

z/OS용 CICS Transaction Server
버전 5 릴리스 3



웹 서비스 안내서

z/OS용 CICS Transaction Server
버전 5 릴리스 3



웹 서비스 안내서

참고

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 먼저 765 페이지의 『주의사항』의 정보를 읽으십시오.

이 개정판은 새 개정판에 별도로 명시하지 않는 한, z/OS용 IBM CICS Transaction Server 버전 5 릴리스 3(제품 번호 5655-Y04) 및 모든 후속 릴리스와 수정에 적용됩니다.

© Copyright IBM Corporation 1999, 2015.

목차

서문.	vii
이 서적에서 제공하는 정보	vii
이 책의 독자.	vii
Information Center에서 주제의 위치.	vii

z/OS용 CICS Transaction Server, 버전 5 릴리스	
3 의 변경사항.	ix

제 1 장 웹 서비스	1
웹 서비스의 개념	2
비즈니스에 웹 서비스를 활용하는 방법.	4
웹 서비스 용어	4

제 2 장 SOAP 웹 서비스 아키텍처	9
웹 서비스 설명	10
서비스 관련 서적	12

제 3 장 SOAP	15
SOAP 메시지 구조.	15
SOAP 헤더	17
SOAP 본문	19
SOAP 결함	19
SOAP 노트	22
SOAP 메시지 경로.	23

제 4 장 CICS가 SOAP 웹 서비스를 지원하는 방 법	25
메시지 핸들러 및 파이프라인	26
전송 관련 핸들러	27
플로우 인터럽트.	28
서비스 제공자 파이프라인.	28
서비스 요청자 파이프라인.	29
CICS 파이프라인 및 SOAP.	30
SOAP 메시지 및 애플리케이션 데이터 구조	31
WSDL 및 애플리케이션 데이터 구조.	34
WSDL 및 메시지 교환 패턴	36
웹 서비스 바인딩 파일.	38
외부 표준	38
SAML	39
SOAP 1.1 및 1.2	39
MTOM 1.0에 대한 SOAP 1.1 바인딩	40
SOAP 메시지 전송 최적화 메커니즘(MTOM).	40
Web Services Addressing 1.0.	40

Web Services Atomic Transaction 버전 1.0	41
Web Services Coordination 버전 1.0	41
Web Services Description Language 버전 1.1 및 2.0	42
웹 서비스 보안: SOAP 메시지 보안	42
웹 서비스 신뢰 언어	43
SOAP 1.2에 대한 WSDL 1.1 바인딩 확장	43
WS-I 기본 프로파일 버전 1.1	43
WS-I 단순 SOAP 바인딩 프로파일 버전 1.0	44
XML(Extensible Markup Language) 버전 1.0	44
XML-2진 최적화 패키징(XOP).	44
XML 암호화 구문 및 처리	45
XML-서명 구문 및 처리	45
CICS의 웹 서비스 표준 준수	45

제 5 장 SOAP 웹 서비스 시작하기	53
SOAP 웹 서비스 사용 계획.	53
SOAP 서비스 제공자 애플리케이션 계획	55
SOAP 서비스 요청자 애플리케이션 계획	57

제 6 장 웹 서비스 인프라 작성	61
웹 서비스를 위한 CICS 시스템 구성.	61
SOAP 웹 서비스 작성	61
JSON 웹 서비스 작성	179
웹 서비스에 대한 CICS 자원	231
WebSphere MQ 전송을 사용하도록 CICS 구성	235
웹 서비스 지원과 WSRR 간의 상호 운용성	244
웹 서비스 인프라	246
서비스 제공자로서 CICS	246
서비스 요청자로서 CICS	247
Java 기반 SOAP 파이프라인	249
웹 서비스에 대한 데이터 형성.	251
JSON 요청에 대한 서비스 제공자로서의 CICS	252
SOAP 서비스 제공자를 위한 CICS 인프라 작성	257
SOAP 서비스 요청자를 위한 CICS 인프라 작성	259
JSON 서비스 제공자를 위한 CICS 인프라 작성	261
CICS용 z/OS 연결 구성	263
CICS JSON 웹 서비스에 대한 CICS용 z/OS 연결 구성	265
Java 파이프라인에서 CICS용 z/OS 연결로 JSON 웹 서비스 이동	267
파이프라인 구성 파일.	269
전송 관련 핸들러	275

서비스 제공자에 대한 파이프라인 정의	278
서비스 요청자에 대한 파이프라인 정의	279
서비스 제공자에서만 사용되는 요소	280
서비스 요청자에서 사용되는 요소	286
서비스 제공자 및 서비스 요청자 파이프라인에서 사용되는 요소	287
MTOM/XOP을 위한 파이프라인 구성	306
WS-Security를 위한 파이프라인 구성	313
애플리케이션 핸들러	327
채널 첨부 애플리케이션 핸들러	328
메시지 핸들러	329
메시지 핸들러 프로토콜	330
자체 메시지 핸들러 제공	333
비터미널 메시지 핸들러에서 메시지에 대한 작업	334
터미널 메시지 핸들러에서 메시지에 대한 작업	337
오류 핸들링	338
메시지 핸들러 인터페이스	339
SOAP 메시지 핸들러	339
헤더 처리 프로그램	339
헤더 처리 프로그램 인터페이스	342
터미널 핸들러에서 인바운드 요청의 동적 라우팅	344
파이프라인에서 사용되는 컨테이너	345
제어 컨테이너	346
컨테이너의 파이프라인 프로토콜 제어 방식	353
컨텍스트 컨테이너	356
헤더 처리 프로그램 컨테이너	370
보안 컨테이너	372
SAML 지원 컨테이너	374
CICS에서 생성된 컨테이너	378
사용자 컨테이너	379
웹 서비스를 위한 런타임 처리	379
CICS가 웹 서비스 지원으로 배치되는 서비스 제 공자 프로그램을 호출하는 방법	379
웹 서비스 지원으로 배치되는 애플리케이션으로 부터 웹 서비스 호출	380
웹 서비스에 대한 로컬 최적화	381
웹 서비스 지원에 의해 생성되는 코드에 대한 런 타임 제한사항	382
파이프라인 처리 사용자 정의	385
요청자 파이프라인 처리 제어를 위한 옵션	386
URI을 사용하여 요청자 파이프라인 처리 제어	388
JSON 웹 서비스 제한사항	390
웹 서비스 트랜잭션 지원	391
등록 서비스	392
웹 서비스 트랜잭션을 위한 CICS 구성	394
웹 서비스 트랜잭션을 위한 서비스 제공자 구성	396

웹 서비스 트랜잭션을 위한 서비스 요청자 구성	397
SOAP 메시지가 원자적 트랜잭션의 파트인지 판 별	399
원자적 트랜잭션의 진행상태 확인	400
2진 데이터의 MTOM/XOP 최적화 지원	401
MTOM/XOP 및 SOAP	401
CICS의 MTOM 메시지 및 2진 첨부 파일	403
MTOM/XOP 사용 시 제한사항	407
CICS에서 MTOM/XOP를 지원하도록 구성	409
WS-Addressing에 대한 지원	411
WS-Addressing 개요	413
WS-Addressing을 위한 요청자 파이프라인 구성	416
WS-Addressing을 위한 제공자 파이프라인 구성	418
WS-Addressing을 사용하는 웹 서비스 작성	419
메시지 교환	426
WS-Addressing의 필수 메시지 주소 지정 특성	428
WS-Addressing 보안	429
WS-Addressing 예제	430
WS-Addressing 용어	435
SAML에 대한 지원	435
Kerberos에 대해 CICS 웹 서비스 구성	436
제 7 장 SOAP 웹 서비스 작성	439
CICS 웹 서비스 지원자	440
DFHLS2WS: 고급 언어 대 WSDL 변환	441
DFHWS2LS: WSDL을 고급 언어로 변환	457
구문 표기법	474
CICS 지원에 대한 맵핑 레벨	475
고급 언어 및 XML 스키마 맵핑	480
웹 서비스 지원을 사용하여 웹 서비스 제공자 작성	528
웹 서비스 설명에서 서비스 제공자 애플리케이션 작성	529
데이터 구조에서 서비스 제공자 애플리케이션 작 성	532
채널 설명 문서 작성	535
생성된 웹 서비스 설명 문서 사용자 정의	537
SOAP 결함 전송	539
웹 서비스 지원을 사용하여 웹 서비스 요청자 작성	541
도구를 사용하여 웹 서비스 작성	544
자체 XML 인식 웹 서비스 애플리케이션 작성	545
XML 인식 서비스 제공자 애플리케이션 작성	546
XML 인식 서비스 요청자 애플리케이션 작성	548
웹 서비스에 Java 사용	550
Java 제공자 모드 웹 서비스를 Axis2 JVM 서 버에 배치	550
XML을 생성하고 구문 분석하는 Java 웹 서비스 작성	552

COBOL 인터페이스가 있는 Java 웹 서비스 작성	553
요청자 모드 JAX-WS 웹 서비스 배치	553
Liberty JVM 서버에서 Java 제공자 모드 웹 서비스 배치	554
SOAP 메시지 유효성 검증	555
제 8 장 JSON 웹 서비스 작성	559
CICS JSON 지원	560
DFHLS2JS: 요청-응답 서비스를 위한 고급 언어 대 JSON 스키마 변환	560
DFHJS2LS: 요청-응답 서비스를 위한 JSON 스키마 대 고급 언어 변환	571
DFHJS2LS: RESTful 서비스를 위한 JSON 스키마 대 고급 언어 변환	584
JSON 지원을 사용하여 JSON 서비스 제공자 애플리케이션 작성	596
JSON 스키마에서 서비스 제공자 애플리케이션 작성	597
데이터 구조에서 서비스 제공자 애플리케이션 작성	598
채널 설명 문서 작성	601
생성된 JSON 스키마 사용자 정의	603
RESTful 웹 서비스 제공자 애플리케이션 작성	604
RESTful 웹 서비스 제공자 애플리케이션에 대한 디자인 고려사항	607
애플리케이션 오류 보고	608
JSON 웹 서비스 제한사항	609
제 9 장 웹 서비스를 위한 런타임 처리	611
CICS가 웹 서비스 지원으로 배치되는 서비스 제공자 프로그램을 호출하는 방법	611
웹 서비스 지원으로 배치되는 애플리케이션으로부터 웹 서비스 호출	612
웹 서비스에 대한 로컬 최적화	613
웹 서비스 지원에 의해 생성되는 코드에 대한 런타임 제한사항	613
파이프라인 처리 사용자 정의	617
요청자 파이프라인 처리 제어를 위한 옵션	618
URI를 사용하여 요청자 파이프라인 처리 제어	620
JSON 웹 서비스 제한사항	621
제 10 장 웹 서비스 트랜잭션 지원	625
등록 서비스	625
웹 서비스 트랜잭션을 위한 CICS 구성	628
웹 서비스 트랜잭션을 위한 서비스 제공자 구성	630
웹 서비스 트랜잭션을 위한 서비스 요청자 구성	631

