファイルが定義されます。

```
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) -
)
```


ここで

- *hlq* は、選択した高位修飾子を示します。
- *catname* は、選択した名前を示します。提供された実例アプリケーションで使用されている名前は、EXMPCAT です。

3. 両方のジョブを実行して、データ・セットを作成し、データを取り込みます。

4. CEDA トランザクションを使用して、カタログ・ファイルの FILE 定義を作成します。

a. CEDA DEF FILE(EXMPCAT) G(EXAMPLE) と入力します。または、CICS 提供のグループ DFH\$EXBS から FILE 定義をコピーすることもできます。

b. 以下の追加属性を入力します。

```
DSNAME(hlq.EXMPLAPP.EXMPCAT)
ADD(YES)
BROWSE(YES)
DELETE(YES)
READ(YES)
UPDATE(YES)
```

c. その他の属性には、すべてデフォルト値を使用します。

5. CEDA トランザクションを使用して、構成ファイルの FILE 定義を作成します。

a. CEDA DEF FILE(EXMPCONF) G(EXAMPLE) と入力します。または、CICS 提供のグループ DFH\$EXBS から FILE 定義をコピーすることもできます。

b. 以下の追加属性を入力します。

```
DSNAME(hlq.EXMPLAPP.EXMPCONF)
ADD(YES)
BROWSE(YES)
DELETE(YES)
READ(YES)
UPDATE(YES)
```

c. その他の属性には、すべてデフォルト値を使用します。

3270 インターフェースの定義

実例アプリケーションには、アプリケーションを実行してカスタマイズするための 3270 ユーザー・インターフェースが用意されています。このユーザー・インターフェースは、EGUI と ECFG の 2 つのトランザクションで構成されます。3 番目のトランザクションである ECLI は、CICS Web サービス・クライアントで使用されます。

1. 以下のトランザクション用の TRANSACTION 定義を作成します。実例アプリケーションが正常に動作するかどうかは、トランザクションの名前には依存しないため、必要に応じて別の名前を使用できます。

a. トランザクション EGUI および ECFG 用の定義を CICS 付属のグループ DFH\$EXBS からコピーします。

- b. オプション: トランザクション ECLI 用の定義を CICS 付属のグループ DFH\$EXWS からコピーします。

または、CEDA トランザクションを使用して、トランザクションの TRANSACTION 定義を作成できます。

```
CEDA DEF TRANS(EGUI) G(EXAMPLE) PROG(DFH0XGUI)
CEDA DEF TRANS(ECFG) G(EXAMPLE) PROG(DFH0XCFG)
CEDA DEF TRANS(ECLI) G(EXAMPLE) PROG(DFH0XCUI)
```

その他の属性には、すべてデフォルト値を使用します。

2. オプション: プログラムの自動インストールを使用したくない場合は、ベース・アプリケーション・プログラム用の PROGRAM 定義および BMS マップ用の MAPSET 定義を CICS 付属のグループ DFH\$EXBS からコピーします。
 - a. メンバー DFH0XS1、DFH0XS2、および DFH0XS3 で BMS マップの MAPSET リソース定義をコピーします。各メンバーに含まれる内容について詳しくは、381 ページの『ベース・アプリケーションのコンポーネント』を参照してください。
 - b. 以下の COBOL プログラム用の PROGRAM リソース定義をコピーします。

表 14. ベース・アプリケーションの COBOL ソースが含まれる SDFHSAMP メンバー

メンバー名	説明
DFH0XCFG	VSAM 構成ファイルを読み取って更新するために、トランザクション ECFG によって呼び出されるプログラム。
DFH0XCMN	カタログ・アプリケーションのコントローラー・プログラム。すべての要求がこのプログラムをパススルーします。
DFH0XGUI	端末ユーザーへの BMS マップの送信および端末ユーザーからのマップの受信を管理するために、トランザクション EGUI によって呼び出されるプログラム。これはプログラム DFH0XCMN にリンクします。
DFH0XODE	注文発送 Web サービスの 2 つのバージョンのエンドポイントのいずれか。これは、CICS で稼働するバージョンです。このエンドポイントは、戻りの COMMAREA でテキスト「Order in dispatch」を設定します。
DFH0XSDS	VSAM カatalog・ファイルがセットアップされていない場合に、アプリケーションが操作できるスタブ化版またはダミー版のデータ・ストア・プログラム。このプログラムは、VSAM ファイルに保管されたデータではなく、プログラムで定義されたデータを使用します。
DFH0XSOD	スタブ化版の注文発送プログラム。このプログラムは、COMMAREA 内の戻りコードを 0 に設定して、呼び出し元に戻します。これは、アウトバウンド Web サービスが必要ないときに使用されます。
DFH0XSSM	スタブ化版の在庫マネージャー (補充) プログラム。このプログラムは、COMMAREA 内の戻りコードを 0 に設定して、呼び出し元に戻します。
DFH0XVDS	VSAM 版のデータ・ストア・プログラム。このプログラムは VSAM ファイルにアクセスして、カタログの読み取りと更新を実行します。

表 14. ベース・アプリケーションの COBOL ソースが含まれる SDFHSAMP メンバー (続き)

メンバー名	説明
DFH0XWOD	Web サービス版の注文発送プログラム。このプログラムは EXEC CICS INVOKE WEBSERVICE を発行して、注文発送プログラムへのアウトバウンド Web サービス呼び出しを行います。

その他の属性には、すべてデフォルト値を使用します。

- c. オプション: DFH0XCUI 用のプログラム定義を CICS 付属のグループ DFH\$EXWS からコピーします。その他の属性には、すべてデフォルト値を使用します。Web サービス・クライアントを開始するトランザクション ECLI を使用する場合、このプログラムは必須です。

```
DIS G(DFH$EXWS)
ENTER COMMANDS
NAME      TYPE      GROUP
DFH0XCUI  PROGRAM   DFH$EXWS
ECLI      TRANSACTION DFH$EXWS
EXMPPORT  TCPIPService DFH$EXWS
EXPIPE01  PIPELINE   DFH$EXWS
EXPIPE02  PIPELINE   DFH$EXWS
```

インストールの完了

インストールを完了するには、リソース定義が格納されている RDO グループをインストールします。

CICS 端末で、CEDA I G(EXAMPLE) というコマンドを入力します。

タスクの結果

これで、アプリケーションを使用する準備ができました。

実例アプリケーションの構成

ベース・アプリケーションには、実例アプリケーションを構成するために使用できるトランザクション (ECFG) が組み込まれています。

始める前に

構成トランザクションでは、大/小文字混合情報を使用します。大/小文字混合情報を正しく処理できる端末を使用する必要があります。

このタスクについて

トランザクションにより、実例アプリケーションのいくつかの性質を指定できます。この内容は次のとおりです。

- Web サービスの使用など、アプリケーションの全体的な構成
- アプリケーションの Web サービス・コンポーネントが使用するネットワーク・アドレス
- データ・ストアに使用されるファイルなどのリソースの名前
- アプリケーションの各コンポーネントに使用されるプログラムの名前

構成トランザクションでは、実例アプリケーションの CICS 提供コンポーネントを、アプリケーションを再始動することなく、独自のコンポーネントに置き換えることができます。

1. トランザクション ECFG を入力して、構成アプリケーションを開始します。
CICS では、次の画面が表示されます。

CONFIGURE CICS EXAMPLE CATALOG APPLICATION

```

Datastore Type ==> VSAM          STUB|VSAM
Outbound WebService? ==> NO      YES|NO
Catalog Manager ==> DFH0XCMN
Data Store Stub ==> DFH0XSDS
Data Store VSAM ==> DFH0XVDS
Order Dispatch Stub ==> DFH0XSOD
Order Dispatch WebService ==> DFH0XWOD
Stock Manager ==> DFH0XSSM
VSAM File Name ==> EXMPCAT
Server Address and Port ==> myserver:99999
Outbound WebService URI ==> http://myserver:80/exampleApp/dispatchOrder
                        ==>
                        ==>
                        ==>
                        ==>

```

PF
3 END
12 CNCL

2. フィールドに値を入力します。

Datastore Type (データ・ストア・タイプ)

Data Store Stub プログラムを使用する場合は、STUB を指定します。

VSAM データ・ストア・プログラムを使用する場合は、VSAM を指定します。

Outbound WebService (アウトバウンド WebService)

Order Dispatch (注文発送) 機能にリモート Web サービスを使用する場合、つまり、Catalog Manager (カタログ・マネージャー) プログラムを Order Dispatch (注文発送) Web サービス・プログラムにリンクする場合は、YES を指定します。

Order Dispatch (注文発送) 機能にスタブ・プログラムを使用する場合、つまり、Catalog Manager (カタログ・マネージャー) プログラムを Order Dispatch Stub (注文発送スタブ) プログラムにリンクする場合は、NO を指定します。

Catalog Manager (カタログ・マネージャー)

Catalog Manager (カタログ・マネージャー) プログラムの名前を指定します。実例アプリケーションに付属するプログラムは DFH0XCMN です。

Data Store Stub (データ・ストア・スタブ)

「Datastore Type (データ・ストア・タイプ)」フィールドに STUB を指定した場合は、Data Store Stub (データ・ストア・スタブ) プログラムの名前を指定します。実例アプリケーションに付属するプログラムは DFH0XSDS です。

Data Store VSAM (データ・ストア VSAM)

「**Datastore Type (データ・ストア・タイプ)**」フィールドに VSAM を指定した場合は、VSAM データ・ストア・プログラムの名前を指定します。実例アプリケーションに付属するプログラムは DFH0XVDS です。

Order Dispatch Stub (注文発送スタブ)

「**Outbound WebService (アウトバウンド WebService)**」フィールドに NO を指定した場合は、Order Dispatch Stub (注文発送スタブ) プログラムの名前を指定します。実例アプリケーションに付属するプログラムは DFH0XSOD です。

Order Dispatch WebService (注文発送 WebService)

「**Outbound WebService (アウトバウンド WebService)**」フィールドに YES を指定した場合は、サービス・リクエスターとして機能するプログラムの名前を指定します。実例アプリケーションに付属するプログラムは DFH0XWOD です。

Stock Manager (在庫マネージャー)

Stock Manager (在庫マネージャー) プログラムの名前を指定します。実例アプリケーションに付属するプログラムは DFH0XSSM です。

VSAM File Name (VSAM ファイル名)

「**Datastore Type (データ・ストア・タイプ)**」フィールドに VSAM を指定した場合は、CICS FILE 定義の名前を指定します。提供された実例アプリケーションで使用されている名前は、EXMPCAT です。

Server Address and Port (サーバー・アドレスおよびポート)

CICS Web サービス・クライアントを使用する場合は、実例アプリケーションが Web サービスとして配置されているシステムの IP アドレスとポートを指定します。

Outbound WebService URI (アウトバウンド WebService URI)

「**Outbound WebService (アウトバウンド WebService)**」フィールドに YES を指定した場合は、注文発送機能を実装する Web サービスのロケーションを指定します。提供された CICS エンドポイントを使用している場合は、これを `http://myserver:myport/exampleApp/dispatchOrder` に指定します。ここで、*myserver* および *myport* は、それぞれ使用している CICS サーバーのアドレスとポートです。

実例アプリケーションに対する Web サービス・サポート

Web サービス・サポートは、実例アプリケーションを拡張して、Web クライアント・フロントエンドと 2 つのバージョンの Web サービス・エンドポイントを、オーダー・ディスパッチャー・コンポーネントに提供します。

Web クライアント・フロントエンドおよび 1 つのバージョンの Web サービス・エンドポイントは、以下の環境で稼働するエンタープライズ・アーカイブ (EAR) として提供されます。

環境	EAR ファイル
WebSphere Application Server バージョン 5.1	ExampleAppClient.ear ExampleAppWrapperClient.ear ExampleAppDispatchOrder.ear
WebSphere Application Server バージョン 6.1	ExampleAppClientV6.ear ExampleAppWrapperClient.ear DispatchOrderV6.ear

Rational Developer for System z は、WebSphere Application Server の Web サービス・エンドポイントを使用します。

2 番目のバージョンの Web サービス・エンドポイントは、CICS サービス・プロバイダー・アプリケーション・プログラム (DFH0XODE) として提供されます。

359 ページの図 31 は、実例アプリケーションのある構造を示しています。

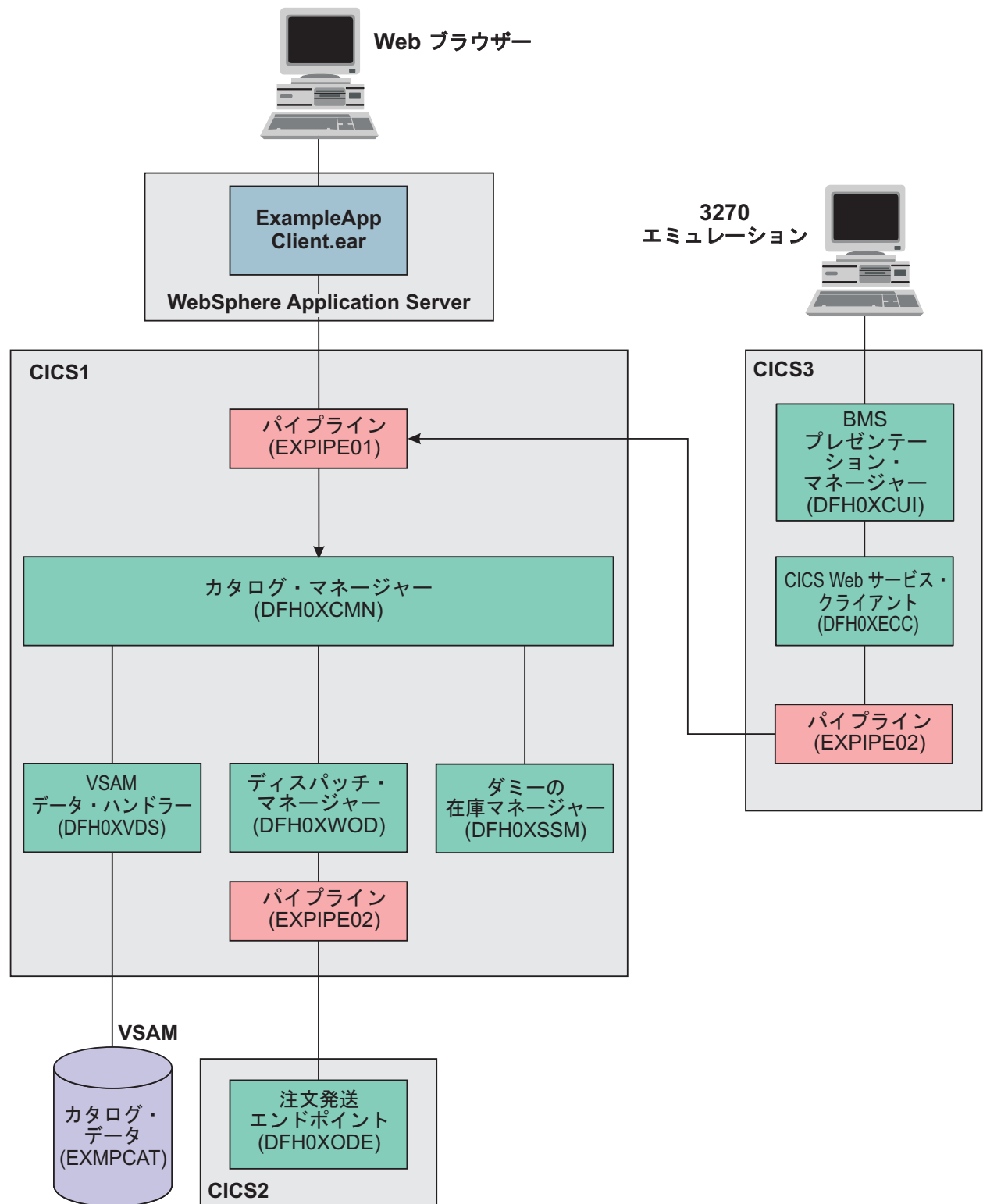


図 31. Web サービス・プロバイダーとして構成された実例アプリケーション

この構成では、アプリケーションは 2 つの異なるクライアントからアクセスされます。

- WebSphere Application Server に接続された Web ブラウザー・クライアント。ここに、ExampleAppClient.ear が配置されます。
- CICS Web サービス・クライアント DFH0XECC。このクライアントはベース・アプリケーションと同じ BMS プレゼンテーション論理を使用しますが、DFH0XGUI の代わりに DFH0XCUI モジュールを使用します。

361 ページの図 32 は、実例アプリケーションを Web サービスとして構成する別の方法を示しています。

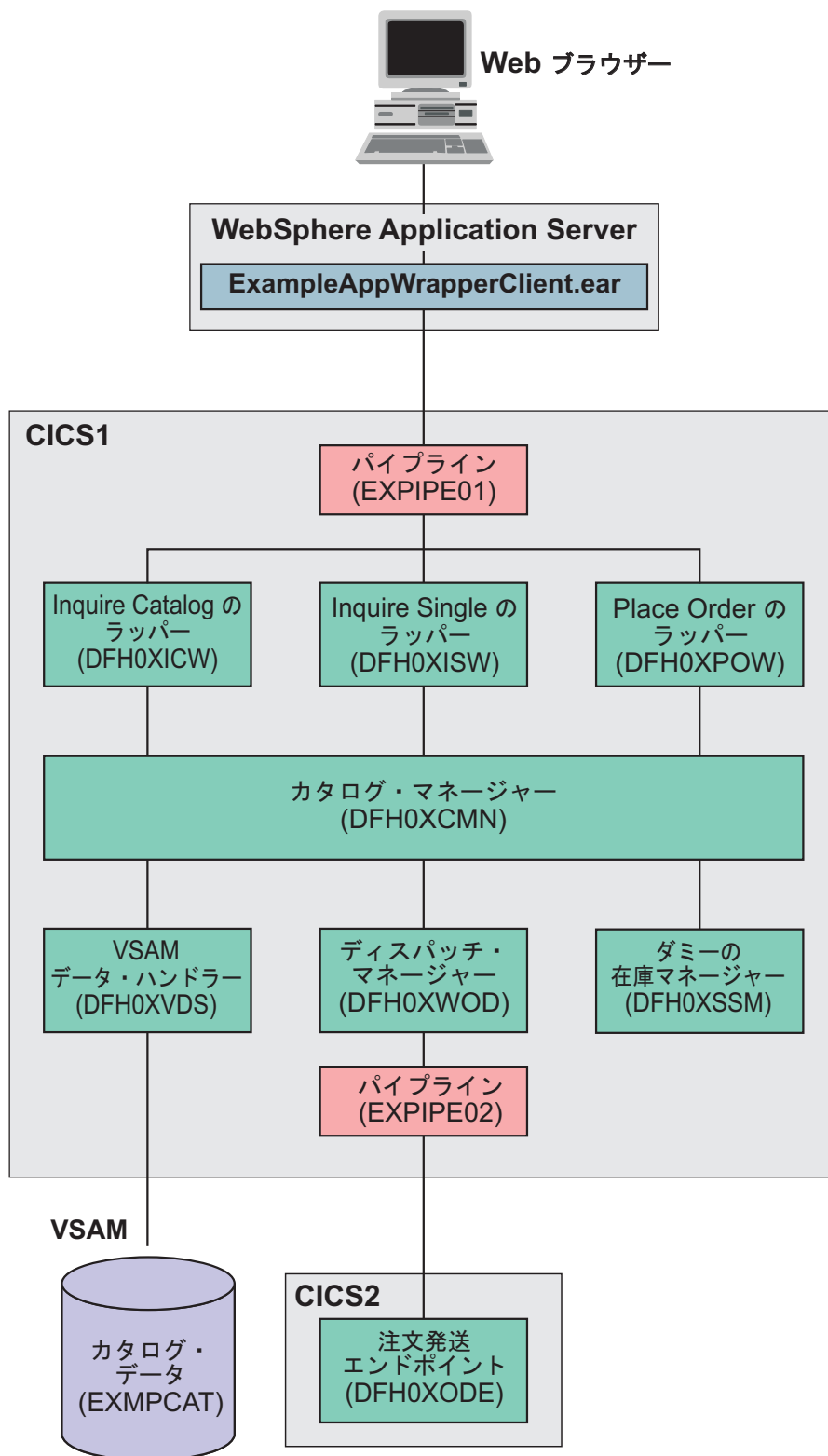


図 32. 代替の Web サービス・プロバイダー構成

この構成では、Web ブラウザー・クライアントは WebSphere Application Server に接続されます。ここに、ExampleAppWrapperClient.ear が配置されます。CICS で

は、3 つのラッパー・アプリケーション (Inquire Catalog、Inquire Single、および Place Order の機能用) がサービス・プロバイダー・アプリケーションとして配置されます。これらのアプリケーションは、ベース・アプリケーションに順にリンクします。

コード・ページ・サポートの構成

実例アプリケーションは、提供された状態では 2 つのコード化文字セットを使用します。2 つの文字セット間でデータ変換を行えるようシステムを構成する必要があります。

このタスクについて

この実例アプリケーションで使用されるコード化文字セットは、以下のとおりです。

CCSID 説明

037 EBCDIC グループ 1: 米国、カナダ (z/OS)、オランダ、ポルトガル、ブラジル、オーストラリア、ニュージーランド)

1208 UTF-8 レベル 3

ご使用の z/OS システムの変換イメージに次のステートメントを追加します。

```
CONVERSION 037,1208;  
CONVERSION 1208,037;
```

詳しくは、*CICS Transaction Server for z/OS* インストール・ガイドを参照してください。

Web サービス・クライアントおよびラッパー・プログラムの定義

プログラムの自動インストールを使用しない場合は、Web サービス・クライアントおよびラッパー・プログラムのリソース定義を定義する必要があります。

このタスクについて

1. コマンド `CEDA DEF PROG(program) G(EXAMPLE)` を使用して、ラッパー・プログラムの `PROGRAM` リソース定義を定義します。次の `COBOL` プログラムの定義を作成する必要があります。

表 15. ラッパー・モジュールの `COBOL` ソース・コードが含まれる `SDFHSAMP` メンバー

メンバー名	説明
DFH0XICW	inquireCatalog サービスのラッパー・プログラム。
DFH0XISW	inquireSingle サービスのラッパー・プログラム。
DFH0XPOW	purchaseOrder サービスのラッパー・プログラム。

2. コマンド `CEDA DEF PROG(DFH0XECC) G(EXAMPLE)` を使用して、Web サービス・クライアント・プログラム `DFH0XECC` の `PROGRAM` リソース定義を定義します。これは `COBOL` プログラムです。その他すべての属性にはデフォルト値を使用することができます。

例

次のタスク

Web サービス・サポートのインストール

実例アプリケーションに対して Web サービス・サポートを実行するには、その前に 2 つの z/OS UNIX ディレクトリーを作成し、いくつかの CICS リソース定義を作成してインストールしておく必要があります。

z/OS UNIX ディレクトリーの作成

実例アプリケーションの Web サービス・サポートでは、z/OS UNIX にシェルフ・ディレクトリーとピックアップ・ディレクトリーが必要です。

このタスクについて

シェルフ・ディレクトリーは、WEBSERVICE リソースに関連付けられている Web サービス・バインディング・ファイルを格納するために使用します。各 WEBSERVICE リソースは、順に PIPELINE に関連付けられます。シェルフ・ディレクトリーは PIPELINE リソースに管理されるため、この内容は直接変更しないでください。CICS では、PIPELINE ごとにシェルフ・ディレクトリーの下位に固有のディレクトリー構造が確保されるため、複数の PIPELINE が同じシェルフ・ディレクトリーを使用できます。

ピックアップ・ディレクトリーとは、PIPELINE と関連付けられている Web サービス・バインディング・ファイルが格納されているディレクトリーのことです。PIPELINE がインストールされると、または PERFORM PIPELINE SCAN コマンドに対する対応として、バインディング・ファイルの情報が使用され、PIPELINE に関連した WEBSERVICE 定義や URIMAP 定義が動的に作成されます。

実例アプリケーションは、シェルフ・ディレクトリーとして /var/cicsts を使用します。

パイプラインは、インストール時に XML パイプライン構成ファイル内を読み取ります。このため、これらのファイルの格納先ディレクトリーを定義することも有益です。

PIPELINE 定義の作成

パイプラインの完全な定義は、PIPELINE リソースとパイプライン構成ファイルで構成されます。このファイルには、Web サービス要求および Web サービス応答がパイプラインをパススルーするときに、これらに作用するメッセージ・ハンドラーの詳細が記述されています。

このタスクについて

実例アプリケーションでは、CICS 提供の SOAP 1.1 ハンドラーを使用して、インバウンド要求およびアウトバウンド要求の SOAP エンベロープを処理します。CICS には、サービス・プロバイダーおよびサービス・リクエスターで使用できるサンプルのパイプライン構成ファイルが用意されています。

複数の WEBSERVICE が 1 つの PIPELINE を共用できるため、実例アプリケーションのインバウンド要求に定義するパイプラインは 1 つのみにする必要があります。ただし、1 つの PIPELINE を、同時にプロバイダー・パイプラインとリクエスター・パイプラインの両方になるように構成することはできないため、アウトバウンド要求に対しては 2 番目の PIPELINE を定義する必要があります。