SOAP 메시지가 원자적 트랜잭션의 पार्ट인지 판별	633
원자적 트랜잭션의 진행상태 확인	634
제 11 장 2진 데이터의 MTOM/XOP 최적화 지원	637
MTOM/XOP 및 SOAP	637
CICS의 MTOM 메시지 및 2진 첨부 파일	639
Java를 지원하지 않는 파이프라인에 대한 인바운드 MTOM 메시지 처리	640
Java를 지원하지 않는 파이프라인에 대한 아웃바운드 MTOM 메시지 처리	642
MTOM/XOP 사용 시 제한사항	643
Java 기반 파이프라인에 대한 제한사항	643
기타 SOAP 파이프라인에 대한 제한사항	644
CICS에서 MTOM/XOP를 지원하도록 구성	645
Java 기반 파이프라인에 대한 MTOM/XOP 지원 구성	645
다른 SOAP 파이프라인에 대한 MTOM/XOP 구성	646
제 12 장 WS-Addressing에 대한 지원	649
WS-Addressing 개요	650
WS-Addressing을 위한 요청자 파이프라인 구성	653
WS-Addressing을 위한 제공자 파이프라인 구성	655
WS-Addressing을 사용하는 웹 서비스 작성	657
기본 엔드포인트 참조	658
명시적 조치	659
WSDL 1.1의 기본 조치	660
WSDL 2.0의 기본 조치	662
메시지 교환	663
WS-Addressing의 필수 메시지 주소 지정 특성	665
WS-Addressing 보안	667
WS-Addressing 예제	667
WS-Addressing 용어	672
제 13 장 웹 서비스 보안 지원	675
웹 서비스 보안을 위한 필수 소프트웨어	676
SOAP 웹 서비스 보안 계획	677
SOAP 메시지 보안 옵션	678
보안 토큰 서비스를 사용하여 인증	680
신뢰 클라이언트 인터페이스	682
SOAP 메시지 서명	683
서명 알고리즘	683
서명된 SOAP 메시지의 예제	683
암호화된 SOAP 메시지에 대한 CICS 지원	684
암호화 알고리즘	685
암호화된 SOAP 메시지의 예제	686
웹 서비스 보안을 위한 RACF 구성	687

ID 전파를 위한 제공자 모드 웹 서비스 구성 . . .	689
웹 서비스 보안을 위한 파이프라인 구성 . . .	692
사용자 정의 보안 핸들러 작성 . . .	695
메시지 핸들러에서 트러스트 클라이언트 호출 . . .	697

제 14 장 웹 서비스 지원과 WSRR 간의 상호 운용성 . . .	701
웹 서비스 지원 및 WSRR에서 SSL을 사용하는 방법의 예제 . . .	701

제 15 장 SOAP 웹 서비스 문제점 해결 . . .	703
배치 오류 진단 . . .	703
서비스 제공자 런타임 오류 진단 . . .	706
서비스 요청자 런타임 오류 진단 . . .	708
MTOM/XOP 오류 진단 . . .	710
데이터 변환 오류 진단 . . .	713
데이터 변환 오류가 발생하는 이유 . . .	714
변환 오류에 대한 SOAP 결함 메시지 . . .	715

제 16 장 CICS 카탈로그 관리자 예제 애플리케이션 . . .	717
기본 애플리케이션 . . .	717
BMS 프리젠테이션 관리자 . . .	718
데이터 핸들러 . . .	718
디스패치 관리자 . . .	718
주문 디스패치 프로그램 . . .	719
재고 관리자 . . .	719
애플리케이션 구성 . . .	719
기본 애플리케이션 설치 및 설정 . . .	719
VSAM 데이터 세트 작성 및 정의 . . .	719
3270 인터페이스 정의 . . .	721
설치 완료 . . .	723

예제 애플리케이션 구성 . . .	723
BMS 인터페이스로 예제 애플리케이션 실행 . . .	726
예제 애플리케이션의 웹 서비스 지원 . . .	728
코드 페이지 지원 구성 . . .	734
웹 서비스 클라이언트 및 랩퍼 프로그램 정의 . . .	734
웹 서비스 지원 설치 . . .	735
웹 클라이언트 구성 . . .	742
웹 서비스 사용 애플리케이션 실행 . . .	743
예제 애플리케이션 배치 . . .	748
프로그램 인터페이스 추출 . . .	748
웹 서비스 지원 프로그램 DFHLS2WS 실행 . . .	750
생성된 WSDL 문서의 예제 . . .	752
웹 서비스 바인딩 파일 배치 . . .	753
기본 애플리케이션의 컴포넌트 . . .	754
카탈로그 관리자 프로그램 . . .	757
파일 구조 및 정의 . . .	762
카탈로그 파일 . . .	762
파일 구성 . . .	762

주의사항 . . .	765
상표 . . .	767

참고문헌 . . .	769
z/OS용 CICS Transaction Server의 CICS 서적 . . .	769
z/OS용 CICS Transaction Server에 대한 CICSplex SM 서적 . . .	770
기타 CICS 서적 . . .	771

내게 필요한 옵션 . . .	773

색인 . . .	775

서문

이 서적에서 제공하는 정보

이 서적은 CICS®의 웹 서비스를 사용하는 방법을 설명합니다.

이 책의 독자

이 책은 다음 역할을 대상으로 합니다.

- 웹 서비스 환경에서 CICS 애플리케이션 배치를 고려하는 계획자 및 설계자
- 웹 서비스를 지원하도록 CICS를 구성하는 작업에 책임이 있는 시스템 프로그래머
- 웹 서비스 환경에 배치할 애플리케이션에 책임이 있는 애플리케이션 프로그래머

Information Center에서 주제의 위치

이 서적의 주제는 z/OS용 CICS Transaction Server Knowledge Center에서도 찾을 수 있습니다. Knowledge Center는 정보가 표시되는 방식을 구조화시키기 위해 콘텐츠 유형을 사용합니다.

Knowledge Center 콘텐츠 유형은 일반적으로 업그레이드, 구성, 설치와 같은 태스크 중심적입니다. 다른 콘텐츠 유형으로는 참조, 개요, 시나리오 또는 학습서 기반 정보가 있습니다. 다음 맵핑은 이 서적의 주제와 외부 Knowledge Center에 대한 링크가 있는 Knowledge Center 콘텐츠 유형 간의 관계를 보여줍니다.

표 1. PDF 주제 대 Knowledge Center 콘텐츠 유형 맵핑. 이 표는 Knowledge Center의 콘텐츠 유형 주제와 PDF의 주제 간 관계를 나열합니다.

이 서적의 주제 세트	Knowledge Center의 위치
• 1 페이지의 제 1 장 『웹 서비스』 • 53 페이지의 제 5 장 『SOAP 웹 서비스 시작하기』 • 61 페이지의 제 6 장 『웹 서비스 인프라 작성』 • 61 페이지의 『SOAP 웹 서비스 작성』 • 675 페이지의 제 13 장 『웹 서비스 보안 지원』 • 703 페이지의 제 15 장 『SOAP 웹 서비스 문제점 해결』	• 제품 개요 • 시작하기 • 구성 • 애플리케이션 개발 • 보안 • 문제점 해결 및 지원

z/OS용 CICS Transaction Server, 버전 5 릴리스 3 의 변경사항

이번 릴리스의 변경사항에 대한 정보는 Knowledge Center의 새로운 기능, 또는 다음 서적을 참조하십시오.

- *z/OS용 CICS Transaction Server* 새로운 기능
- *CICS TS 버전 5.3으로 z/OS용 CICS Transaction Server 업그레이드*

릴리스 후 텍스트의 기술 변경사항은 각각의 변경된 또는 새 정보 행의 왼쪽에 새로 막대(I)로 표시됩니다.

제 1 장 웹 서비스

z/OS®용 CICS Transaction Server는 웹 서비스를 위한 종합적인 지원을 제공합니다.

- CICS 애플리케이션은 서비스 요청자로, 또는 서비스 제공자로 또는 둘 다로 이기종 웹 서비스 환경에 참여할 수 있습니다.
- CICS는 HTTP 및 IBM® WebSphere® MQ 전송 프로토콜을 지원합니다.
- z/OS용 CICS Transaction Server에는 사용자가 WSDL 및 JSON 서비스 설명을 상위 레벨 프로그래밍 언어 데이터 구조로 맵핑하거나 그 반대로 맵핑하는 것을 돕는 일련의 유틸리티 프로그램인 CICS 웹 서비스 지원이 포함됩니다. 이러한 유틸리티 프로그램은 다음 프로그래밍 언어를 지원합니다.

COBOL

PL/I

C

C++

- SOAP 웹 서비스에 대한 CICS 지원은 다음 표준을 포함하여 공개 표준을 준수합니다.

SOAP 1.1 및 1.2

HTTP 1.1

WSDL 1.1 및 2.0

JSON

- SOAP 웹 서비스에 대한 CICS 지원은 WS-I 기본 프로파일 버전 1.1을 포함한 다수의 웹 서비스 스펙을 조건부로 또는 완전히 준수하여 다른 웹 서비스 구현과의 상호 운용성을 최대한 보장합니다. 이 프로파일은 일련의 비독점 웹 서비스 스펙으로, 스펙에 대한 설명 및 수정사항을 포함하며 함께 사용될 때 여러 웹 서비스 구현 간 상호 운용성을 향상시킵니다.
- SOAP 웹 서비스에 대한 CICS 지원에는 Java™ 기반 및 비Java 기반인 웹 서비스 파이프라인에 대한 지원이 포함됩니다. Java 기반 파이프라인은 T8 TCB를 사용하여 처리되며 비Java 기반 파이프라인은 L8 TCB를 사용하여 처리됩니다. 이 경우 웹 서비스를 처리하는 데 필요한 QR TCB 처리의 양이 감소됩니다.
- CICS는 IBM z/OS XMLSS(XML System Services) 구문 분석기를 사용하여 SOAP 엔벨로프를 구문 분석합니다. 이 XMLSS 구문 분석기는 CICS 리전에서 64비트(막대 위) 저장영역을 사용하며 막대 아래로 사용자 프로그램을 위한 더 많은 저장영역을 남깁니다. 또한 XMLSS 구문 분석기는 XML 구문 분석이 zEnterprise® Application Assist Processor(zAAP)로 분담될 수 있도록 합니다. 웹 서비스 인프라의 zAAP 사용 가능 비율은 작지만 zAAP 용량이 사용 가능하면 CICS의 웹 서

비스 호스팅 비용을 줄일 수 있습니다. 자세한 정보는 IBM Redbooks® 서적 zAAP(zSeries Application Assist Processor) 구현을 참조하십시오.

웹 서비스의 개념

웹 서비스는 네트워크 주소 가능 위치에서 호스팅되는 상호 작동 가능한 기간 소프트웨어 기능에 대한 일반 용어입니다.

웹 서비스는 구현되는 하드웨어 또는 소프트웨어 플랫폼과 독립적으로 사용될 수 있고 작성되는 프로그래밍 언어와 독립적으로 사용될 수 있도록 구현 세부사항을 숨기는 인터페이스를 갖습니다. 이 독립성은 웹 서비스 기반 애플리케이션이 느슨한 결합, 구성요소 지향 및 교차 기술 구현이 되도록 합니다. 웹 서비스는 복잡한 집계 또는 비즈니스 트랜잭션을 수행하기 위해 단독으로 또는 기타 웹 서비스와 함께 사용될 수 있습니다.

CICS는 두 개의 특징적 웹 서비스 프로토콜(SOAP 및 JSON(JavaScript Object Notation) 프로토콜)을 지원합니다. 이러한 두 개의 프로토콜에는 다른 특성 및 이점이 있습니다.

SOAP 웹 서비스에는 WSDL(Web Service Definition Language) 문서라고 하는 기기 처리 가능 형식으로 설명되는 인터페이스가 있습니다. SOAP 웹 서비스는 메시지 형식, 전송 프로토콜 및 위치를 포함하여 서비스와의 상호작용을 위해 필요한 모든 세부사항을 제공하는 표준 형식 XML 개념을 사용하여 설명됩니다. 도구는 WSDL을 처리하기 위해 사용될 수 있고 XML 기반 SOAP 프로토콜을 사용하여 서비스와 통신할 수 있는 클라이언트 프로그램을 생성할 수 있습니다. SOAP는 상세한 통신 프로토콜일 수 있지만, 확장성의 이점을 갖습니다. 추가 스펙이 분배된 2단계 확약 지원 및 정교한 보안 옵션과 같은 엔터프라이즈 서비스 품질(QoS)을 지원하기 위해 존재합니다.

JSON 웹 서비스는 덜 형식적으로 정의됩니다. 데이터 형식은 JSON 스키마 표기법을 사용하여 설명되고 HTTP 전송 프로토콜의 사용이 필요합니다. JSON은 일반적 모바일 장치와 JavaScript 기반 애플리케이션에 더 편리한 데이터 표시 형식입니다. 그러나 SOAP의 확장성 옵션이 부족하여 엔터프라이즈 서비스 품질(QoS)에 대한 더 적은 옵션을 제공합니다. SOAP과 달리 경량 프로토콜입니다.

모바일 장치에서 CICS로 연결하려고 할 때 JSON을 사용하십시오. 서버 통신에 대한 서버를 원할 때 SOAP을 사용하십시오.

CICS의 SOAP과 JSON 웹 서비스 사이의 차이점

SOAP과 JSON 사이에 일부 중요한 차이가 있습니다.

- SOAP 메시지의 콘텐츠는 XML 데이터입니다. 반면에 JSON 메시지에는 JSON 데이터가 포함됩니다. JSON과 XML은 구조화 데이터를 설명하기 위한 다른 인코딩 메커니즘입니다.

- JSON은 더 효율적인 인코딩 메커니즘일 수 있습니다. 따라서 일반 JSON 메시지는 동등한 XML 메시지 보다 더 작습니다.
- JSON은 JavaScript 애플리케이션에서 통합하기 용이하지만, XML은 그렇지 않습니다. 이 차이는 JSON에게 많은 모바일 애플리케이션 개발자에 선호되는 데이터 형식을 작성해줍니다.
- SOAP에서는 메시지에 Headers를 추가하기 위한 메커니즘 및 서비스 품질(QoS)에 대한 스펙 제품군(예: 보안 구성 및 분배 트랜잭션)을 제공합니다. JSON은 이 메커니즘을 제공하지 않습니다. 대신 기본 HTTP 네트워크 프로토콜의 서비스에 따라 다릅니다. 이 신뢰는 워크로드를 보안하고 구성하기 위한 더 적은 옵션의 결과를 가져옵니다. JSON 아키텍처는 SOAP과 비교하여 종종 경량으로 설명됩니다.
- SOAP 웹 서비스는 WSDL 문서를 사용하여 설명됩니다. JSON 웹 서비스는 덜 형식적으로 구조화됩니다. 느슨하게 연결되고 예제를 포함하여 종종 비공식적 문서를 선호하는 경향이 있습니다.
- SOAP는 애플리케이션 개발을 도울 수 있는 관련 도구의 더 광범위한 생태계를 가집니다.
- SOAP 웹 서비스는 SOAP 결함 메시지를 사용하는 것을 포함하는 명시적 오류 형식을 가지고 있습니다. JSON에 대한 평가가 없습니다.
- SOAP 웹 서비스는 HTTP 및 WebSphere MQ 기반 메시징 모두의 사용을 지원하고 JSON에는 HTTP가 필요합니다.
- JSON 웹 서비스는 RESTful 및 요청-응답 구동형 인터페이스 모두를 지원하고, SOAP는 요청-응답 인터페이스만 지원합니다.
- SOAP 웹 서비스는 CICS에서 **INVOKE** API 명령을 지원합니다. 이 API 명령 요청자(또는 클라이언트) 모드 애플리케이션을 사용하면 원격 SOAP 웹 서비스를 호출할 수 있습니다. JSON은 **INVOKE** 명령을 지원하지 않지만, JSON 클라이언트 애플리케이션이 기본 WEB API의 사용을 통해 CICS에 호스팅될 수 있습니다.

이러한 차이점에도 불구하고 JSON과 SOAP 사이에 많은 유사성이 있습니다. 양쪽 모든 프로토콜은 크로스벤더 개방 기술이고 양쪽 모두는 CICS의 인프라, 동일한 CICS 제공 도구 및 많은 동일한 구성을 공유합니다. SOAP과 JSON 인터페이스 모두를 CICS에서 호스팅되는 기존 애플리케이션 프로그램에 제공할 수 있습니다.

SOAP에 대한 자세한 정보는 시작하기에서 SOAP 웹 서비스로 시작하기의 내용을 참조하십시오. JSON에 대한 자세한 정보는 시작하기에서 JSON 웹 서비스로 시작하기의 내용을 참조하십시오.

비즈니스에 웹 서비스를 활용하는 방법

웹 서비스는 월드 와이드 웹(WWW)을 통해 비즈니스 기능을 배치하고 이 기능에 대한 액세스를 제공하기 위한 기술입니다. 웹 서비스를 사용하여 애플리케이션을 웹에 통합하십시오.

웹 서비스는 다음과 같은 측면에서 비즈니스에 도움이 될 수 있습니다.

- 비즈니스 비용 감소
- 보다 빠른 솔루션 배치를 가능하게 함
- 새로운 기회가 생김

이러한 이점을 모두 얻기 위한 키는 기존 및 신규 표준(예: HTTP, JSON, SOAP, WSDL 및 XML)을 기반으로 빌드되는 공통 프로그램 간 통신 모델입니다.

웹 서비스에 대해 CICS가 제공하는 지원을 통해 재프로그래밍의 양을 최소로 유지하면서 기존 애플리케이션을 새로운 방식으로 배치할 수 있습니다.

웹 서비스 용어

웹 서비스 섹션의 주제를 이해하려면 다음과 같은 용어에 익숙해져야 합니다.

XML(Extensible Markup Language)

문서 마크업의 표준으로, 일반 구문을 사용하여 단순하고 사람이 읽을 수 있는 태그가 포함된 데이터를 마크업합니다. 이 표준은 W3C(World Wide Web Consortium)에 의해 승인됩니다.

초기 SOAP 발신인

SOAP 메시지 경로의 시작점에서 SOAP 메시지를 작성하는 SOAP 발신인입니다.

JSON(JavaScript Object Notation)

JavaScript의 오브젝트 리터럴 표기법을 기반으로 하는 경량 데이터 교환 형식입니다. JSON은 프로그래밍 언어 중립적이지만 C, C++, C#, Java, JavaScript, Perl, Python과 같은 언어의 규칙을 사용합니다.

JSON 스키마

구조를 설명하고 다른 JSON 문서의 콘텐츠를 제한하는 JSON(JavaScript Object Notation) 문서입니다.

RESTful

REST(Representational State Transfer) 제한조건에 따른 애플리케이션과 서비스에 관한 용어입니다.

서비스 제공자

웹 서비스를 제공하는 소프트웨어의 컬렉션입니다.

서비스 제공자 애플리케이션

서비스 제공자에서 사용되는 애플리케이션입니다. 일반적으로 서비스 제공자 애플리케이션은 서비스 제공자의 비즈니스 로직 컴포넌트를 제공합니다.

서비스 요청자

웹 서비스를 서비스 제공자에게 요청하는 데 책임이 있는 소프트웨어의 콜렉션입니다.

서비스 요청자 애플리케이션

서비스 요청자에서 사용되는 애플리케이션입니다. 일반적으로 서비스 요청자 애플리케이션은 서비스 요청자의 비즈니스 로직 컴포넌트를 제공합니다.

SOAP(Simple Object Access Protocol)

SOAP을 참조하십시오.

SOAP 이전에 *Simple Object Access Protocol*의 약어입니다. 분산된 분배 환경에서 정보를 교환하기 위한 경량 프로토콜입니다. 이는 다음 세 개의 파트로 구성되는 XML 기반 프로토콜입니다.

- 메시지의 내용과 처리 방법을 설명하는 프레임워크를 정의하는 엔벨로프
- 애플리케이션 정의 데이터 유형의 인스턴스를 표시하기 위한 인코딩 규칙 세트
- 원격 프로시저 호출 및 응답 표시 관련 규칙

SOAP는 HTTP와 같은 다른 프로토콜과 함께 사용할 수 있습니다.

SOAP 1.1의 스펙은 SOAP(Simple Object Access Protocol) 1.1에 공개되어 있습니다.

SOAP 1.2의 스펙은 다음에 공개되어 있습니다.

SOAP 버전 1.2 파트 0: 입문서

SOAP 버전 1.2 파트 1: 메시징 프레임워크

SOAP 버전 1.2 파트 2: 부속서

SOAP 중개자

SOAP 노드는 SOAP 수신인 및 SOAP 발신인 둘 다이며 SOAP 메시지 내에서 대상으로 정할 수 있습니다. 이 노드는 자신을 대상으로 하는 SOAP 헤더 블록을 처리하고 SOAP 메시지를 최종 SOAP 수신인에게 전달합니다.

SOAP 메시지 경로

단일 SOAP 메시지가 통과하는 SOAP 노드 세트입니다. 이러한 노드에는 초기 SOAP 발신인, 0개 이상의 SOAP 중개자 및 최종 SOAP 수신인이 포함되어 있습니다.

SOAP 노드

SOAP 메시지를 조작하는 처리 로직입니다.

SOAP 수신인

SOAP 메시지를 승인하는 SOAP 노드입니다.

SOAP 발신인

SOAP 메시지를 전송하는 SOAP 노드입니다.

최종 SOAP 수신인

SOAP 메시지의 최종 목적지인 SOAP 수신인입니다. SOAP 본문의 콘텐츠 및 자신을 대상으로 하는 SOAP 헤더 블록의 처리를 담당합니다.

UDDI UDDI(Universal Description, Discovery and Integration)를 참조하십시오.

UDDI(Universal Description, Discovery and Integration)

UDDI(Universal Description, Discovery and Integration)는 웹 서비스의 분배된 웹 기반 정보 레지스트리 스펙입니다. UDDI는 비즈니스에서 제공하는 웹 서비스에 대한 정보를 다른 비즈니스에서 찾을 수 있도록 등록하는 데 비즈니스에서 사용할 수 있는 스펙의 구현 세트로, 공개적으로 액세스 가능합니다. 이 스펙은 OASIS에 공개되어 있습니다.

웹 서비스

네트워크를 통해 상호 운영 가능한 시스템 간 상호작용을 지원하도록 설계된 소프트웨어 시스템입니다. 여기에는 시스템 처리 가능한 형식(특히, WSDL)으로 설명된 인터페이스가 있습니다.

WS-Addressing

웹 서비스 주소 지정(WS-Addressing)은 웹 서비스 및 메시지의 주소를 지정하기 위한 전송 중립 메커니즘을 제공합니다.