1. CEDA トランザクションを使用して、サービス・プロバイダーの PIPELINE 定義を作成します。
 - a. CEDA DEF PIPE(EXPIPE01) G(EXAMPLE) と入力します。または、CICS 提供のグループ DFH\$EXWS から PIPELINE 定義をコピーすることもできます。
 - b. 以下の追加属性を入力します。

```
STATUS(Enabled)
CONFIGFILE(/usr/lpp/cicsts
           /samples/pipelines/basicsoap11provider.xml)
SHELF(/var/cicsts)
WSDIR(/usr/lpp/cicsts/samples/webservices/wsbind/provider/)
```

z/OS UNIX の項目では大/小文字が区別され、デフォルトの CICS z/OS UNIX インストール・ルートは /usr/lpp/cicsts であることが前提になっていることに注意してください。

2. CEDA トランザクションを使用して、サービス・リクエスターの PIPELINE 定義を作成します。
 - a. CEDA DEF PIPE(EXPIPE02) G(EXAMPLE) と入力します。または、CICS 提供のグループ DFH\$EXWS から PIPELINE 定義をコピーすることもできます。
 - b. 以下の追加属性を入力します。

```
STATUS(Enabled)
CONFIGFILE(/usr/lpp/cicsts
           /samples/pipelines/basicsoap11requester.xml)
SHELF(/var/cicsts)
WSDIR(/usr/lpp/cicsts/samples/webservices/wsbind/requester/)
```

z/OS UNIX の項目では大/小文字が区別され、デフォルトの CICS z/OS UNIX インストール・ルートは /usr/lpp/cicsts であることが前提になっていることに注意してください。

TCPIPSERVICE の作成

クライアントは HTTP トランスポートを介して Web サービスに接続するため、TCPIPSERVICE を定義してインバウンド HTTP トラフィックを受信する必要があります。

インバウンド HTTP 要求を処理するには、CEDA トランザクションを使用して TCPIPSERVICE 定義を作成します。

1. CEDA DEF TCPIPSERVICE(EXMPPORT) G(EXAMPLE) と入力します。または、CICS 提供のグループ DFH\$EXWS から TCPIPSERVICE 定義をコピーすることもできます。
2. 以下の追加属性を入力します。

```
URM(DFHWBAAX)
```

PORTNUMBER(*port*) ここで、*port* は CICS システムにおける未使用のポート番号です。

PROTOCOL(HTTP)

TRANSACTION(CWXN)

3. その他の属性には、すべてデフォルト値を使用します。

WEBSERVICE リソースおよび URIMAP リソースの動的なインストール

Web サービスとして公開される各機能には、SOAP BODY の着信 XML とプログラムの COMMAREA インターフェース間をマップするための WEBSERVICE リソースと、着信要求を正しい PIPELINE および WEBSERVICE に送信する URIMAP リソースが必要です。RDO を使用して WEBSERVICE リソースと URIMAP リソースを定義してインストールできますが、PIPELINE リソースをインストールした場合は、CICS を使用しても、これらのリソースを動的に作成できます。

PIPELINE リソースをインストールします。以下のコマンドを使用します。

```
CEDA INSTALL PIPELINE(EXPIPE01) G(EXAMPLE)
```

```
CEDA INSTALL PIPELINE(EXPIPE02) G(EXAMPLE)
```

各 PIPELINE リソースをインストールすると、CICS は、PIPELINE の WSDIR 属性で指定されたディレクトリー (ピックアップ・ディレクトリー) をスキャンします。このディレクトリーの Web サービス・バインディング・ファイルごとに、つまり、.wsbind という接尾部を持つファイルごとに、CICS は、WEBSERVICE および URIMAP のいずれかが存在しなかった場合、これらをインストールします。バインディング・ファイル内の情報の方が既存のリソースよりも新しい場合、既存のリソースは置き換えられます。

PIPELINE が後で使用不可になり、破棄されると、関連付けられていたすべての WEBSERVICE リソースおよび URIMAP リソースも破棄されます。

PIPELINE をインストール済みの場合は、PERFORM PIPELINE SCAN コマンドを使用して、PIPELINE のピックアップ・ディレクトリーのスキャンを開始してください。

PIPELINE をインストールすると、プロバイダー PIPELINE 用の以下の WEBSERVICE と、それに関連した URIMAP がシステムにインストールされます。

dispatchOrder

dispatchOrderEndpoint

inquireCatalog

inquireCatalogClient

inquireCatalogWrapper

inquireSingle

inquireSingleClient

inquireSingleWrapper

placeOrder

placeOrderClient

placeOrderWrapper

WEBSERVICE の名前は、Web サービス・バインディング・ファイルの名前から得られます。URIMAP の名前は動的に生成されます。CEMT INQUIRE WEBSERVICE コマンドを使用すると、リソースを表示できます。

```
I WEBS
STATUS: RESULTS - OVERTYPE TO MODIFY
Webs(dispatchOrder                ) Pip(EXPIPE02)
  Ins Ccs(00000)
Webs(dispatchOrderEndpoint        ) Pip(EXPIPE01)
  Ins Ccs(00000) Uri(£135550 ) Pro(DFH0XODE) Com
Webs(inquireCatalog                ) Pip(EXPIPE01)
  Ins Ccs(00000) Uri(£135551 ) Pro(DFH0XCMN) Com
Webs(inquireCatalogClient          ) Pip(EXPIPE02)
  Ins Ccs(00000)
Webs(inquireCatalogWrapper         ) Pip(EXPIPE01)
  Ins Ccs(00000) Uri(£135560 ) Pro(DFH0XICW) Cha
Webs(inquireSingle                 ) Pip(EXPIPE01)
  Ins Ccs(00000) Uri(£135570 ) Pro(DFH0XCMN) Com
Webs(inquireSingleClient            ) Pip(EXPIPE02)
  Ins Ccs(00000)
Webs(inquireSingleWrapper           ) Pip(EXPIPE01)
  Ins Ccs(00000) Uri(£136010 ) Pro(DFH0XISW) Cha
Webs(placeOrder                    ) Pip(EXPIPE01)
  Ins Ccs(00000) Uri(£135580 ) Pro(DFH0XCMN) Com
Webs(placeOrderClient              ) Pip(EXPIPE02)
  Ins Ccs(00000)
Webs(placeOrderWrapper              ) Pip(EXPIPE01)
  Ins Ccs(00000) Uri(£135571 ) Pro(DFH0XP0W) Cha
```

この表示には、PIPELINE や URIMAP の名前、および各 WEBSERVICE に関連したターゲット・プログラムの名前が表示されています。この例では、WEBSERVICE はアウトバウンド要求を対象にしているため、WEBSERVICE(dispatchOrder) には URIMAP もターゲット・プログラムも表示されないことに注意してください。WEBSERVICE(dispatchOrderEndpoint) は、注文発送サービスのローカル側 CICS 実装を表しています。

RDO による WEBSERVICE リソースの作成

PIPELINE スキャン機能を使用して WEBSERVICE リソースをインストールする代わりに、オンライン・リソース定義 (RDO) を使用して WEBSERVICE リソースを作成し、インストールすることができます。

始める前に

重要: RDO を使用して WEBSERVICE リソースおよび URIMAP リソースを定義する場合は、Web サービス・バインディング・ファイルが PIPELINE のピックアップ・ディレクトリに**存在しない**ことを確認する必要があります。

1. CEDA トランザクションを使用して、実例アプリケーションの INQUIRE CATALOG 機能の WEBSERVICE 定義を作成します。
 - a. CEDA DEF WEBSERVICE(EXINQCWS) G(EXAMPLE) と入力します。
 - b. 以下の追加属性を入力します。

```
PIPELINE(EXPIPE01)
WSBIND(/usr/lpp/cicsts/samples
      /webservices/wsbinding/provider/inquireCatalog.wsbinding)
```


2. 実例アプリケーションの以下の機能ごとに、前のステップを繰り返します。

機能	WEBSERVICE 名	PIPELINE 属性	WSBIND 属性
INQUIRE SINGLE ITEM	EXINQSWWS	EXPIPE01	/usr/lpp/cicsts/samples /webservices/wsbinding /provider/inquireSingle.wsbinding
PLACE ORDER	EXORDRWS	EXPIPE01	/usr/lpp/cicsts/samples /webservices/wsbinding /provider/placeOrder.wsbinding
DISPATCH STOCK	EXODRQWS	EXPIPE02	/usr/lpp/cicsts/samples /webservices/wsbinding /requester/dispatchOrder.wsbinding
DISPATCH STOCK endpoint (オプション)	EXODEPWS	EXPIPE01	/usr/lpp/cicsts/samples /webservices/wsbinding /provider/dispatchOrderEndpoint.wsbinding

RDO による URIMAP リソースの作成

PIPELINE スキャン機能を使用して URIMAP リソースをインストールする代わりに、オンライン・リソース定義 (RDO) を使用して URIMAP リソースを作成し、インストールすることができます。

始める前に

重要: RDO を使用して WEBSERVICE リソースおよび URIMAP リソースを定義する場合は、Web サービス・バインディング・ファイルが PIPELINE のピックアップ・ディレクトリーに存在しないことを確認する必要があります。

1. CEDA トランザクションを使用して、実例アプリケーションの INQUIRE CATALOG 機能の URIMAP 定義を作成します。
 - a. CEDA DEF URIMAP(INQCURI) G(EXAMPLE) と入力します。
 - b. 以下の追加属性を入力します。

```

USAGE(PIPELINE)
HOST(*)
PATH(/exampleApp/inquireCatalog)
TCPIPService(SOAPPORT)
PIPELINE(EXPIPE01)
WEBSERVICE(EXINQCWS)

```

2. 実例アプリケーションの残りの機能ごとに、前のステップを繰り返します。
URIMAP には、以下の名前を使用します。

機能	URIMAP 名
INQUIRE SINGLE ITEM	INQSURI
PLACE ORDER	ORDRURI
DISPATCH STOCK	必要なし
DISPATCH STOCK endpoint (オプション)	ODEPURI

- a. URIMAP ごとに以下に示す別個の属性を指定します。

機能	URIMAP 名	PATH	WEBSERVICE
INQUIRE SINGLE ITEM	INQSURI	/exampleApp/inquireSingle	EXINQSWs
PLACE ORDER	ORDRURI	/exampleApp/placeOrder	EXORDRWS
DISPATCH STOCK endpoint (オブ シヨナル)	ODEPURI	/exampleApp/dispatchOrder	EXODEPWS

- b. 以下の追加属性を入力します。これらの属性は、すべての URIMAP で同じです。

USAGE(PIPELINE)

HOST(*)

TCPIPService(SOAPPORT)

PIPELINE(EXPIPE01)

インストールの完了

インストールを完了するには、リソース定義が格納されている RDO グループをインストールします。

CICS 端末で、CEDA I G(EXAMPLE) というコマンドを入力します。

タスクの結果

これで、アプリケーションを使用する準備ができました。

Web クライアントの構成

Web クライアントを使用する前に、サポートされるいずれかの環境にクライアントのエンタープライズ・アーカイブ (EAR) を配置して、ご使用の CICS システムで適切なエンドポイントを呼び出すよう構成しておく必要があります。

このタスクについて

サポートされる環境は次のとおりです。

- WebSphere Application Server バージョン 5 リリース 1 以上
- WebSphere 単体テスト環境を備えた WebSphere Studio Application Developer バージョン 5 リリース 1 以上
- WebSphere 単体テスト環境を備えた WebSphere Studio Enterprise Developer バージョン 5 リリース 1 以上