WS-Addressing의 스펙은 다음에 공개되어 있습니다.

- Web Services Addressing 1.0 - 코어
- Web Services Addressing 1.0 - SOAP 바인딩
- Web Services Addressing 1.0 - 메타데이터
- Web Services Addressing - 제출

WSAT(Web Services Atomic Transaction)

특성이 "all" 또는 "nothing"인 활동을 조정하는 데 사용되는 원자적 트랜잭션 조정 유형의 정의를 제공하는 스펙입니다.

이 스펙은 OASIS에 의해 WSAT(Web Services Atomic Transaction)에 공개되어 있습니다.

웹 서비스 바인딩 파일

CICS가 입력 및 출력 메시지 및 애플리케이션 데이터 구조 사이에서 데이터를 맵핑하는 데 사용하는 정보를 포함하는 파일이며 WEBSERVICE 자원과 연관됩니다.

웹 서비스 설명

서비스 제공자가 서비스 요청자에 대한 웹 서비스 호출을 위해 스펙을 통신하는 데 사용하는 XML 문서입니다. 웹 서비스 설명은 WSDL(Web Service Description Language)로 작성됩니다.

WSDL(Web Service Description Language)

웹 서비스를 설명하기 위한 XML 애플리케이션입니다. 이 애플리케이션은 서비스에서 제공되는 추상 기능에 대한 설명과 해당 기능이 제공되는 방식 및 위치와 같은 서비스의 구체적 세부사항을 구분하도록 디자인됩니다.

이 스펙은 WSDL(Web Services Description Language)에 공개되어 있습니다.

웹 서비스 보안

메시지 무결성 및 기밀성을 제공하는 SOAP 메시지에 대한 일련의 개선사항입니다. 이 스펙은 OASIS에 의해 웹 서비스 보안: SOAP Message Security 1.0(WS-Security 2004)에 공개되어 있습니다.

WS-Atomic 트랜잭션

WSAT(Web Services Atomic Transaction)를 참조하십시오.

WS-I 기본 프로파일

일련의 비독점 웹 서비스 스펙으로, 스펙에 대한 설명 및 수정사항을 포함하며 여러 웹 서비스 구현 간 상호 운용성을 향상시킵니다. 이 프로파일은 WS-I(Web Services Interoperability Organization)에 의해 정의되며 버전 1.0은 WS-I(Web Services Interoperability Organization) Basic Profile 1.0에 있습니다.

WSDL

WSDL(Web Service Description Language)을 참조하십시오.

WSS WSS(Web Services Security)를 참조하십시오.

XML XML(Extensible Markup Language)입니다.

XML의 스펙은 다음에 공개되어 있습니다.

SOAP 버전 1.2 파트 0: 입문서

SOAP 버전 1.2 파트 1: 메시징 프레임워크

SOAP 버전 1.2 파트 2: 부속서

XML 네임스페이스

URI 참조로 식별되며 XML 문서에서 요소 유형 및 속성 이름으로 사용되는 이름의 컬렉션입니다.

XML 스키마

구조를 설명하고 다른 XML 문서의 콘텐츠를 제한하는 XML 문서입니다.

XML 스키마 정의 언어

W3C(World Wide Web Consortium)에서 권장되는 XML 스키마를 작성하기 위한 XML 구문입니다.

제 2 장 SOAP 웹 서비스 아키텍처

SOAP 웹 서비스 아키텍처는 세 개의 구성요소(서비스 제공자, 서비스 요청자 및 선택적 서비스 레지스트리) 사이의 상호작용을 기반으로 합니다.

서비스 제공자

웹 서비스를 제공하는 소프트웨어의 콜렉션입니다.

- 애플리케이션 프로그램
- 미들웨어
- 실행하는 플랫폼

서비스 요청자

웹 서비스를 서비스 제공자에게 요청하는 데 책임이 있는 소프트웨어의 콜렉션입니다.

- 애플리케이션 프로그램
- 미들웨어
- 실행하는 플랫폼

서비스 레지스트리

서비스 레지스트리는 서비스 제공자가 해당 서비스 설명을 공개할 수 있고 서비스 요청자가 해당 서비스 설명을 찾을 수 있는 중심 위치입니다.

서비스 요청자 및 제공자가 많은 상황에서 레지스트리 없이 통신할 수 있으므로 레지스트리는 웹 서비스 아키텍처의 선택적 구성요소입니다. 예를 들어, 서비스를 제공하는 조직은 FTP 사이트로부터 서비스를 다운로드 하여 제공하는 것을 포함하여 많은 방법으로 서비스의 사용자에게 서비스 설명을 직접적으로 배포할 수 있습니다.

서비스 레지스트리 사용은 요청자 및 제공자 모두에게 많은 이점을 제공합니다. 예를 들어, IBM WSRR(WebSphere Service Registry and Repository) 사용은 요청자가 더 빠르게 서비스를 찾을 수 있도록 도와주고 제공자가 제공되는 서비스의 버전 제어를 강제 실행할 수 있도록 도움을 줄 수 있습니다.

CICS에서는 서비스 요청자 및 서비스 제공자 구성요소를 구현하기 위한 직접적 지원을 제공합니다. 그러나, CICS에서 서비스 레지스트리를 배치하기 위한 추가 소프트웨어가 필요합니다. IBM WSRR(WebSphere Service Registry and Repository)을 사용하는 경우, CICS에서는 웹 서비스 지원을 통하여 WSRR에 대한 지원을 제공합니다. 그렇지 않으면, 다른 플랫폼에 서비스 레지스트리를 배치할 수 있습니다.

서비스 제공자, 서비스 요청자 및 서비스 레지스트리 사이의 상호작용

서비스 제공자, 서비스 요청자 및 서비스 레지스트리 사이의 상호작용에는 다음 조작이 포함됩니다.

공개 서비스 레지스트리가 사용될 때, 서비스 제공자는 찾을 서비스 요청자의 서비스 레지스트리에서 해당 서비스 설명을 공개합니다.

찾기 서비스 레지스트리가 사용될 때, 서비스 요청자는 레지스트리에서 서비스 설명을 찾습니다.

바인드 서비스 요청자는 서비스 설명을 사용하여 서비스 제공자와 바인딩하고 웹 서비스 구현과 상호작용합니다.

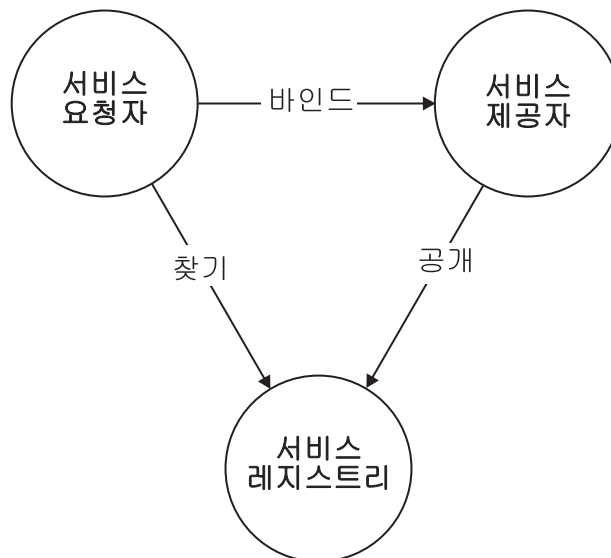


그림 1. 웹 서비스 구성요소 및 상호작용

웹 서비스 설명

웹 서비스 설명은 서비스 제공자가 웹 서비스를 시작하기 위한 스펙을 서비스 요청자로 전달하는 문서입니다. 웹 서비스 설명은 WSDL(Web Service Description Language)로 알려진 XML 애플리케이션으로 표시됩니다.

서비스 설명은 서비스 제공자와 서비스 요청자 간의 통신을 보장하는 데 사용되는 사용자 정의 프로그래밍과 공유 지식의 양을 최소화하는 방식으로 웹 서비스를 설명합니다. 예를 들어, 요청자와 제공자 모두 다른 요청자 또는 제공자가 실행되는 플랫폼과 다른 요청자 또는 제공자가 작성되는 프로그래밍 언어를 인식하지 않아도 됩니다.

서비스 설명은 WSDL 1.1 또는 WSDL 2.0 스펙을 준수할 수 있습니다. 각 스펙은 서비스 설명에 포함될 수 있는 주 요소와 용어가 둘 다 다릅니다. 다음 정보는 WSDL 1.1 용어와 요소를 사용하여 서비스 설명의 목적을 설명합니다.

WSDL 구조에서는 서비스 설명이 다음 두 정의로 분할될 수 있습니다.

- 서비스 인터페이스를 설명하고 서비스를 구현 및 시작하는 프로그램을 작성할 수 있게 하는 추상 서비스 인터페이스 정의
- 제공자 웹 서비스의 네트워크(또는 엔드포인트) 위치와 기타 구현 특정 세부사항을 설명하는 구체적인 서비스 구현 정의. 서비스 요청자가 서비스 제공자에 연결될 수 있습니다.

12 페이지의 그림 2를 참조하십시오.

WSDL 1.1 문서는 네트워크 서비스 정의에서 다음 주 요소를 사용합니다.

<types>

일부 유형 시스템(예: XML 스키마)을 사용하는 데이터 유형 정의의 컨테이너. 메시지 내에서 사용되는 데이터 유형을 정의합니다. 모든 메시지가 단순 데이터 유형으로 구성되는 경우에는 <types> 요소가 필요하지 않습니다.

<message>

조작의 입력 및 출력 매개변수를 정의하는 데 사용되는 XML 데이터 유형을 지정합니다.

<portType>

하나 이상의 엔드포인트가 지원하는 조작 세트를 정의합니다. <portType> 요소 내에서는 각 조작을 <operation> 요소로 설명합니다.

<operation>

입력 및 출력 데이터 플로우에서 나타날 수 있는 XML 메시지를 지정합니다. 조작과 프로그래밍 언어의 메소드 서명을 비교할 수 있습니다.

<binding>

특정 <portType> 요소의 프로토콜, 데이터 형식, 보안 및 기타 속성을 설명합니다.

<port>

엔드포인트의 네트워크 주소를 지정하고 <binding> 요소와 연관시킵니다.

<service>

웹 서비스를 관련 엔드포인트의 컬렉션으로 정의합니다. <service> 요소에는 하나 이상의 <port> 요소가 포함됩니다.

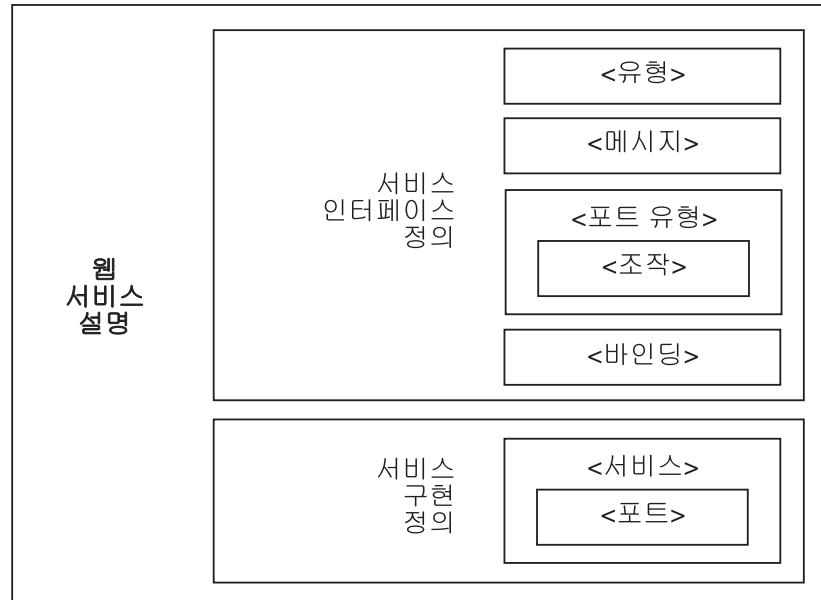


그림 2. 웹 서비스 설명 구조

웹 서비스 설명을 파티션화할 수 있으므로 전체 서비스 설명을 작성해야 하는 책임을 나눌 수 있습니다. 예를 들어, 산업에서 사용할 표준 본문으로 정의되고 해당 산업의 개별 회사에서 구현하는 서비스를 고려하십시오.

- 표준 본문은 다음 요소를 포함하는 서비스 인터페이스 정의를 제공합니다.

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

- 서비스 구현을 제공하려는 서비스 제공자는 다음 요소를 포함하는 서비스 구현 정의를 제공합니다.

```
<port>
<service>
```

서비스 관련 서적

수많은 다른 메커니즘을 사용하여 서비스 설명을 공개할 수 있습니다. 각 메커니즘은 다른 상황에서 사용하기에 적합합니다. CICS는 서비스 설명을 공개하기 위한 WSRR(IBM WebSphere Service Registry and Repository)의 사용을 지원합니다. 그렇지 않으면, 서비스 설명을 공개하기 위해 다른 메소드를 사용할 수 있습니다.

WSSR

CICS는 서비스 설명을 공개하기 위해 WSRR의 사용을 지원합니다. CICS에서 WSSR에 제공하는 지원에 대한 자세한 정보는 Information Center에서 "웹 서비스 지원과 WSRR 사이의 상호 운용성" 주제를 참조하십시오.

아무것도 CICS에 의해 직접적으로 지원되지 않는 다음 메커니즘은 서비스 설명을 공개하기 위해 CICS와 함께 사용될 수 있습니다.

직접 공개

이 메커니즘은 서비스 설명을 공개하기 위해 가장 간단합니다. 서비스 제공자는 이메일 첨부, FTP 사이트 또는 CD ROM 배포를 사용하여 서비스 요청자에 직접적으로 서비스 설명을 전송합니다.

DISCO

이러한 배타적 프로토콜은 동적 서적 메커니즘을 제공합니다. 서비스 요청자는 단순 HTTP GET 메커니즘을 사용하여 서비스 제공자에 의해 지정되고 URI로 식별되는 네트워크 위치로부터 웹 서비스 설명을 검색합니다.

UDDI(Universal Description, Discovery and Integration)

웹 서비스의 분배된 웹 기반 정보 레지스트리에 대한 스펙. UDDI는 기타 비즈니스가 찾을 수 있도록 제공하는 웹 서비스에 대한 정보를 비즈니스가 등록할 수 있도록 허용하는 공개적으로 액세스 가능한 스펙의 구현 세트입니다.

서비스 설명은 필요한 경우 둘 이상의 양식으로 공개될 수 있습니다.

제 3 장 SOAP

SOAP는 분산 환경에서 정보 교환을 위한 프로토콜입니다. SOAP 메시지는 XML 문서로 인코딩되며 다양한 기본 프로토콜을 사용하여 교환될 수 있습니다.

이전 *Simple Object Access Protocol*의 약어인 SOAP는 W3C(World Wide Web Consortium)에서 개발되었으며 W3C에서 발행한 다음 문서에 그 정의가 나와 있습니다. SOAP에 대한 정확한 전체 정보는 다음 문서를 참조하십시오.

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

SOAP 버전 1.2 파트 0: 입문서(W3C 권장사항)

SOAP 버전 1.2 파트 1: 메시징 프레임워크(W3C 권장사항)

SOAP 버전 1.2 파트 2: 부록(W3C 권장사항)

SOAP 스펙은 SOAP 메시지가 SOAP 노드 사이에 전달되는 분산 처리 모델에 대해 설명합니다. 메시지는 SOAP 발신인에서 발송되어 SOAP 수신인으로 전송됩니다. 발신인과 수신인 사이에서 하나 이상의 SOAP 중개자가 메시지를 처리할 수 있습니다.

SOAP 메시지는 SOAP 발신인에서 SOAP 수신인까지 SOAP 노드 간 단방향 전송이지만 메시지가 결합되어 보다 복잡한 상호작용을 구성할 수 있습니다(예를 들어, 요청 및 응답, 피어 투 피어 대화).

스펙에는 또한 다음 정보가 포함됩니다.

- 애플리케이션 정의 데이터 유형의 인스턴스를 표시하기 위한 인코딩 규칙 세트
- 원격 프로시저 호출(RPC) 및 응답 표시 규칙

SOAP 메시지 구조

SOAP 메시지는 XML 문서로 인코딩되며, 선택적 <Header> 요소와 필수 <Body> 요소를 포함하는 <Envelope> 요소로 구성됩니다. <Body>에 포함되는 <Fault> 요소는 오류 보고를 위해 사용됩니다.

SOAP 엔벨로프

SOAP <Envelope>는 모든 SOAP 메시지의 루트 요소입니다. 두 개의 하위 요소(선택적 <Header> 및 필수 <Body>)가 포함됩니다.

SOAP 헤더

SOAP <Header>는 SOAP 엔벨로프의 선택적 하위 요소입니다. 이 요소는 메시지 경로를 따라 SOAP 노드가 처리하는 애플리케이션 관련 정보를 전달하는 데 사용됩니다.

SOAP 본문

SOAP <Body>는 SOAP 엔벨로프의 필수 하위 요소입니다. 메시지의 최종 수신인을 위한 정보를 포함합니다.

SOAP 결함

SOAP <Fault>는 SOAP 본문의 하위 요소로, 오류 보고를 위해 사용됩니다.

SOAP 메시지의 <Body>에 포함되는 <Fault> 요소를 제외하고, <Header> 및 <Body>의 XML 요소는 이 요소를 사용하는 애플리케이션에서 정의됩니다. 그러나 SOAP 스펙은 해당 요소 구조에 일부 제한조건을 적용합니다.

그림 3은 SOAP 메시지의 주 요소를 보여줍니다.



그림 3. SOAP 메시지 구조

17 페이지의 그림 4는 헤더 블록(<m:reservation> 및 <n:passenger> 요소)과 본문 (<p:itinerary> 및 <q:lodging> 요소 포함)을 포함하는 SOAP 메시지의 예제입니다.

```

<?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>

```

그림 4. SOAP 1.2 메시지 예제

SOAP 헤더

SOAP <Header>는 SOAP 메시지의 선택적 요소입니다. 이 요소는 메시지 경로를 따라 SOAP 노드가 처리하는 애플리케이션 관련 정보를 전달하는 데 사용됩니다.

<Header> 요소의 즉각적인 하위 요소를 헤더 블록이라고 합니다. 헤더 블록은 애플리케이션 정의 XML 요소이며 발신인으로부터 최종 수신인까지의 메시지 경로에 나타날 수 있는 SOAP 노드에서 대상이 될 수 있는 데이터의 논리적 그룹을 나타냅니다.

SOAP 헤더 블록은 SOAP 중개자 노드와 최종 SOAP 수신인 노드에서 처리할 수 있습니다. 그러나 실제 애플리케이션에서 모든 노드가 모든 헤더 블록을 처리하지는 않습니다.

니다. 오히려, 각 노드는 일반적으로 특정 헤더 블록을 처리하도록 디자인되고 반대로 각 헤더 블록은 특정 노드에서 처리되어야 합니다.

SOAP 헤더는 통신 주체 간에 사전 동의 없이 분산 방식으로 SOAP 메시지에 기능을 추가할 수 있도록 허용합니다. SOAP는 기능 처리 주체와 기능이 선택사항 또는 필수 사항인지 여부를 나타내는 데 사용될 수 있는 몇 가지 속성을 정의합니다. 그러한 "제어" 정보에는 예를 들어, 메시지 처리와 관련된 지시 또는 컨텍스트 정보 전달이 포함됩니다. 이러한 방식으로, 애플리케이션에 따라 SOAP 메시지가 확장될 수 있습니다.

헤더 블록은 애플리케이션으로 정의되지만 헤더 블록에 대한 SOAP 정의 속성은 SOAP 노드가 헤더 블록을 처리하는 방식을 나타냅니다. 다음 중요 속성을 참조하십시오.

encodingStyle

SOAP 메시지의 파트를 인코딩하는 데 사용되는 규칙을 나타냅니다. SOAP는 데이터 인코딩에 있어 XML이 허용하는 아주 유연한 인코딩보다 제한적인 규칙 세트를 정의합니다.

role(SOAP 1.2)

actor(SOAP 1.1)

SOAP 1.2의 경우 **role** 속성은 특정 노드가 메시지에 대해 실행되는지 여부를 지정합니다. 노드에 지정된 역할이 헤더 블록의 역할 속성과 일치하면 노드가 헤더를 처리합니다. 역할이 일치하지 않으면 노드가 헤더 블록을 처리하지 않습니다. SOAP 1.1의 경우에는 **actor** 속성의 기능이 같습니다.

역할은 애플리케이션이 정의할 수 있으며 URI로 지정됩니다. 예를 들어, `http://example.com/Log`는 로깅을 수행하는 노드의 역할을 지정할 수 있습니다. 이 노드로 처리되는 헤더 블록이 `env:role="http://example.com/Log"`를 지정합니다. 여기서 네임스페이스 접두부 `env`는 `http://www.w3.org/2003/05/soap-envelope`의 SOAP 네임스페이스 이름과 연관됩니다.

SOAP 1.2 스펙은 애플리케이션이 정의하는 역할 이외에 다음 세 가지 표준 역할을 정의합니다.

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

메시지 경로의 SOAP 노드가 헤더 블록을 직접 처리하지 않습니다. 이 역할의 헤더 블록은 다른 SOAP 헤더 블록을 처리하는 데 필요한 데이터를 전달하는 데 사용될 수 있습니다.

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

메시지 경로에서 이전에 노드가 헤더를 제거하지 않은 경우 메시지 경로의 모든 SOAP 노드가 헤더 블록을 검사합니다.

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

최종 수신인 노드만 헤더 블록을 검사합니다.

mustUnderstand

이 속성은 SOAP 노드가 애플리케이션의 전체 목적에 중요한 헤더 블록을 무시하지 않도록 하는 데 사용됩니다. SOAP 노드가 **role** 또는 **actor** 속성을 사용하여 헤더 블록을 처리할 것으로 판별하고 **mustUnderstand** 속성이 "true" 값을 갖는 경우에는 노드가 해당 스펙과 일치하는 방식으로 헤더 블록을 처리하거나 절대로 처리하지 않고 결함을 처리해야 합니다. 그러나 속성이 "false" 값을 갖는 경우에는 노드가 헤더 블록을 처리하지 않습니다.