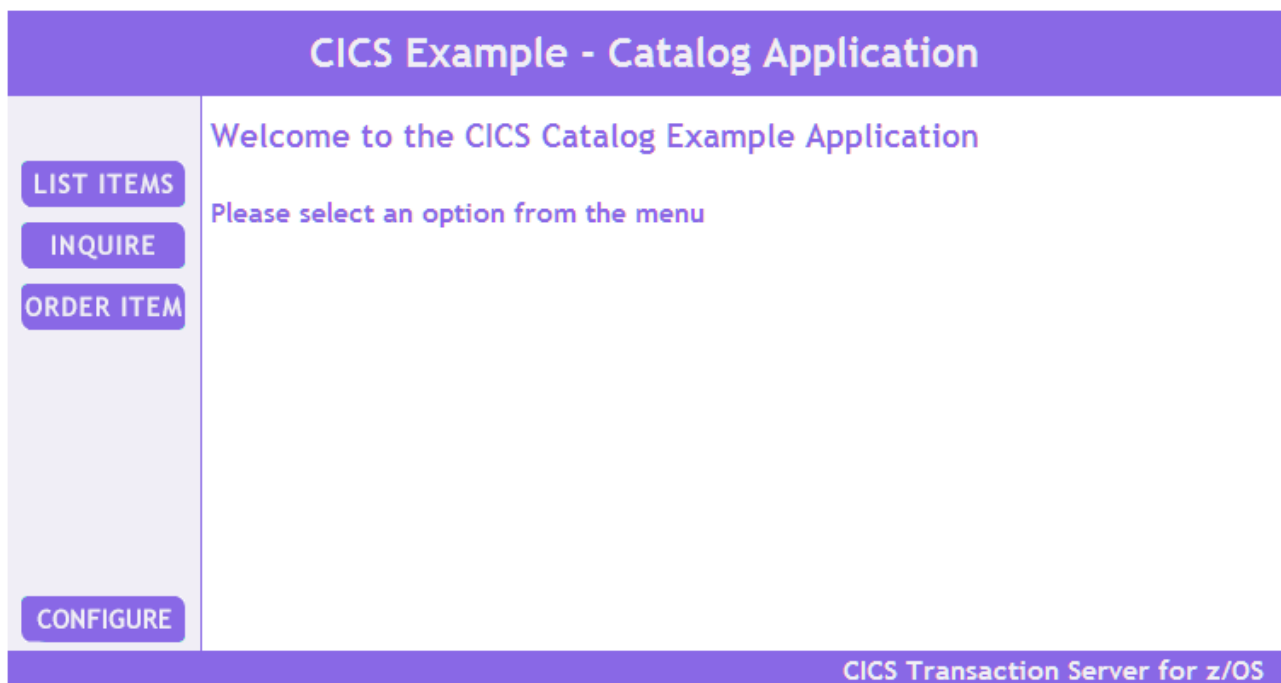
ExampleAppClientV6.ear クライアント・アプリケーションでサポートされる環境は次のとおりです。

- WebSphere Application Server バージョン 6
- WebSphere 単体テスト環境を備えた Rational Application Developer バージョン 6 以上

- WebSphere 単体テスト環境を備えた WebSphere Developer for zSeries バージョン 6 以上

EAR ファイルは、z/OS UNIX の *hlq/samples/webservices/client* ディレクトリー内にあります。

1. WebSphere バージョン 5 製品を使用する場合、Web クライアントを開始するには、Web ブラウザーで、<http://myserver:9080/ExampleAppClientWeb/> と入力します。ここで、*myserver* は、Web サービス・クライアントのインストール先サーバーのホスト名です。WebSphere バージョン 6 製品を使用する場合、Web クライアントを開始するには、Web ブラウザーで、<http://myserver:9080/ExampleAppClientV6Web/> と入力します。 実例アプリケーションにより、次のページが表示されます。



2. 「構成 (CONFIGURE)」ボタンをクリックして、構成ページを表示します。 構成ページが表示されます。

CICS Example - Catalog Application	
LIST ITEMS INQUIRE ORDER ITEM BACK	Configure Application
	Inquire Catalog Service Endpoint
	Current <code>http://myCicsServer:9999/exampleApp/inquireCatalog</code>
	New <code>http://myCicsServer:9999/exampleApp/inquireCatalog</code>
	Inquire Item Service Endpoint
	Current <code>http://myCicsServer:9999/exampleApp/inquireSingle</code>
	New <code>http://myCicsServer:9999/exampleApp/inquireSingle</code>
	Place Order Service Endpoint
	Current <code>http://myCicsServer:9999/exampleApp/placeOrder</code>
	New <code>http://myCicsServer:9999/exampleApp/placeOrder</code>
SUBMIT RESET	
CICS Transaction Server for z/OS	

- Web サービスの新しいエンドポイントを入力します。 構成するエンドポイントには次の 3 つがあります。

Inquire Catalog

Inquire Item

Place Order

- URL では、ストリング「myCicsServer」を、CICS が動作しているシステムの名前に置き換えます。
 - ポート番号「9999」を、TCPIP SERVICE リソースで構成したポート番号 (この例では、30000) に置き換えます。
- 「SUBMIT (発注登録)」ボタンをクリックします。

タスクの結果

これで、Web アプリケーションを実行する準備ができました。

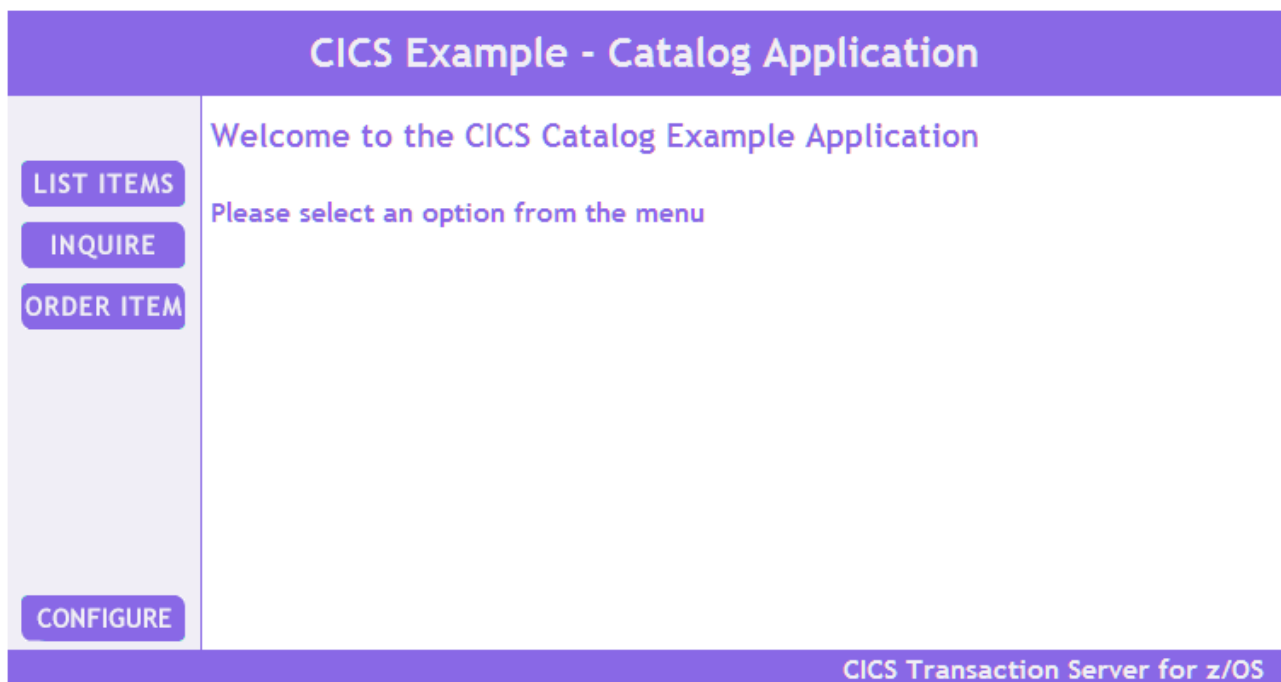
次のタスク

注: Web サービスによって呼び出される URL は、HTTP セッションで保管されます。したがって、ブラウザが初めてクライアントに接続されるたびに、この構成ステップを繰り返す必要があります。

Web サービス対応アプリケーションの実行

実例アプリケーションは Web ブラウザーから起動できます。

1. Web ブラウザーで、次のように入力します。http://myserver:9080/ExampleAppClientWeb/。ここで、*myserver* は、Web サービス・クライアントのインストール先サーバーのホスト名です。 実例アプリケーションにより、次のページが表示されます。



2. 「**INQUIRE (問い合わせ)**」ボタンをクリックします。 実例アプリケーションにより、次のページが表示されます。

CICS Example - Catalog Application

INQUIRE

ORDER ITEM

BACK

CONFIGURE

Enter Catalog Item Reference Number

Start List From Item Number

0010

SUBMIT

CICS Transaction Server for z/OS

3. 品目番号を入力し、「**SUBMIT (発注登録)**」 ボタンをクリックします。

ヒント: ベース・アプリケーションには、0010、0020、... の順に 0210 まで品目番号が付与されます。

アプリケーションは、入力した品目番号から開始されたカタログの品目リストを含む、次のページを表示します。

CICS Example - Catalog Application						
Item Details - Select Item to Place Order						
LIST ITEMS INQUIRE ORDER ITEM BACK CONFIGURE	Item	Description	In Stock	On Order	Cost	Select
	0010	Ball Pens Black 24pk	13	0	£2.90	☐
	0020	Ball Pens Blue 24pk	2	50	£2.90	☐
	0030	Ball Pens Red 24pk	38	0	£2.90	☐
	0040	Ball Pens Green 24pk	71	0	£2.90	☒
	0050	Pencil with eraser 12pk	70	0	£1.78	☐
	0060	Highlighters Assorted 5pk	11	40	£3.89	☐
	0070	Laser Paper 28-lb 108 Bright 500/ream	90	20	£7.44	☐
	0080	Laser Paper 28-lb 108 Bright 2500/case	25	0	£33.54	☐
	0090	Blue Laser Paper 20lb 500/ream	22	0	£5.35	☐
	0100	Green Laser Paper 20lb 500/ream	3	20	£5.35	☐
	0110	IBM Network Printer 24 - Toner cart	8	0	£169.56	☐
	0120	Standard Diary: Week to view 8 1/4x5 3/4	7	0	£25.99	☐
	0130	Wall Planner: Eraseable 36x24	3	0	£18.85	☐
	0140	70 Sheet Hard Back wire bound notepad	84	0	£5.89	☐
0150	Sticky Notes 3x3 Assorted Colors 5pk	22	45	£5.35	☐	
						SUBMIT

4. 注文する品目を選択します。
 - a. 注文する品目の「**Select (選択)**」列のラジオ・ボタンをクリックします。
 - b. 「**SUBMIT (発注登録)**」ボタンをクリックします。

アプリケーションにより、次のページが表示されます。

CICS Example - Catalog Application

LIST ITEMS

INQUIRE

BACK

CONFIGURE

Enter Order Details

Item Reference Number

0040

Quantity

001

User Name

AUSER

Department Name

CICS1

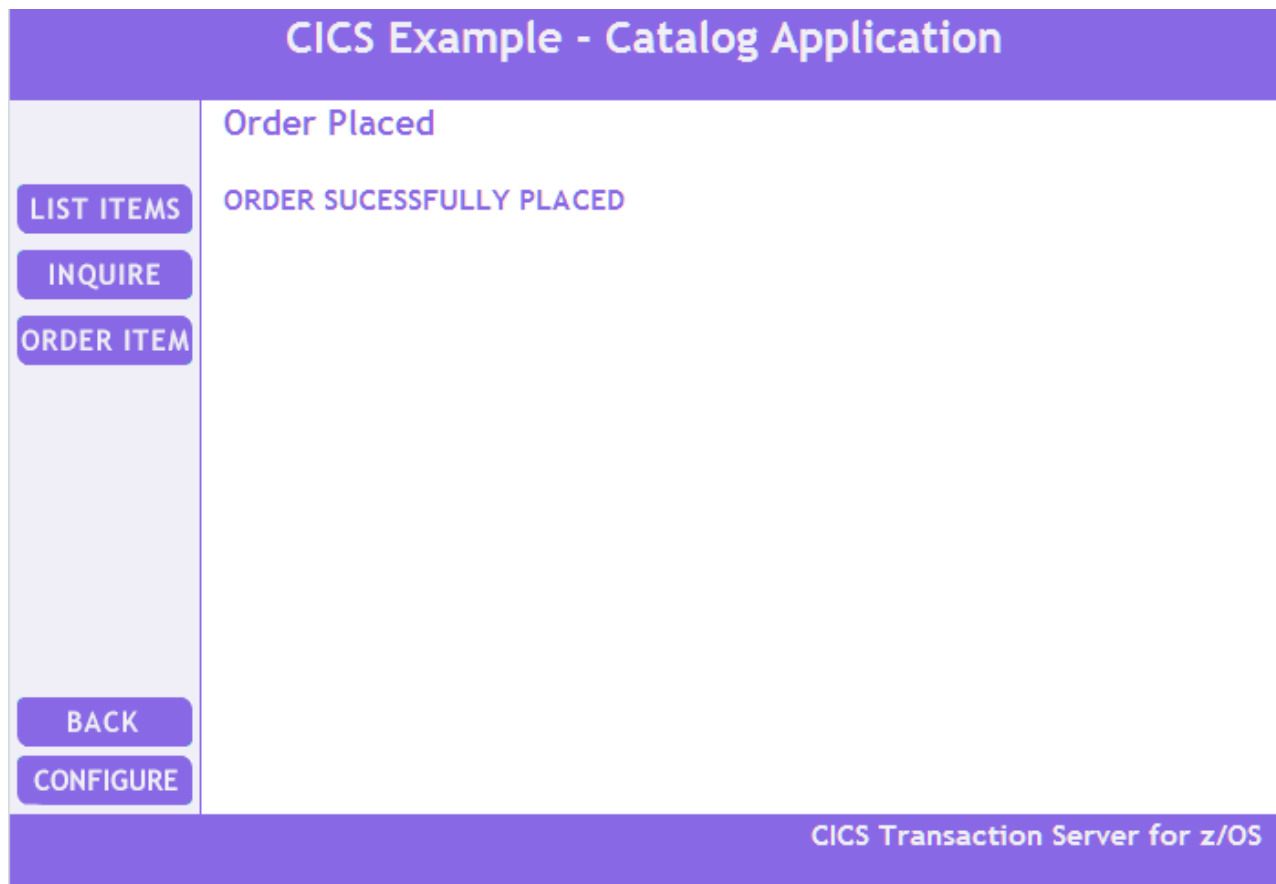
SUBMIT

CICS Transaction Server for z/OS

5. 発注するには、以下の情報を入力します。

- 「Quantity (数量)」フィールドに値を入力します。 注文する品目の数を指定します。
- 「ユーザー名」フィールドに値を入力します。 1 から 8 文字のSTRINGを入力します。 ベース・アプリケーションは、ここに入力された値を検査しません。
- 「Department Name (部門名)」フィールドに値を入力します。 1 から 8 文字のSTRINGを入力します。 ベース・アプリケーションは、ここに入力された値を検査しません。
- 「**SUBMIT (発注登録)**」ボタンをクリックします。