실제로 **mustUnderstand** 속성은 헤더 블록 처리가 필수 또는 선택사항인지 여부를 나타냅니다.

mustUnderstand 속성은 다음과 값을 갖습니다.

true(SOAP 1.2)

1(SOAP 1.1)

노드가 해당 스펙과 일치하는 방식으로 헤더 블록을 처리하거나 절대로 처리하지 않고 결함을 처리해야 합니다.

false(SOAP 1.2)

0(SOAP 1.1)

노드가 헤더 블록을 처리하지 않아도 됩니다.

relay(SOAP 1.2만 해당)

SOAP 중재가 노드가 헤더 블록을 처리할 때 SOAP 메시지에서 제거합니다. 기본적으로 무시된 헤더 블록도 제거합니다. **mustUnderstand** 속성 값이 "false"이기 때문입니다. 그러나 **relay** 속성이 "true" 값으로 지정되는 경우에는 노드가 처리되지 않은 헤더 블록을 메시지에 보유합니다.

SOAP 본문

<Body>는 SOAP 메시지로 전달되는 기본 엔드투엔드 정보를 제공하는 SOAP 엔벨로프의 필수 요소입니다.

<Body> 요소와 연관된 하위 요소는 초기 SOAP 발신인과 궁극의 SOAP 수신인 간에 정보를 교환하는 데 사용됩니다. SOAP는 <Body>의 하위 요소 중 하나(오류 보고에 사용되는 <Fault> 요소)를 정의합니다. <Body>의 다른 요소는 해당 요소를 사용하는 웹 서비스로 정의됩니다.

SOAP 결함

SOAP <Fault> 요소는 SOAP 메시지에서 오류 및 상태 정보를 제공합니다.

웹 서비스에서 오류가 발생하면 클라이언트로 결함 메시지가 리턴됩니다. 결함 메시지의 기본 구조는 SOAP 스펙에 정의됩니다. 각 결함 메시지에는 특정 오류 조건을 설명하는 XML이 포함될 수 있습니다. 예를 들어, CICS 웹 서비스에서 애플리케이션 이상 종료 발생하면 이상 종료를 보고하는 클라이언트로 결함 메시지가 리턴됩니다.

CICS는 다음과 같이 다양한 유형의 결함 메시지를 보낼 수 있습니다.

- 표준 SOAP 결함 메시지는 SOAP 스펙 또는 CICS에서 지원되는 웹 서비스 스펙 중 하나로 정의됩니다. 결함 메시지는 잘못된 양식의 SOAP 엔벨로프와 같은 공통 오류 조건을 보고합니다.
- 애플리케이션 SOAP 결함 메시지는 애플리케이션이 발견하거나 처리하는 조건에 대한 응답으로 **EXEC CICS SOAPFAULT** API 명령을 사용하여 생성됩니다. 이러한 결함 메시지의 구조는 애플리케이션에는 알려져 있지만 CICS에는 알려져 있지 않습니다.
- SOAP 핸들러 결함 메시지는 CICS의 일반 오류 핸들링에 대한 응답으로 SOAP 핸들러 프로그램에서 생성됩니다. 예를 들어, SOAP 핸들러 프로그램은 이상 종료에 대한 SOAP 결함, XML 구문 분석 실패, 기타 공통 오류를 보냅니다.
- 애플리케이션 핸들러 결함 메시지는 SOAP 메시지 본문을 처리할 때 오류 찾기에 대한 응답으로 CICS SOAP 애플리케이션 핸들러에 의해 생성됩니다. 이러한 결함은 XML을 2진 애플리케이션 데이터로 변환하는 프로세스 또는 응답을 생성할 때 발생합니다.

오류가 발생하는 경우 SOAP <Fault> 요소는 본문 항목이어야 하며 <Body> 요소에서 여러 번 나타나서는 안됩니다. SOAP <Fault> 요소 내부의 XML 요소는 SOAP 1.1과 SOAP 1.2에서 다릅니다.

SOAP 1.1

SOAP 1.1의 경우 SOAP <Fault> 요소에는 다음 요소가 포함됩니다.

<faultcode>

<faultcode> 요소는 <Fault> 요소의 필수 요소입니다. 결함에 대한 정보를 소프트웨어가 처리할 수 있는 양식으로 제공합니다. SOAP는 기본 SOAP 결함을 다루는 작은 SOAP 결함 코드 세트를 정의하며 이 세트는 애플리케이션에 의해 확장될 수 있습니다.

<faultstring>

<faultstring> 요소는 <Fault> 요소의 필수 요소입니다. 결함에 대한 정보를 사용자를 위한 양식으로 제공합니다.

<faultactor>

<faultactor> 요소에는 결함을 생성한 SOAP 노드의 URI가 포함됩니다. 최종 SOAP 수신인이 아닌 SOAP 노드에는 결함이 작성되는 경우 <faultactor> 요소가 포함되어야 합니다. 최종 SOAP 수신인은 이 요소를 포함해서는 안되지만 포함할 수도 있습니다.

<detail>

<detail> 요소는 <Body> 요소와 관련된 애플리케이션 특정 오류 정보를 전달합니다. <Body> 요소의 콘텐츠가 처리되지 않은 경우 이 요소가 있어야 함

니다. 이 요소는 헤더 항목에 속하는 오류 정보에 대한 정보를 전달하기 위해 사용해서는 안됩니다. 헤더 항목에 속하는 세부 오류 정보는 헤더 항목으로 전달해야 합니다.

SOAP 1.2

SOAP 1.2의 경우 SOAP <Fault> 요소에는 다음 요소가 포함됩니다.

<Code>

<Code> 요소는 <Fault> 요소의 필수 요소입니다. 결함에 대한 정보를 소프트웨어가 처리할 수 있는 양식으로 제공합니다. 이 요소에는 <Value> 요소와 선택적 <Subcode> 요소가 포함됩니다.

<Reason>

<Reason> 요소는 <Fault> 요소의 필수 요소입니다. <Reason> 요소에는 하나 이상의 <Text> 요소가 포함되며 이들 요소는 각각 다른 기본 언어의 결함에 대한 정보를 포함합니다.

<Node>

<Node> 요소에는 결함을 생성한 SOAP 노드의 URI가 포함됩니다. 최종 SOAP 수신인이 아닌 SOAP 노드에는 결함이 작성되는 경우 <Node> 요소가 포함되어야 합니다. 최종 SOAP 수신인은 이 요소를 포함해서는 안되지만 포함할 수도 있습니다.

<Role>

<Role> 요소에는 결함 발생 시점에 노드가 수행한 역할을 식별하는 URI가 포함됩니다.

<Detail>

<Detail> 요소는 선택적 요소로, 결함을 설명하는 SOAP 결함 코드와 관련된 애플리케이션 특정 오류 정보를 포함합니다. <Detail> 요소 존재는 결함이 있는 SOAP 메시지에서 처리된 파트와 관련하여 아무런 의미가 없습니다.

SOAP 결함 예제와 스키마

다음 예제는 SOAP 메시지의 본문을 처리할 때 애플리케이션 핸들러, DFHPITP에 의해 생성되는 SOAP 결함 메시지를 보여줍니다.

```
<SOAP-ENV:Fault xmlns="">
  <faultcode>SOAP-ENV:Server</faultcode>
  <faultstring>Conversion to SOAP failed</faultstring>
  <detail>
    <CICSFault xmlns="http://www.ibm.com/software/http/cics/WSFault">
      DFHP11008 25/01/2010 14:16:50 IYCWZCFU 00340 XML
      generation failed because of incorrect input
      (CONTAINER_NOT_FOUND container name) for WEBSERVICE
    </CICSFault>
  </detail>
</SOAP-ENV:Fault>
```

```

        servicename.
      </CICSFault>
    </detail>
  </SOAP-ENV:Fault>

```

이 예제의 콘텐츠는 대부분 모든 결함 메시지에 공통적입니다. <Detail> 요소에는 애플리케이션 핸들러에서 발생한 문제점을 설명하는 고유 정보가 포함됩니다. 이 특정 결함 메시지에는 CICS 메시지 로그에 기록되는 오류 메시지의 사본이 포함됩니다. 애플리케이션 핸들러 SOAP 결함 구문을 프로그래밍 방식으로 분석하려면 다음 XML 스키마를 사용하십시오.

```

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

```

일반적인 결함 메시지는 보다 복잡합니다. 이는 <Detail> 섹션이 여러 가지 다른 방식으로 구성될 수 있기 때문입니다. SOAP 핸들러 결함 구문을 프로그래밍 방식으로 분석하려는 경우, `usshome/schemas/soapfault/soapfault.xsd`에서 제공되는 XML 스키마를 사용하십시오. 여기서 `usshome`은 **USSHOME** 시스템 초기화 매개변수의 값입니다.

SOAP 노드

SOAP 노드는 SOAP 메시지에 대해 실행되는 처리 로직입니다.

SOAP 노드는 다음 조작을 수행할 수 있습니다.

- SOAP 메시지 전송
- SOAP 메시지 수신
- SOAP 메시지 처리
- SOAP 메시지 릴레이

SOAP 노드는 다음 유형 중 한 가지입니다.

SOAP 발신인

SOAP 메시지를 전송하는 SOAP 노드입니다.

SOAP 수신인

SOAP 메시지를 승인하는 SOAP 노드입니다.

초기 SOAP 발신인

SOAP 메시지 경로의 시작점에서 SOAP 메시지를 작성하는 SOAP 발신인입니다.

SOAP 중개자

SOAP 중개자는 SOAP 수신인이자 SOAP 발신인이며 SOAP 메시지 내에서 대상으로 지정될 수 있습니다. SOAP 중개자는 대상으로 지정된 SOAP 헤더 블록을 처리하며 최종 SOAP 수신인으로 SOAP 메시지를 전송합니다.

최종 SOAP 수신인

SOAP 메시지의 최종 목적지인 SOAP 수신인입니다. SOAP 본문의 콘텐츠와 대상으로 지정된 모든 SOAP 헤더 블록을 처리합니다. 경우에 따라 SOAP 메시지가 최종 SOAP 수신인에 도달하지 않을 수 있습니다. SOAP 중개자에 문제점이 있는 경우를 예로 들 수 있습니다.

SOAP 메시지 경로

SOAP 메시지 경로는 초기 SOAP 발신인, 0개 이상의 SOAP 중개자, 최종 SOAP 수신인을 포함하여 단일 SOAP 메시지가 전달되는 SOAP 노드의 세트입니다.

가장 단순한 상황에서는 두 노드 간에 SOAP 메시지가 전송됩니다(즉, SOAP 발신인로부터 SOAP 수신인으로). 그러나 보다 복잡한 상황에서는 SOAP 중개자 노드에서 메시지를 처리할 수 있습니다. 이 노드에서는 SOAP 메시지를 수신한 후 다음 노드로 보냅니다. 24 페이지의 그림 5는 SOAP 메시지가 해당 루트에서 두 SOAP 중개자 노드를 통과하여 초기 SOAP 발신인 노드에서 최종 SOAP 수신인 노드로 전송되는 SOAP 메시지 경로 예제를 표시합니다.