発注が行われたか確認するため、アプリケーションにより次のページが表示されます。



実例アプリケーションの配置

Web サービス・アシスタントを使用すると、実例アプリケーションの一部を Web サービスとして配置できます。実例アプリケーションは、提供されたままの状態です。この作業を実行することなく動作しますが、独自のアプリケーションを配置して実例アプリケーションを拡張する場合は、類似の作業を実行する必要があります。

プログラム・インターフェースの抽出

CICS Web サービス・アシスタントを使用してプログラムを配置するには、プログラムの COMMAREA またはコンテナー・インターフェースに一致するコピーブックを作成する必要があります。

このタスクについて

この例では、中央のカatalog・マネージャー・プログラム (DFH0XCMN) の INQUIRE SINGLE ITEM 機能が、Web サービスとして配置されます。このプログラムに対するインターフェースは、COMMAREA です。COMMAREA の構造は、コピーブック DFH0XCPI で次のように定義されます。

```
*   カタログ COMMAREA 構造
      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).
*   Inquire Catalog (発注) で使用されているフィールド
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).
*   Inquire Single (1 件の問い合わせ) で使用されているフィールド
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).
*   Place Order (発注) で使用されているフィールド
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).

```

コピーブックでは、INQUIRE CATALOG、INQUIRE SINGLE ITEM、および PLACE ORDER の機能それぞれに対して、3 つの異なるインターフェースが定義されます。これらのインターフェースは、コピーブック内で互いにオーバーレイされています。ただし、DFHLS2WS ユーティリティは、REDEFINES ステートメントをサポートしていません。したがって、Inquire Single (1 件の問い合わせ) 機能に関連するセクションのみを結合コピーブックから抽出する必要があります。

```

*   カタログ COMMAREA 構造
03 CA-REQUEST-ID             PIC X(6).
03 CA-RETURN-CODE            PIC 9(2) DISPLAY.
03 CA-RESPONSE-MESSAGE      PIC X(79).
*   Inquire Single (1 件の問い合わせ) で使用されているフィールド
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).

```

再定義の要素 CA-REQUEST-SPECIFIC は、除去され、Inquire Single (1 件の問い合わせ) 機能に対してこの要素を再定義したコピーブックの部分によっ

て置き換えられます。これで、このコピーブックは、Web サービス・アシスタントとの組み合わせ使用に適合するようになりました。

このコピーブックは、コピーブック DFH0XCP4 として実例アプリケーションに付属しています。

Web サービス・アシスタント・プログラム DFHLS2WS の実行

CICS Web サービス・アシスタントは、2 つのバッチ・プログラムで構成されており、既存の CICS アプリケーションを Web サービスに変換するのに役立ちます。また、Web サービス・アシスタントを使用すると、CICS アプリケーションが、外部のプロバイダーによって提供された Web サービスを使用できるようになります。プログラム DFHLS2WS は、言語構造を変換して Web サービス・バインディング・ファイルと Web サービス記述を生成します。

1. 付属のサンプル JCL を適切な作業ファイルにコピーします。JCL は、`samples/webservices/JCL/LS2WS` にあります。
2. 有効な JOB カードを JCL に追加します。
3. DFHLS2WS のパラメーターを指定します。実例アプリケーションの INQUIRE SINGLE ITEM 機能に必要なパラメーターは、次のとおりです。

```
//INPUT.SYSUT1 DD *  
LOGFILE=/u/exampleapp/wsbind/inquireSingle.log  
PDSLIB=CICSHLQ.SDFHSAMP  
REQMEM=DFH0XCP4  
RESPMEM=DFH0XCP4  
LANG=COBOL  
PGMNAME=DFH0XCMN  
PGMINT=COMMAREA  
URI=exampleApp/inquireSingle  
WSBIND=/u/exampleapp/wsbind/inquireSingle.wsbind  
WSDL=/u/exampleapp/wsd1/inquireSingle.wsd1  
*/
```

パラメーターは以下のとおりです。

LOGFILE=/u/exampleapp/wsbind/inquireSingle.log

DFHLS2WS から診断情報を記録するときに使用するファイル。このファイルを使用するのは、通常、IBM のソフトウェア・サポート組織のみです。

PDSLIB=CICSHLQ.SDFHSAMP

要求構造と応答構造を定義するコピーブックが、Web サービス・アシスタントによって検索される区分データ・セット (PDS) の名前。この例では、CICS によってインストールされたデータ・セットの SDFHSAMP になります。

REQMEM=DFH0XCP4

RESPMEM=DFH0XCP4

これらのパラメーターは、プログラムに対する要求と応答の言語構造を定義します。この例では、要求と応答の構造は同じであり、同じコピーブックによって定義されます。

LANG=COBOL

ターゲット・プログラムおよびデータ構造は、COBOL で記述されます。

PGMNAME=DFH0XCMN

Web サービス要求を受け取ると呼び出されるターゲット・プログラムの名前。

PGMINT=COMMAREA

ターゲット・プログラムは、COMMAREA インターフェースによって呼び出されます。

URI=exampleApp/inquireSingle

URI の固有の部分。この URI は、生成された Web サービス定義で使用され、着信要求を正しい Web サービスにマップする URIMAP リソースを作成するために使用されます。指定された値により、外部クライアントに対して、サービスが次の場所で利用可能になります。

`http://mycicsserver:myport/exampleApp/inquireSingle`

ここで、*mycicsserver* および *myport* は、この WEBSERVICE がインストールされている CICS サーバーのアドレスおよびポートを表します。

注: このパラメーターには、先頭に「/」がありません。

WSBIND=/u/exampleapp/wsbind/inquireSingle.wsbind

Web サービス・バインディング・ファイルの書き込み先となる z/OS UNIX 上の場所。

注: このファイルが PIPELINE スキャン機能と組み合わせて使用される場合、このファイルには拡張子 *.wsbind* が必要です。

WSDL=/u/exampleapp/wsd1/inquireSingle.wsd1

生成された Web サービス記述が格納されているファイルの書き込み先となる z/OS UNIX 上の場所。Web サービス・バインディング・ファイルと、これに対応する Web サービス記述にマッチングする名前を使用する習慣をつけておくことをお勧めします。

ヒント: 従来、Web サービス記述が格納されているファイルの拡張子は *.wsdl* です。

Web サービス記述は、クライアントが Web サービスにアクセスするために必要な情報を提供します。ここには、要求と応答の XML スキーマ定義と、サービスのロケーション情報が格納されています。

4. ジョブを実行します。Web サービス記述と Web サービス・バインディング・ファイルが、指定のロケーションに作成されます。
5. Web サービス記述内のサービス・ロケーションをカスタマイズします。生成したままの状態では、<service> エレメントの内容は次のようになっています。

```
<service name="DFHCMNService">
  <port binding="tns:DFH0XCMNHTTPSoapBinding" name="DFH0XCMNPort">
    <soap:address location="http://my-server:my-port/exampleApp/inquireSingle"/>
  </port>
</service>
```

Web サービス記述をクライアントに公開するには、その前に次の変更を行う必要があります。

- a. *my-server* を CICS サーバー・ロケーションに置き換えます。

b. *my-port* をポート番号に置き換えます。

生成される WSDL 文書の例

```
<?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:element name="DFH0XCMNOperation" nillable="false" type="tns:ProgramInterface"/>
    </xsd:schema>
    <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://www.DFH0XCMN.DFH0XCP4.Response.com" xmlns:tns="http://www.DFH0XCMN.DFH0XCP4.Response.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      ... schema content for the reply is removed
    </xsd:schema>
  </types>
  <message name="DFH0XCMNOperationResponse">
    <part element="resns:DFH0XCMNOperationResponse" name="ResponsePart"/>
  </message>
  <message name="DFH0XCMNOperationRequest">
    <part element="reqns:DFH0XCMNOperation" name="RequestPart"/>
  </message>
  <porttype name="DFH0XCMNPort">
    <operation name="DFH0XCMNOperation">
      <input message="tns:DFH0XCMNOperationRequest" name="DFH0XCMNOperationRequest"/>
      <output message="tns:DFH0XCMNOperationResponse" name="DFH0XCMNOperationResponse"/>
    </operation>
  </porttype>
  <binding name="DFH0XCMNHTTPSoapBinding" type="tns:DFH0XCMNPort">
    <!-- This soap:binding indicates the use of SOAP 1.1 -->
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <!-- This soap:binding indicates the use of SOAP 1.2 -->
    <!-- <soap:binding style="document" transport="http://www.w3.org/2003/05/soap-http"/> -->
    <operation name="DFH0XCMNOperation">
      <soap:operation soapAction="" style="document"/>
      <input name="DFH0XCMNOperationRequest">
        <soap:body parts="RequestPart" use="literal"/>
      </input>
      <output name="DFH0XCMNOperationResponse">
        <soap:body parts="ResponsePart" use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="DFH0XCMNService">
    <port binding="tns:DFH0XCMNHTTPSoapBinding" name="DFH0XCMNPort">
      <!-- This soap:address indicates the location of the Web service over HTTP.

```

```

Please replace "my-server" with the TCPIP host name of your CICS region.
Please replace "my-port" with the port number of your CICS TCPIP SERVICE. -->
<soap:address location="http://my-server:my-port/exampleApp/inquireSingles.log"/>
<!-- This soap:address indicates the location of the Web service over HTTPS. -->
<!-- <soap:address location="https://my-server:my-port/exampleApp/inquireSingles.log"/> -->
<!-- This soap:address indicates the location of the Web service over MQSeries.
Please replace "my-queue" with the appropriate queue name. -->
<!-- <soap:address location="jms:/queue?destination=my-queue&connectionFactory=()&
targetService=/exampleApp/inquireSingles.log&initialContextFactory=com.ibm.mq.jms.NoJndi" /> -->
</port>
</service>
</definitions>

```

Web サービス・バインディング・ファイルの配置

DFHLS2WS によって作成された Web サービス・バインディング・ファイルは、PIPELINE リソースのインストール時に CICS 領域に動的に配置されます。

このタスクについて

PIPELINE スキャン・コマンドを実行すると、CICS は、CEMT P PIPELINE() SCAN を介して明示的に、または PIPELINE インストール時に自動的にピックアップ・ディレクトリーをスキャンして、Web サービス・バインディング・ファイル、つまり .wsbind という拡張子を持つファイル名を検索します。CICS は、見つかったバインディング・ファイルごとに、WEBSERVICE リソースをインストールするかどうかを判別します。

URIMAP リソースも JCL に用意されているように作成され、これにより、インストール済み WEBSERVICE と、WEBSERVICE のインストール先 PIPELINE に URI がマップされます。スキャンされた WEBSERVICE が破棄されると、これに関連付けられている URIMAP も破棄されます。

1. プロバイダー・パイプラインの PIPELINE 定義である PIPELINE(EXPIPE01) を変更します。WSDIR パラメーターを /u/exampleapp/wsbind に変更します。このピックアップ・ディレクトリーには、DFHLS2WS を使用して生成した Web サービス・バインディング・ファイルが格納されています。
2. アプリケーションが使用するその他の Web サービス・バインディング・ファイルを同じディレクトリーにコピーします。この例では、コピーの対象ファイルは次のとおりです。

```

inquireCatalog
placeOrder