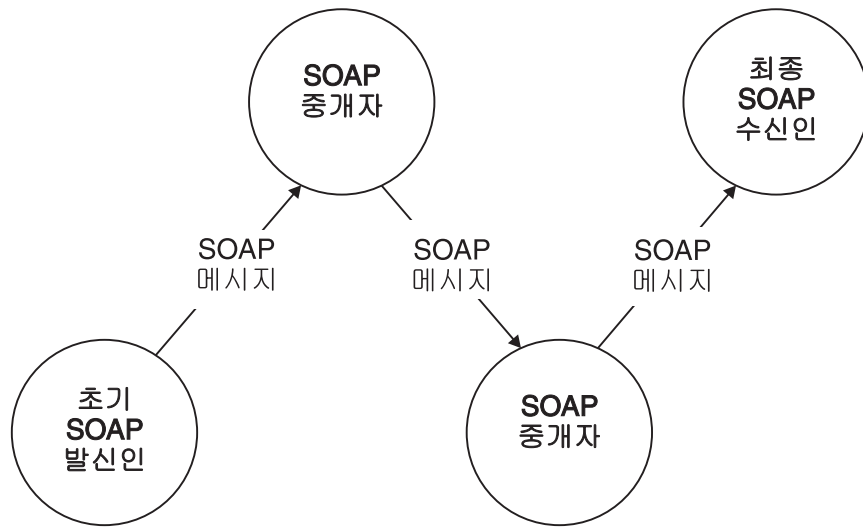


그림 5. SOAP 메시지 경로 예제

SOAP 중개자는 SOAP 수신인이자 SOAP 발신인입니다. SOAP 중개자는 SOAP 메시지에서 헤더 블록을 처리할 수 있으며(경우에 따라서는 반드시 처리해야 함) SOAP 메시지를 최종 수신인으로 보냅니다.

최종 SOAP 수신인은 SOAP 메시지의 최종 대상입니다. 헤더 블록을 처리할 뿐 아니라 SOAP 본문을 처리합니다. 경우에 따라 SOAP 메시지가 최종 SOAP 수신인에 도달하지 않을 수 있습니다. SOAP 중개자에 문제점이 있는 경우를 예로 들 수 있습니다.

제 4 장 CICS가 SOAP 웹 서비스를 지원하는 방법

CICS는 웹 서비스 환경에서 CICS 애플리케이션의 배치에 대한 두 개의 다른 접근법을 지원합니다. 하나의 접근법은 최소량의 프로그래밍 노력으로 신속한 배치를 사용합니다. 다른 하나는 특별한 필요성에 맞추기 위해 작성하는 코드를 사용하여 웹 서비스 애플리케이션을 통해 유연성과 제어를 완료합니다. 둘 다의 접근법은 웹 서비스 요청 및 응답에서 작동하는 하나 이상의 파이프라인 및 메시지 핸들러 프로그램으로 구성되는 인프라에 의해 보장됩니다.

웹 서비스 환경에서 CICS 애플리케이션을 배치할 때, 다음 옵션으로부터 선택할 수 있습니다.

- CICS 웹 서비스 지원을 사용하여 최소량의 프로그래밍 노력으로 애플리케이션을 배치하는 데 도움을 주십시오.

예를 들어, 기존 애플리케이션을 웹 서비스로 노출하려면, 고급 언어 데이터 구조로 시작하고 웹 서비스 설명을 생성할 수 있습니다. 그렇지 않으면, 기존 웹 서비스와 통신하려는 경우, 해당 웹 서비스 설명으로 시작하고 사용자의 프로그램에 사용할 수 있는 고급 언어 구조를 생성할 수 있습니다.

CICS 웹 서비스 지원은 또한 사용자의 애플리케이션을 배치해야 하는 CICS 지원을 생성합니다. 그리고 애플리케이션이 실행될 때, CICS는 사용자의 애플리케이션 데이터를 결과물의 SOAP 메시지로 변형하고, 입력에서 애플리케이션 데이터로 SOAP 메시지를 다시 변형시킵니다.

- 서비스 요청자와 제공자 사이에 플로우하는 애플리케이션 데이터와 메시지 사이에 맵핑하기 위한 자체 코드를 작성하여 데이터의 처리에 대한 완전한 제어를 수행하십시오.

예를 들어, 웹 서비스 인프라 내의 비SOAP 메시지를 사용하려는 경우, 애플리케이션이 사용하는 형식 및 메시지 형식 사이에 변환하기 위한 자체 코드를 작성할 수 있습니다.

어느 접근을 따르든지, 사용자의 요청 및 응답 메시지에 추가적 처리를 수행하기 위해 사용자의 자신의 메시지 핸들러를 사용할 수 있거나, SOAP 메시지를 처리할 수 있도록 특히 설계되는 CICS-제공된 메시지 핸들러를 사용합니다.

메시지 핸들러 및 파이프라인

메시지 핸들러는 웹 서비스 요청 및 응답의 자체 처리를 수행할 수 있는 프로그램입니다. 파이프라인은 순서대로 실행되는 메시지 핸들러 세트입니다.

파이프라인의 조작에 두 개의 고유한 단계가 있습니다.

1. CICS가 파이프라인에서 순서대로 각 핸들러를 호출하는 동안 요청 단계. CICS에 제어를 리턴하기 전에 각 메시지 핸들러는 요청을 처리할 수 있습니다.
2. 이는 응답 단계가 이어지며, 그 동안 CICS가 차례로 각 핸들러를 호출하지만, 거꾸로 된 순서로 호출합니다. 즉, 요청 단계에서 처음 호출되는 메시지 핸들러는 응답 단계에서 마지막으로 호출됩니다. 각 메시지 핸들러는 이 단계 동안 응답을 처리할 수 있습니다.

모든 요청이 응답에 따르지 않는 않습니다. 일부 애플리케이션은 서비스 요청자에서 제공자로 단방향 메시지 플로우를 사용합니다. 이 경우, 처리되는 메시지가 없더라도 각 핸들러는 응답 단계 동안 차례로 호출됩니다.

그림 6에서는 세 개의 메시지 핸들러의 파이프라인을 표시합니다.

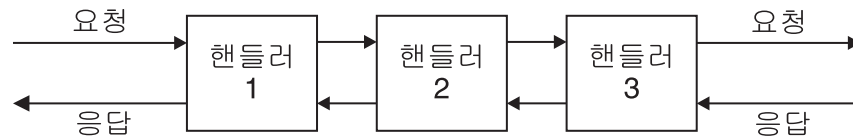


그림 6. 일반 CICS 파이프라인

이 예제에서, 핸들러는 다음 순서로 실행됩니다.

요청 단계에서

1. 핸들러 1
2. 핸들러 2
3. 핸들러 3

응답 단계에서

1. 핸들러 3
2. 핸들러 2
3. 핸들러 1

서비스 제공자에서 단계 사이의 상태 전이는 일반적으로 파이프라인의 마지막 핸들러(터미널 핸들러라고 함)에서 발생합니다. 이는 요청을 흡수하고 응답을 생성합니다. 서비스 요청자에서 상태 전이는 요청이 서비스 제공자에서 처리될 때 발생합니다. 그러나, 요청 단계의 메시지 핸들러는 응답 단계에 즉각적 상태 전이를 강제 실행할 수 있고 즉각적 상태 전이는 CICS가 오류를 감지하면 발생합니다.

메시지 핸들러는 메시지를 수정할 수 있거나 이를 변경하지 않은 채로 남겨둘 수 있습니다. 예를 들어, 다음과 같습니다.

- 암호화 및 복호화를 수행하는 메시지 핸들러는 입력 시 암호화된 메시지를 수신하고 다음 핸들러에 복호화된 메시지를 전달합니다. 출력 시, 그 반대를 수행합니다. 일반 텍스트 메시지를 수신하고 다음 핸들러에 암호화된 버전을 전달합니다.
- 로깅을 수행하는 메시지 핸들러는 메시지를 검사하고 해당 메시지로부터 로그로 관련 정보를 복사합니다. 다음 핸들러에 전달되는 메시지는 변경되지 않습니다.

중요사항: CICS TS에 대한 SOAP 기능에 익숙하면 CICS의 이 릴리스에 있는 파이프라인의 구조가 기능에 사용되는 것과 동일하지 않다는 것을 인식해야 합니다.

전송 관련 핸들러

CICS는 웹 서비스 요청자 및 제공자 사이의 두 개의 전송 메커니즘 사용을 지원합니다. 일부 경우, 사용 중인 전송 메커니즘에 따라 호출되는 다른 메시지 핸들러가 필요할 수 있습니다. 예를 들어, 외부 네트워크에서 통신하기 위해 HTTP 전송을 사용 중일 때 일부 메시지의 암호화를 수행하는 메시지 핸들러를 포함하려고 할 수 있습니다. 그러나, 암호화는 보안 내부 네트워크에서 MQ 전송을 사용 중일 때 필요하지 않을 수 있습니다.

이를 지원하기 위해, 특정 전송(HTTP 또는 MQ)이 사용 중일 때에만 호출되는 핸들러를 지정하기 위해 사용자 파이프라인을 구성할 수 있습니다. 서비스 제공자의 경우, 더 특정적일 수 있고 특별히 이름 지정된 자원(HTTP 전송의 경우 TCPIP SERVICE, MQ 전송의 경우 QUEUE)이 사용 중일 때에만 호출되는 핸들러를 지정할 수 있습니다.

이는 그림 7에서 설명됩니다.

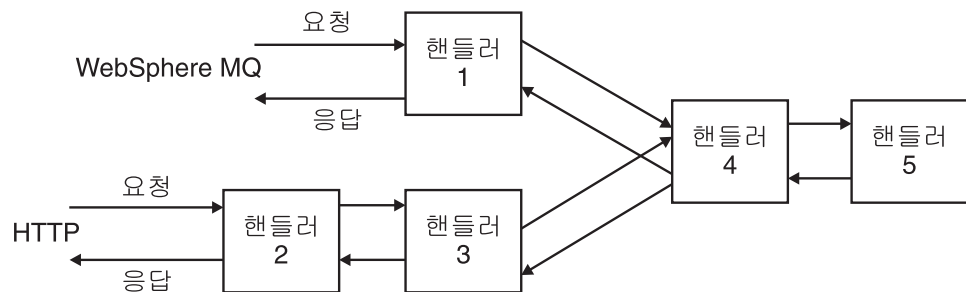


그림 7. 전송 관련 핸들러를 가진 파이프라인

이 예제에서, 이는 서비스 제공자에 적용됩니다.

- 핸들러 1은 MQ 전송을 사용하는 메시지를 위해 호출됩니다.
- 핸들러 2와 3은 HTTP 전송을 사용하는 메시지를 위해 호출됩니다.
- 핸들러 4와 5는 모든 메시지를 위해 호출됩니다.

- 핸들러 5는 터미널 핸들러입니다.

플로우 인터럽트

요청 처리 동안, 메시지 핸들러는 다음 핸들러에 메시지를 전달하지 않기로 결정할 수 있지만, 대신 응답을 생성할 수 있습니다. 메시지의 일반 처리가 인터럽트되고 파이프라인의 일부 핸들러가 호출되지 않습니다. 예를 들어, 그림 8의 핸들러 2가 보안 검사를 수행하는 것에 책임이 있다고 가정하십시오.

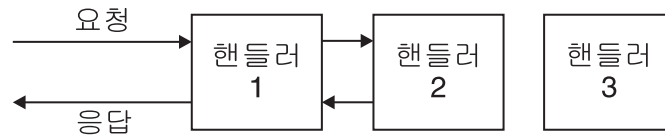


그림 8. 파이프라인 플로우 인터럽트

요청이 올바른 보안 신임 정보를 감당하지 못하는 경우, 요청을 핸들러 3에 전달하는 대신, 핸들러 2가 요청을 억제하고 적절한 응답을 구성합니다. 파이프라인은 현재 응답 단계에 있고 핸들러 2가 CICS에 제어를 리턴할 때 호출되는 다음 핸들러는 핸들러 1이고 핸들러 3은 함께 무시됩니다.