```

これらのファイルは、ディレクトリー /usr/lpp/cicsts/samples/webservices/wsbind/provider にあります。

3. PIPELINE をインストールします。CICS は、Web サービス・バインディング・ファイルを基にして、WEBSERVICE リソースと URIMAP リソースを作成します。

ベース・アプリケーションのコンポーネント

表 16. BMS マップを含む SDFHSAMP メンバー

メンバー名	説明
DFH0XS1	「Main Menu (メインメニュー)」画面のマップ (EXMENU) および「Details of your order (注文の詳細)」画面のマップ (EXORDR) で構成されるマップ・セットの BMS マクロ。
DFH0XS2	「Inquire Catalog (カタログの問い合わせ)」画面のマップ (EXINQC) で構成されるマップ・セットの BMS マクロ。
DFH0XS3	「Configure CICS example catalog application (CICS 実例カタログ・アプリケーションの構成)」画面のマップ (EXCONF) で構成されるマップ・セットの BMS マクロ。
DFH0XM1	DFH0XS1 をアセンブルすることによって生成される COBOL コピーブック。DFH0XGUI および DFH0XCUI には、このコピーブックが組み込まれています。
DFH0XM2U	DFH0XS2 をアセンブルして、コピーブックのプログラミングを容易にするために索引付きの配列構造を組み込むようその結果を編集することによって生成される COBOL コピーブック。DFH0XGUI および DFH0XCUI には、このコピーブックが組み込まれています。
DFH0XM3	DFH0XS3 をアセンブルすることによって生成される COBOL コピーブック。DFH0XCFG には、このコピーブックが組み込まれています。

表 17. ベース・アプリケーションの COBOL ソースが含まれる SDFHSAMP メンバー

メンバー名	説明
DFH0XCFG	VSAM 構成ファイルを読み取って更新するために、トランザクション ECFG によって呼び出されるプログラム。
DFH0XCMN	カタログ・アプリケーションのコントローラー・プログラム。すべての要求がこのプログラムをパススルーします。
DFH0XGUI	端末ユーザーへの BMS マップの送信および端末ユーザーからのマップの受信を管理するために、トランザクション EGUI によって呼び出されるプログラム。これはプログラム DFH0XCMN にリンクします。
DFH0XODE	注文発送 Web サービスの 2 つのバージョンのエンドポイントのいずれか。これは、CICS で稼働するバージョンです。このエンドポイントは、戻りの COMMAREA でテキスト「Order in dispatch」を設定します。
DFH0XSDS	VSAM カatalog・ファイルがセットアップされていない場合に、アプリケーションが操作できるスタブ化版またはダミー版のデータ・ストア・プログラム。このプログラムは、VSAM ファイルに保管されたデータではなく、プログラムで定義されたデータを使用します。
DFH0XSOD	スタブ化版の注文発送プログラム。このプログラムは、COMMAREA 内の戻りコードを 0 に設定して、呼び出し元に戻します。これは、アウトバウンド Web サービスが必要ないときに使用されます。

表 17. ベース・アプリケーションの COBOL ソースが含まれる SDFHSAMP メンバー (続き)

メンバー名	説明
DFH0XSSM	スタブ化版の在庫マネージャー (補充) プログラム。このプログラムは、COMMAREA 内の戻りコードを 0 に設定して、呼び出し元に戻します。
DFH0XVDS	VSAM 版のデータ・ストア・プログラム。このプログラムは VSAM ファイルにアクセスして、カタログの読み取りと更新を実行します。
DFH0XWOD	Web サービス版の注文発送プログラム。このプログラムは EXEC CICS INVOKE WEBSERVICE を発行して、注文発送プログラムへのアウトバウンド Web サービス呼び出しを行います。

表 18. 基本アプリケーションの COBOL コピーブックが含まれる SDFHSAMP メンバー

メンバー名	説明
DFH0XCP1	Inquire Catalog、Inquire Single、および Place Order の機能の要求と応答が含まれる COMMAREA 構造を定義します。プログラム DFH0XCMN、DFH0XCUI、DFH0XECC、DFH0XGUI、DFH0XICW、DFH0XISW、DFH0XPOW、DFH0XSDS、および DFH0XVDS には、このコピーブックが組み込まれています。
DFH0XCP2	注文発送モジュールと在庫マネージャー・モジュールの COMMAREA 構造を定義します。プログラム DFH0XCMN、DFH0XSOD、DFH0XSSM、および DFH0XWOD には、このコピーブックが組み込まれています。
DFH0XCP3	Inquire Catalog の要求と応答のデータ構造を定義します。inquireCatalog.wsd1 および inquireCatalog.wsbinding を生成するために、DFHLS2WS への入力として使用されます。
DFH0XCP4	Inquire Single の要求と応答のデータ構造を定義します。inquireSingle.wsd1 および inquireSingle.wsbinding を生成するために、DFHLS2WS への入力として使用されます。
DFH0XCP5	Place Order の要求と応答のデータ構造を定義します。placeOrder.wsd1 および placeOrder.wsbinding を生成するために、DFHLS2WS への入力として使用されます。
DFH0XCP6	Dispatch Order の要求と応答のデータ構造を定義します。dispatchOrder.wsd1 および dispatchOrder.wsbinding を生成するために、DFHLS2WS への入力として使用されます。
DFH0XCP7	Dispatch Order 要求のデータ構造を定義します。プログラム DFH0XODE および DFH0XWOD には、このコピーブックが組み込まれています。
DFH0XCP8	Dispatch Order 応答のデータ構造を定義します。プログラム DFH0XODE および DFH0XWOD には、このコピーブックが組み込まれています。

表 19. CICS で稼働する Web サービス・クライアント・アプリケーションの COBOL ソース・コードを含む SDFHSAMP メンバー

メンバー名	説明
DFH0XCUI	端末ユーザーへの BMS マップの送信および端末ユーザーからのマップの受信を管理するために、トランザクション ECLI によって起動されるプログラム。これはプログラム DFH0XECC にリンクします。
DFH0XECC	EXEC CICS INVOKE WEBSERVICE コマンドを使用して、ベース・アプリケーションにアウトバウンド Web サービス要求を行います。指定される WEBSERVICE は、以下のいずれかです。 inquireCatalogClient inquireSingleClient placeOrderClient

表 20. CICS で稼働する Web サービス・クライアント・アプリケーションの COBOL コピーブックを含む SDFHSAMP メンバー：これらのメンバーはすべて DFHWS2LS によって生成され、プログラム DFH0XECC によって組み込まれます。

メンバー名	説明
DFH0XCPA	Inquire Catalog 要求のデータ構造を定義します。
DFH0XCPB	Inquire Catalog 応答のデータ構造を定義します。
DFH0XCPC	Inquire Single 要求のデータ構造を定義します。
DFH0XCPD	Inquire Single 応答のデータ構造を定義します。
DFH0XCPE	Place Order 要求のデータ構造を定義します。
DFH0XCPF	Place Order 応答のデータ構造を定義します。

表 21. ラッパー・モジュールの COBOL ソース・コードが含まれる SDFHSAMP メンバー

メンバー名	説明
DFH0XICW	inquireCatalog サービスのラッパー・プログラム。
DFH0XISW	inquireSingle サービスのラッパー・プログラム。
DFH0XPOW	purchaseOrder サービスのラッパー・プログラム。

表 22. ラッパー・モジュールの COBOL コピーブックが含まれる SDFHSAMP メンバー

メンバー名	説明
DFH0XWC1	Inquire Catalog 要求のデータ構造を定義します。プログラム DFH0XICW には、このコピーブックが組み込まれています。
DFH0XWC2	Inquire Catalog 応答のデータ構造を定義します。プログラム DFH0XICW には、このコピーブックが組み込まれています。
DFH0XWC3	Inquire Single 要求のデータ構造を定義します。プログラム DFH0XISW には、このコピーブックが組み込まれています。
DFH0XWC4	Inquire Single 応答のデータ構造を定義します。プログラム DFH0XISW には、このコピーブックが組み込まれています。
DFH0XWC5	Place Order 要求のデータ構造を定義します。プログラム DFH0XPOW には、このコピーブックが組み込まれています。
DFH0XWC6	Place Order 応答のデータ構造を定義します。プログラム DFH0XPOW には、このコピーブックが組み込まれています。

表 23. CICS リソース定義

リソース名	リソース・タイプ	コメント
EXAMPLE	CICS リソース定義グループ	実例アプリケーションに必要な CICS リソース定義
EGUI	TRANSACTION	プログラム DFH0XGUI を呼び出して、アプリケーションに対する BMS インターフェースを開始するためのトランザクション (カスタマイズ可能)
ECFG	TRANSACTION	プログラム DFH0XCFG を呼び出して実例構成 BMS インターフェースを開始するためのトランザクション (カスタマイズ可能)
EXMPCAT	FILE	アプリケーション・カタログの EXMPCAT VSAM ファイルのファイル定義 (カスタマイズ可能)
EXMPCONF	FILE	EXMPCONF アプリケーション構成ファイルのファイル定義。

カタログ・マネージャー・プログラム

カタログ・マネージャーは、実例アプリケーションのビジネス・ロジックの制御プログラムであり、実例アプリケーションとのすべての対話は、カタログ・マネージャーをパススルーします。

プログラム・ロジックが単純であることを確認するため、カタログ・マネージャーが制限したのは、型検査とエラー・リカバリーのみでした。

カタログ・マネージャーは、いくつかの操作をサポートしています。各操作の入出力パラメーターは、COMMAREA 内のプログラムとの間で受け渡される単一のデータ構造に定義されます。

COMMAREA 構造

```
*   カatalog COMMAREA 構造
      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).
*   Inquire Catalog (発注) で使用されているフィールド
      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).
* Inquire Single (1 件の問い合わせ) で使用されているフィールド
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).
* Place Order (発注) で使用されているフィールド
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).

* 発送プログラム/在庫マネージャーの COMMAREA 構造
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).
* 発送プログラムで使用されているフィールド
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).
* 在庫マネージャーで使用されているフィールド
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).

```

戻りコード

カタログ・マネージャーの各操作では、いくつかの戻りコードが戻る場合があります。

タイプ	コード	説明
一般	00	機能はエラーが発生することなく実行されました。
カタログ・ファイル	20	品目の参照番号が見つかりませんでした。
	21	カタログ・ファイルの参照のオープン、読み取り、または終了時のエラーです。
	22	ファイルの更新エラーです。

タイプ	コード	説明
構成ファイル	50	構成ファイルのオープン時エラーです。
	51	データ・ストア・タイプが STUB と VSAM のいずれでもありませんでした。
	52	アウトバウンド Web サービスの切り替えが Y と N のいずれでもありませんでした。
リモート Web サービス	30	EXEC CICS INVOKE WEBSERVICE コマンドが INVREQ 状態を戻しました。
	31	EXEC CICS INVOKE WEBSERVICE コマンドが NOTFND 状態を戻しました。
	32	EXEC CICS INVOKE WEBSERVICE コマンドが INVREQ または NOTFND 以外の状態を戻しました。
アプリケーション	97	注文を完了するには在庫が不足しています。
	98	注文数量が正数ではありませんでした。
	99	DFH0XCMN が、CA-REQUEST-ID フィールドが 01INQC、01INQS、または 01ORDR のいずれかに設定されていない COMMAREA を受け取りました。

INQUIRE CATALOG 操作

この操作を実行すると、呼び出し元が指定した品目を先頭に、最大 15 品目のカタログ品目リストが戻されます。

入力パラメーター

CA-REQUEST-ID

操作を指定するストリング。INQUIRE CATALOG コマンドの場合、このストリングには「01INQC」が含まれます。

CA-LIST-START-REF

戻される最初の品目の参照番号。

出力パラメーター

CA-RETURN-CODE

CA-RESPONSE-MESSAGE

「*num* ITEMS RETURNED」を含む、人が読めるストリング。ここで、*num* は、戻される品目の数を表します。

CA-LAST-ITEM-REF

戻された最後の品目の参照番号。

CA-ITEM-COUNT

戻された品目の数。

CA-CAT-ITEM

戻されたカタログ品目のリストを含む配列。この配列のエLEMENTの数は 15 です。戻された品目数が 15 未満の場合、残りの配列ELEMENTには空白が入ります。

INQUIRE SINGLE ITEM 操作

この操作を実行すると、呼び出し元が指定した単一のカタログ品目が戻ります。

入力パラメーター**CA-REQUEST-ID**

操作を指定するストリング。INQUIRE SINGLE ITEM コマンドの場合、このストリングには「01INQS」が含まれます。

CA-ITEM-REF-REQ

戻される品目の参照番号。

出力パラメーター**CA-RETURN-CODE****CA-RESPONSE-MESSAGE**

「RETURNED ITEM: REF=*item-reference*」が含まれる、人が読めるストリング。ここで、*item-reference* は、戻された品目の参照番号を表します。

CA-SINGLE-ITEM

戻されたカタログ項目がその最初のエLEMENTに含まれている配列。

PLACE ORDER 操作

この操作を実行すると、単一品目が発注されます。必要な数量が入力されていないと、ユーザーにメッセージが戻されます。注文が正常に実行されると、在庫マネージャーが呼び出され、注文された品目とその数量が通知されます。

入力パラメーター**CA-REQUEST-ID**

操作を指定するストリング。PLACE ORDER 操作の場合、このストリングには「01ORDR」が入ります。

CA-USERID

発送および請求のためにアプリケーションが使用する 8 文字のユーザー ID。

CA-CHARGE-DEPT

発送および請求のためにアプリケーションが使用する 8 文字の部門 ID。

CA-ITEM-REF-NUMBER

注文された品目の参照番号。

CA-QUANTITY-REQ

品目の必要数。

出力パラメーター**CA-RETURN-CODE****CA-RESPONSE-MESSAGE**

「ORDER SUCCESSFULLY PLACED」を含む、人が読めるストリング。

DISPATCH STOCK 操作

この操作を実行すると、在庫発送プログラムが呼び出され、このプログラムによって注文品が顧客に順に発送されます。

入力パラメーター**CA-ORD-REQUEST-ID**

操作を指定するストリング。 DISPATCH ORDER 操作の場合、このストリングには「01DSP0」が入ります。

CA-ORD-USERID

発送および請求のためにアプリケーションが使用する 8 文字のユーザー ID。

CA-ORD-CHARGE-DEPT

発送および請求のためにアプリケーションが使用する 8 文字の部門 ID。

CA-ORD-ITEM-REF-NUMBER

注文された品目の参照番号。

CA-ORD-QUANTITY-REQ

品目の必要数。

出力パラメーター**CA-ORD-RETURN-CODE****NOTIFY STOCK MANAGER 操作**

この操作では、在庫の補充が必要かどうかを判断するために、出された注文の詳細を取り込みます。

入力パラメーター**CA-ORD-REQUEST-ID**

操作を指定するストリング。 NOTIFY STOCK MANAGER 操作の場合、このストリングには「01STK0」が入ります。

CA-STK-ITEM-REF-NUMBER

注文された品目の参照番号。

CA-STK-QUANTITY-REQ

品目の必要数。

出力パラメーター**CA-ORD-RETURN-CODE**

ファイル構造と定義

実例アプリケーションでは、2 つの VSAM ファイルが使用されます。1 つは、在庫になっているすべての品目の詳細とその在庫レベルが記録されているカタログ・ファイルで、もう 1 つは、アプリケーションのユーザー選択オプションが保持されている構成ファイルです。

カタログ・ファイル

カタログ・ファイルとは、製品の在庫に関連するすべての情報が格納されている KSDS VSAM ファイルのことです。

このファイルのレコードは、以下の構造になっています。

名前	COBOL データ・タイプ	説明
WS-ITEM-REF-NUM	PIC 9(4)	品目の参照番号
WS-DESCRIPTION	PIC X(40)	品目の説明
WS-DEPARTMENT	PIC 9(3)	部門識別番号
WS-COST	PIC ZZZ.99	品目の価格
WS-IN-STOCK	PIC 9(4)	品目の在庫数
WS-ON-ORDER	PIC 9(3)	品目の注文数

構成ファイル

構成ファイルとは、実例アプリケーションの構成に使用される情報が格納されている KSDS VSAM ファイルのことです。

構成ファイルは、4 つの異なるレコードを持つ KSDS VSAM ファイルです。

表 24. 一般情報レコード

名前	COBOL データ・タイプ	説明
PROGS-KEY	PIC X(9)	「EXMP-CONF」を含む一般情報レコードのキー・フィールド
充てん文字	PIC X	
DATASTORE	PIC X(4)	使用するデータ・ストア・プログラムのタイプを指定する文字ストリング。値は以下のとおりです。 「STUB」 「VSAM」
充てん文字	PIC X	

表 24. 一般情報レコード (続き)

名前	COBOL データ・タイプ	説明
DO-OUTBOUND-WS	PIC X	発送マネージャーがアウトバウンドの Web サービス要求を行うかどうかを指定する文字。値は以下のとおりです。 「Y」 「N」
充てん文字	PIC X	
CATMAN-PROG	PIC X(8)	カタログ・マネージャー・プログラムの名前
充てん文字	PIC X	
DSSTUB-PROG	PIC X(8)	ダミーのデータ・ハンドラー・プログラムの名前
充てん文字	PIC X	
DSVSAM-PROG	PIC X(8)	VSAM データ・ハンドラー・プログラムの名前
充てん文字	PIC X	
ODSTUB-PROG	PIC X(8)	ダミーの注文発送モジュールの名前
充てん文字	PIC X	
ODWEBS-PROG	PIC X(8)	アウトバウンドの Web サービス注文発送プログラムの名前
充てん文字	PIC X	
STKMAN-PROG	PIC X(8)	在庫マネージャー・プログラムの名前
充てん文字	PIC X(10)	

表 25. アウトバウンド URL レコード

名前	COBOL データ・タイプ	説明
URL-KEY	PIC X(9)	「OUTBNDURL」を含む一般情報レコードのキー・フィールド
充てん文字	PIC X	
OUTBOUND-URL	PIC X(255)	注文発送 Web サービス要求のアウトバウンド URL

表 26. カatalog・ファイル情報レコード

名前	COBOL データ・タイプ	説明
URL-FILE-KEY	PIC X(9)	「VSAM-NAME」を含む一般情報レコードのキー・フィールド
充てん文字	PIC X	

表 26. カタログ・ファイル情報レコード (続き)

名前	COBOL データ・タイプ	説明
CATALOG-FILE-NAME	PIC X(8)	カタログ・ファイルに使用された CICS FILE リソースの名前

表 27. サーバー情報レコード

名前	COBOL データ・タイプ	説明
WS-SERVER-KEY	PIC X(9)	「WS-SERVER」を含むサーバー情報レコードのキー・フィールド
充てん文字	PIC X	
CATALOG-FILE-NAME	PIC X(8)	CICS Web サービス・クライアントの場合に限り、実例アプリケーションが Web サービスとして配置されているシステムの IP アドレスとポート

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態を提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

本書には、技術的に正確でない記述や誤植があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。 IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN 本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

商標

IBM、IBM ロゴ、および `ibm.com` は、International Business Machines Corporation の米国およびその他の国における商標または登録商標です。これらおよび他の IBM 商標に、この情報の最初に現れる個所で商標表示 (® または TM) が付されている場合、これらの表示は、この情報が公開された時点で、米国において、IBM が所有する登録商標またはコモン・ロー上の商標であることを示しています。このような商標は、その他の国においても登録商標またはコモン・ロー上の商標である可能性があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml の「Copyright and trademark information」をご覧ください。

Java およびすべての Java 関連の商標およびロゴは Sun Microsystems, Inc. の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

参考文献

CICS Transaction Server for z/OS の CICS ブック

一般

CICS Transaction Server for z/OS Program Directory, GI13-0536
CICS Transaction Server for z/OS リリース・ガイド, GC88-5845
CICS Transaction Server for z/OS V2.3 からのアップグレード, GC88-5847
CICS Transaction Server for z/OS V3.1 からのアップグレード, GC88-5848
CICS Transaction Server for z/OS V3.2 からのアップグレード, GC88-5849
CICS Transaction Server for z/OS インストール・ガイド, GC88-5846

CICS へのアクセス

CICS インターネット・ガイド, SC88-5853
CICS Web サービス・ガイド, SC88-5852

管理

CICS System Definition Guide, SC34-6999
CICS Customization Guide, SC34-7001
CICS Resource Definition Guide, SC34-7000
CICS Operations and Utilities Guide, SC34-7002
CICS RACF Security Guide, SC34-7003
CICS Supplied Transactions, SC34-7004

プログラミング

CICS アプリケーション・プログラミング・ガイド, SC88-5850
CICS アプリケーション・プログラミング・リファレンス, SC88-5851
CICS System Programming Reference, SC34-7024
CICS Front End Programming Interface User's Guide, SC34-7027
CICS C++ OO Class Libraries, SC34-7026
CICS Distributed Transaction Programming Guide, SC34-7028
CICS Business Transaction Services, SC34-7029
Java Applications in CICS, SC34-7025

診断

CICS Problem Determination Guide, SC34-7034
CICS パフォーマンス・ガイド, SC88-5854
CICS Messages and Codes, SC34-7035
CICS Diagnosis Reference, GC34-7038
CICS Recovery and Restart Guide, SC34-7012
CICS Data Areas, GC34-7014
CICS Trace Entries, SC34-7013
CICS Supplementary Data Areas, GC34-7015
CICS Debugging Tools Interfaces Reference, GC34-7039

通信

CICS 相互通信ガイド、SC88-5857

CICS External Interfaces Guide、SC34-7019

データベース

CICS DB2 Guide、SC34-7011

CICS IMS Database Control Guide、SC34-7016

CICS Shared Data Tables Guide、SC34-7017

CICS Transaction Server for z/OS の CICSplex SM ブック

一般

CICSplex SM 概念および計画、SC88-5855

CICSplex SM WUI ガイド、SC88-5856

管理

CICSplex SM Administration、SC34-7005

CICSplex SM Operations Views Reference、SC34-7006

CICSplex SM Monitor Views Reference、SC34-7007

CICSplex SM Managing Workloads、SC34-7008

CICSplex SM Managing Resource Usage、SC34-7009

CICSplex SM Managing Business Applications、SC34-7010

プログラミング

CICSplex SM Application Programming Guide、SC34-7030

CICSplex SM Application Programming Reference、SC34-7031

診断

CICSplex SM Resource Tables Reference、SC34-7032

CICSplex SM Messages and Codes、GC34-7035

CICSplex SM Problem Determination、GC34-7037

他の CICS 資料

以下の資料には CICS に関する詳しい情報が含まれますが、これらの資料は CICS Transaction Server for z/OS バージョン 4 リリース 1 の一部としては提供されません。

Designing and Programming CICS Applications、SR23-9692

CICS Application Migration Aid Guide、SC33-0768

CICS ファミリー: API の構成、SC88-7261

CICS ファミリー クライアント・サーバー プログラミングの手引き、SC88-7429

CICS Family: Interproduct Communication、SC34-6853

CICS Family: Communicating from CICS on System/390、SC34-6854

CICS Transaction Gateway (OS/390 版) 管理の手引き、SD88-7246

CICS Family: General Information、GC33-0155

CICS 4.1 Sample Applications Guide、SC33-1173

CICS/ESA 3.3 XRF Guide、SC33-0661

アクセシビリティ

アクセシビリティ機能は、運動障害または視覚障害など身体に障害を持つユーザーがソフトウェア・プロダクトを快適に使用できるようにサポートします。

CICS システムのセットアップ、実行、および保守に必要なほとんどの作業は、以下のいずれかの方法で行うことができます。

- CICS にログオンした 3270 エミュレーターを使用する
- TSO にログオンした 3270 エミュレーターを使用する
- 3270 エミュレーターを MVS™ システム・コンソールとして使用する

IBM パーソナル・コミュニケーションズは、身体障害のある方々のためのアクセシビリティ機能を持つ 3270 エミュレーションを提供します。CICS システムで必要なアクセシビリティ機能を提供するためにこの製品を使用することができます。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

アシスタント、Web サービス 152
アトミック・トランザクション 265, 273
 サービス・プロバイダーの構成 270
 サービス・リクエスターの構成 271
状態 274
登録サービス 265
CICS の構成 268
アルゴリズム 319, 321
永続メッセージ 63
永続メッセージのサポート 64
エンドポイント参照 (EPR) 288
エンベロープ、SOAP 13

[カ行]

カスタムのセキュリティー・ハンドラー 328
カタログ・サンプルカタログ・サンプル 347
起動、Trust クライアント 329
繰り返しコンテンツ 228
言語構造
 WSDL への変換 153
高水準言語構造
 WSDL への変換 153
構成、パイプラインの 324
構成ファイル、パイプライン 70
構文表記 181
互換モード 279
コンテキスト・コンテナー 135
 DFHWS-CID-DOMAIN 142
 DFHWS-IDTOKEN 146
 DFHWS-MEP 136
 DFHWS-MTOM-IN 143
 DFHWS-MTOM-OUT 143
 DFHWS-RESPWAIT 138
 DFHWS-RESTOKEN 146
 DFHWS-SERVICEURI 147
 DFHWS-STSACTION 147
 DFHWS-STSFault 147
 DFHWS-STSURI 148
 DFHWS-TOKENTYPE 148
 DFHWS-WSDL-CTX 145

コンテキスト・コンテナー (続き)

DFHWS-XOP-IN 145
DFHWS-XOP-OUT 145

コンテナー

コンテキスト・コンテナー

DFHWS-APPHANDLER 136, 137
DFHWS-DATA 136
DFHWS-PIPELINE 137
DFHWS-SOAPLEVEL 138
DFHWS-STReason 147
DFHWS-TRANID 138
DFHWS-URI 138
DFHWS-USERID 142
DFHWS-WEBSERVICE 142
DFH-HANDLERPLIST 135
DFH-SERVICEPLIST 136

制御コンテナー

DFHERROR 126
DFHFUNCTION 128
DFHHTTPSTATUS 131
DFHMEDIATYPE 131
DFHNORESPONSE 131
DFHREQUEST 132
DFHRESPONSE 132

チャネル記述 237

パイプラインで使用する 125

DFHERROR 126
DFHFUNCTION 128
DFHHTTPSTATUS 131
DFHMEDIATYPE 131
DFHNORESPONSE 131
DFHREQUEST 132
DFHRESPONSE 132
DFHWS-APPHANDLER 136, 137
DFHWS-CID-DOMAIN 142
DFHWS-DATA 136
DFHWS-IDTOKEN 146
DFHWS-MEP 136
DFHWS-MTOM-IN 143
DFHWS-MTOM-OUT 143
DFHWS-PIPELINE 137
DFHWS-RESPWAIT 138
DFHWS-RESTOKEN 146
DFHWS-SERVICEURI 147
DFHWS-SOAPLEVEL 138
DFHWS-STSACTION 147
DFHWS-STSFault 147
DFHWS-STReason 147
DFHWS-STSURI 148
DFHWS-TOKENTYPE 148
DFHWS-TRANID 138

コンテナー (続き)

DFHWS-URI 138
DFHWS-USERID 142
DFHWS-WEBSERVICE 142
DFHWS-XOP-IN 145
DFHWS-XOP-OUT 145
DFH-HANDLERPLIST 135
DFH-SERVICEPLIST 136

コンテナー DFHWS-CID-DOMAIN 142

コンテナー DFHWS-IDTOKEN 146

コンテナー DFHWS-MEP 136

コンテナー DFHWS-MTOM-IN 143

コンテナー DFHWS-MTOM-OUT 143

コンテナー DFHWS-RESPWAIT 138

コンテナー DFHWS-RESTOKEN 146

コンテナー DFHWS-SERVICEURI 147

コンテナー DFHWS-STSACTION 147

コンテナー DFHWS-STSFault 147

コンテナー DFHWS-STSURI 148

コンテナー DFHWS-TOKENTYPE 148

コンテナー DFHWS-WSDL-CTX 145

コンテナー DFHWS-XOP-IN 145

コンテナー DFHWS-XOP-OUT 145

[サ行]

サービス

 パイプライン構成エレメント 89

サービス・パラメーター・リスト

 <service_parameter_list> 91

サービス・プロバイダー

 問題の診断 338

サービス・プロバイダー・アプリケーション

 アトミック・トランザクションの使用 270

 作成、データ構造を基にして 234

サービス・リクエスター

 パイプライン定義 77

 問題の診断 339

サービス・リクエスター・アプリケーション

 アトミック・トランザクションの使用 271

実行時の制限 255

障害、SOAP 13

商標 394

図

 構文 181

スキーマ

 チャネル記述 237

制御コンテナ 126
制限、実行時の 255
セキュリティ・コンテナ 146
セキュリティ・ハンドラー
作成、独自の 328

[タ行]

端末以外のメッセージ・ハンドラー 114,
115, 116
チャンネル記述 237
チャンネル・インターフェース 237
直接モード 279
デフォルト EPR 295
WSDL 1.1 295
動的ルーティング
サービス・プロバイダーの場合 124
端末ハンドラーでの 124

[ハ行]

バイナリー添付ファイル
パイプライン構成 93
パイプライン構成
MTOM/XOP 93
Web Services Security 98
パイプライン構成エレメント
<apphandler> 80
<authentication> 100
<auth_token_type> 106
<cics_mtom_handler> 93
<cics_soap_1.1_handler> 82
<cics_soap_1.2_handler> 85
<default_http_transport_
handler_list> 87
<default_mq_transport_handler_list> 88
<default_transport_handler_list> 88
<dfhmtom_configuration> 94
<dfhwsse_configuration> 98
<encrypt_body> 108
<handler> 89
<mime_options> 97
<mtom_options> 94
<named_transport_entry> 79
<provider_pipeline> 80
<requester_pipeline> 82
<service> 89
<service_handler_list> 90
<sign_body> 107
<sts_authentication> 104
<sts_endpoint> 107
<terminal_handler> 80
<transport> 92
<transport_handler_list> 81
<wsse_handler> 98

パイプライン構成エレメント (続き)
<xop_options> 96
パイプライン構成ファイル 70
パイプライン処理
URI の指定変更 262
パイプライン定義
サービス・リクエスター 77
バッチ・ユーティリティ
Web サービス・アシスタント 152
ハンドラー
パイプライン構成エレメント 89
表記
構文 181
ヘッダー、SOAP 13
変数配列 214
本体、SOAP 13

[マ行]

メッセージ交換パターン (MEP) 32
メッセージ・ハンドラー
起動、Trust クライアント 329
端末以外の 114, 115, 116
問題の診断
サービス・プロバイダー 338
サービス・リクエスター 339

[ヤ行]

ユーザー・コンテナ 151
ユーティリティ・プログラム
Web サービス・アシスタント 152

[ワ行]

ワークロード管理
サービス・プロバイダーの場合 124
端末ハンドラーでの 124

A

apphandler
パイプライン構成エレメント 80
authentication
パイプライン構成エレメント 100
auth_token_type
パイプライン構成エレメント 106

C

C および C++
XML スキーマへのマッピング 198,
200
C および C++ へのマッピング 198, 200

cics_mtom_handler
パイプライン構成エレメント 93
cics_soap_1.1_handler
パイプライン構成エレメント 82
cics_soap_1.2_handler
パイプライン構成エレメント 85
COBOL
可変の繰り返しコンテンツ 228
XML スキーマへのマッピング 189,
193
COBOL へのマッピング 189, 193

D

default_http_transport_handler_list
パイプライン構成エレメント 87
default_mq_transport_handler_list
パイプライン構成エレメント 88
default_target
パイプライン構成エレメント 92
default_transport_handler_list
パイプライン構成エレメント 88
DFHERROR コンテナ 126
DFHFHFUNCTION コンテナ 128
DFHHTTPSTATUS コンテナ 131
DFHLS2WS
カタログ式プロシージャー 153
DFHMEDIATYPE コンテナ 131
dfhmtom_configuration
パイプライン構成エレメント 94
DFHNORESPONSE コンテナ 131
DFHREQUEST コンテナ 132
DFHRESPONSE コンテナ 132
DFHWS2LS
カタログ式プロシージャー 166
dfhwsse_configuration
パイプライン構成エレメント 98
DFHWS-APPHANDLER コンテナ 136,
137
DFHWS-DATA コンテナ 136
DFHWS-PIPELINE コンテナ 137
DFHWS-SOAPLEVEL コンテナ 138
DFHWS-STSREASON コンテナ 147
DFHWS-TRANID コンテナ 138
DFHWS-URI コンテナ 138
DFHWS-USERID コンテナ 142
DFHWS-WEBSERVICE コンテナ 142
DFH-HANDLERPLIST コンテナ 135
DFH-SERVICEPLIST コンテナ 136

E

encrypt_body
パイプライン構成エレメント 108

EXEC CICS SOAPFAULT CREATE コマンド 240

M

maxOccurs
XML スキーマ内で 214
MEP 32
MIME メッセージ
パイプライン構成 93
mime_options
パイプライン構成エレメント 97
minOccurs
XML スキーマ内で 214
MTOM/XOP
パイプライン構成 93
mtom_options
パイプライン構成エレメント 94

N

named_transport_entry
パイプライン構成エレメント 79

P

PL/I
XML スキーマへのマッピング 204, 208
PL/I へのマッピング 204, 208
provider_pipeline
パイプライン構成エレメント 80

R

RACF の構成 322
requester_pipeline
エレメント、パイプライン定義の 77
パイプライン構成エレメント 82

S

Security Token Service
Trust クライアント・インターフェース 317
service_handler_list
パイプライン構成エレメント 90
service_parameter_list
サービス・パラメーター・リスト 91
sign_body
パイプライン構成エレメント 107
SOAP
エンベロープ 13
概要 13

SOAP (続き)
障害 13
ヘッダー 13
本体 13
SOAP の概要 13
SOAP Message Security 38
SOAP 障害 240
SOAP メッセージ
暗号化 320
構造 13
署名 318
例 13
XML スキーマ
SOAP メッセージの検証 250
XML スキーマとの照合 250
SOAP メッセージの検証 250
SOAP メッセージ・パス 19
sts_authentication
パイプライン構成エレメント 104
sts_endpoint
パイプライン構成エレメント 107

T

terminal_handler
パイプライン構成エレメント 80
transport
パイプライン構成エレメント 92
transport_handler_list
パイプライン構成エレメント 81
Trust クライアント
インターフェース 317
起動 329

U

URI
MQ トランスポートの 62
URI の指定変更 262

W

Web Services Addressing
アドレッシング・ハンドラー 291
サポート 287
仕様 287
デフォルト EPR 293, 295
デフォルトのアクション 293, 295, 296, 298
パイプライン構成 291
パラメーター 293
明示的アクション 293, 295
DFHWSADH 291
DFHWS-URI 288
EPR 288

Web Services Addressing (続き)
MAP 288
WSADDR-EPR-ANY 293
WSDL 1.1 296
WSDL 2.0 298
WSDL の構成 293
<wsa:Action> 295
Web Services Security
パイプライン構成 98
Web Services Security (WSS) 311, 322, 324
Web Services Security: SOAP Message Security 38
Web サービスのエラー 338, 339
Web サービスのためのセキュリティ 311
Web サービス・アシスタント 152
サービス・プロバイダー・アプリケーションの作成 234
WSDL
およびアプリケーション・データ構造 30
言語構造への変換 166
WSDL 1.1
デフォルト EPR 295
WSDL 仕様 37
WSDL の構成
Web Services Addressing 293
WS-Addressing 293
wsse_handler
パイプライン構成エレメント 98
WSS: SOAP Message Security 38
WS-Addressing
アドレッシング・ハンドラー 291
仕様 287
デフォルト EPR 293, 295
デフォルトのアクション 293, 295, 296, 298
パイプライン構成 291
パラメーター 293
明示的アクション 293, 295
DFHWSADH 291
DFHWS-URI 288
EPR 288
MAP 288
WSADDR-EPR-ANY 293
WSDL 1.1 296
WSDL 2.0 298
WSDL の構成 293
<wsa:Action> 295
WS-AT 265

X

XML スキーマ 189, 193, 198, 200, 204, 208

xop_options
 パイプライン構成エレメント 96

[特殊文字]

<apphandler>
 パイプライン構成エレメント 80

<authentication>
 パイプライン構成エレメント 100

<auth_token_type>
 パイプライン構成エレメント 106

<cics_mtom_handler>
 パイプライン構成エレメント 93

<cics_soap_1.1_handler>
 パイプライン構成エレメント 82

<cics_soap_1.2_handler>
 パイプライン構成エレメント 85

<default_http_transport_handler_list>
 パイプライン構成エレメント 87

<default_mq_transport_handler_list>
 パイプライン構成エレメント 88

<default_target>
 パイプライン構成エレメント 92

<default_transport_handler_list>
 パイプライン構成エレメント 88

<dfhmtom_configuration>
 パイプライン構成エレメント 94

<dfhwsse_configuration>
 パイプライン構成エレメント 98

<encrypt_body>
 パイプライン構成エレメント 108

<handler>
 パイプライン構成エレメント 89

<mime_options>
 パイプライン構成エレメント 97

<mtom_options>
 パイプライン構成エレメント 94

<named_transport_entry>
 パイプライン構成エレメント 79

<provider_pipeline>
 パイプライン構成エレメント 80

<requester_pipeline>
 パイプライン構成エレメント 82

<service>
 パイプライン構成エレメント 89

<service_handler_list>
 パイプライン構成エレメント 90

<service_parameter_list>
 パイプライン構成エレメント 91

<sign_body>
 パイプライン構成エレメント 107

<sts_authentication>
 パイプライン構成エレメント 104

<sts_endpoint>
 パイプライン構成エレメント 107

<terminal_handler>
 パイプライン構成エレメント 80

<transport>
 パイプライン構成エレメント 92

<transport_handler_list>
 パイプライン構成エレメント 81

<wsse_handler>
 パイプライン構成エレメント 98

<xop_options>
 パイプライン構成エレメント 96



SC88-5852-00



日本アイ・ビー・エム株式会社
〒106-8711 東京都港区六本木3-2-12