이 방법으로 일반 메시지 플로우를 인터럽트하는 핸들러는 메시지의 기원자가 응답을 예상하는 경우에 수행해야 합니다. 예를 들어, 애플리케이션이 서비스 요청자로부터 제공자로 단방향 메시지 플로우를 사용할 때 핸들러가 응답을 생성하지 않아야 합니다.

서비스 제공자 파이프라인

서비스 제공자 파이프라인에서 CICS는 대상 애플리케이션 프로그램에 파이프라인을 통해 전달되는 요청을 수신합니다. 애플리케이션으로부터의 응답은 동일한 파이프라인을 통해 서비스 요청자에게 리턴됩니다.

CICS가 서비스 제공자의 역할에 있을 때, 이는 다음 조작을 수행합니다.

1. 서비스 요청자로부터 요청을 수신하십시오.
2. 요청을 검사하고 대상 애플리케이션 프로그램과 관련된 콘텐츠를 추출하십시오.
3. 요청으로부터 추출된 데이터를 전달하여 애플리케이션 프로그램을 호출하십시오.
4. 애플리케이션 프로그램이 제어를 리턴할 때, 애플리케이션 프로그램이 리턴하는 데이터를 사용하여 응답을 구성하십시오.
5. 서비스 요청자에 응답을 전송하십시오.

그림 9는 서비스 제공자 설정에서 세 개의 메시지 핸들러 중 파이프라인을 설명합니다.

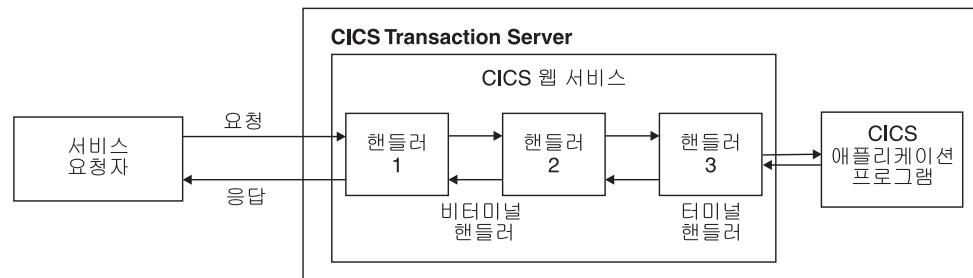


그림 9. 서비스 제공자 파이프라인

1. CICS는 서비스 요청자로부터 요청을 수신합니다. 이는 메시지 핸들러 1에 요청을 전달합니다.
2. 메시지 핸들러 1은 일부 처리를 수행하고 요청을 핸들러 2를 전달합니다(더 정확히 말하면, 이는 파이프라인을 관리하는 CICS에 제어를 리턴합니다. 그런 다음, CICS가 다음 메시지 핸들러에 제어를 전달합니다).
3. 메시지 핸들러 2는 핸들러 1에서 요청을 수신하고 일부 처리를 수행하며 핸들러 3에 요청을 전달합니다.
4. 메시지 핸들러 3은 파이프라인의 터미널 핸들러입니다. 이는 요청에서 정보를 사용하여 애플리케이션 프로그램을 호출합니다. 그런 다음, 이는 애플리케이션 프로그램의 결과물을 사용하여 응답을 생성하고, 이는 핸들러 2로 다시 전달합니다.
5. 메시지 핸들러 2는 핸들러 3에서 응답을 수신하고 일부 처리를 수행하며 핸들러 1에 이를 전달합니다.
6. 메시지 핸들러 1은 핸들러 2에서 응답을 수신하고 일부 처리를 수행하며 서비스 요청자에 응답을 리턴합니다.

서비스 요청자 파이프라인

서비스 요청자 파이프라인에서 애플리케이션 프로그램이 요청을 작성하고, 이는 파이프라인을 통해 서비스 제공자로 전달됩니다. 서비스 제공자로부터의 응답은 샘플 파이프라인을 통해 애플리케이션 프로그램으로 리턴됩니다.

CICS가 서비스 요청자의 역할에 있을 때, 이는 다음 조작을 수행합니다.

1. 애플리케이션 프로그램이 제공하는 데이터를 사용하여 요청을 구성하십시오.
2. 요청을 서비스 제공자에 전송하십시오.
3. 서비스 제공자로부터 응답을 수신하십시오.
4. 응답을 검사하고 원래 애플리케이션 프로그램과 관련된 콘텐츠를 추출하십시오.

5. 애플리케이션 프로그램으로 제어를 리턴하십시오.

그림 10은 서비스 요청자 설정에서 세 개의 메시지 핸들러 중 파이프라인을 설명합니다.

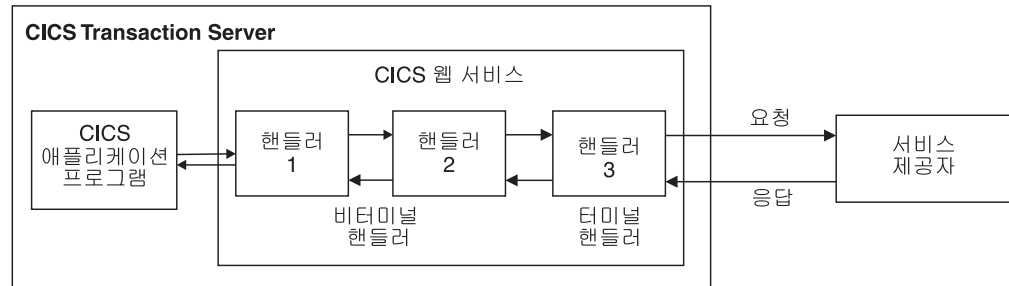


그림 10. 서비스 요청자 파이프라인

1. 애플리케이션 프로그램은 요청을 작성합니다.
2. 메시지 핸들러 1은 애플리케이션 프로그램에서 요청을 수신하고 일부 처리를 수행하며 핸들러 2로 요청을 전달합니다(더 정확히 말하면, 파이프라인을 관리하는 CICS에 제어를 리턴합니다. 그런 다음, CICS가 다음 메시지 핸들러에 제어를 전달합니다).
3. 메시지 핸들러 2는 핸들러 1에서 요청을 수신하고 일부 처리를 수행하며 핸들러 3에 요청을 전달합니다.
4. 메시지 핸들러 3은 핸들러 2에서 요청을 수신하고 일부 처리를 수행하며 서비스 제공자로 요청을 전달합니다.
5. 메시지 핸들러 3은 서비스 제공자에서 응답을 수신하고 일부 처리를 수행하며 핸들러 2에 이를 전달합니다.
6. 메시지 핸들러 2는 핸들러 3에서 응답을 수신하고 일부 처리를 수행하며 핸들러 1에 이를 전달합니다.
7. 메시지 핸들러 1은 핸들러 2에서 응답을 수신하고 일부 처리를 수행하며 애플리케이션 프로그램에 응답을 리턴합니다.

CICS 파이프라인 및 SOAP

각 메시지 핸들러에서 수행될 수 있는 처리에 제한이 거의 없다는 점에서 CICS가 웹 서비스 요청 및 응답을 처리하기 위해 사용하는 파이프라인은 일반적입니다. 그러나, 많은 웹 서비스 애플리케이션이 SOAP 메시지를 사용하고 해당 메시지의 처리가 SOAP 스펙을 따라야 합니다. 따라서, CICS에서는 SOAP 노드로 파이프라인을 구성할 수 있도록 하는 특수 SOAP 메시지 핸들러 프로그램을 제공합니다.

- 파이프라인은 서비스 요청자 또는 서비스 제공자로 사용하기 위해 구성될 수 있습니다.

- 서비스 요청자 파이프라인은 요청에 대한 초기 SOAP 발신인이고 응답에 대한 궁극적인 SOAP 수신인입니다
- 서비스 제공자 파이프라인은 요청자에 대한 궁극적 SOAP 수신인이고 응답에 대한 초기 SOAP 발신인입니다

SOAP 매개자로서 기능하도록 CICS 파이프라인을 구성할 수 없습니다.

- 서비스 요청자 파이프라인은 SOAP 1.1 또는 SOAP 1.2를 지원하도록 구성될 수 있지만, 둘 다를 지원하도록 구성될 수는 없습니다. 그러나, 서비스 제공자 파이프라인은 SOAP 1.1 및 SOAP 1.2 모두를 지원하도록 구성될 수 있습니다. CICS 시스템에서 일부는 SOAP 1.1 또는 SOAP 1.2를 지원하고 일부는 둘 다를 지원하는 많은 파이프라인이 있을 수 있습니다.
- 둘 이상의 SOAP 메시지 핸들러를 가지도록 CICS 파이프라인을 구성할 수 있습니다.
- CICS 제공 SOAP 메시지 핸들러는 하나 이상의 사용자 작성 헤더 핸들링 루틴을 호출하도록 구성될 수 있습니다.
- CICS 제공 SOAP 메시지 핸들러는 WS-I 기본 프로파일 버전 1.1의 규제 준수의 몇몇 양상을 실행하고 SOAP 메시지에서 특정 헤더의 존재를 실행하도록 구성될 수 있습니다.

SOAP 메시지 핸들러 및 해당 헤더 핸들링 루틴은 파이프라인 구성 파일에 지정됩니다.

SOAP 메시지 및 애플리케이션 데이터 구조

많은 경우, CICS 웹 서비스 지원은 애플리케이션 프로그램에서 사용되는 상위 레벨 데이터 구조와 SOAP 메시지의 <Body> 요소 콘텐츠 사이에 데이터를 전송하기 위한 코드를 생성할 수 있습니다. 이러한 경우, 사용자 애플리케이션 프로그램을 작성할 때, SOAP 본문을 구문 분석하거나 구성할 필요가 없습니다. CICS는 사용자를 위해 이를 수행합니다.

데이터를 변환하기 위해 CICS에는 런타임 시 애플리케이션 데이터 구조 및 SOAP 메시지의 형식에 대한 정보가 필요합니다. 이 정보는 두 개의 파일에 보유됩니다.

- 웹 서비스 바인딩 파일

이 파일은 애플리케이션 언어 데이터 구조에서 유틸리티 프로그램 DFHLS2WS를 사용하거나 웹 서비스 설명에서 유틸리티 프로그램 DFHWS2LS를 사용하여 CICS 웹 서비스 지원에 의해 생성됩니다. CICS는 바인딩 파일을 사용하여 웹 서비스 애플리케이션에 의해 사용되는 자원을 생성하고 애플리케이션의 데이터 구조와 SOAP 메시지 사이에 맵핑을 수행하십시오.

- 웹 서비스 설명

이는 기존 웹 서비스 설명일 수 있거나 유틸리티 프로그램 DFHLS2WS를 사용하여 애플리케이션 언어 데이터 구조로부터 생성될 수 있습니다. CICS는 웹 서비스 설명을 사용하여 SOAP 메시지의 전체 유효성 검증을 수행합니다.

그림 11에서는 이러한 파일이 서비스 제공자에서 사용되는 위치를 표시합니다.

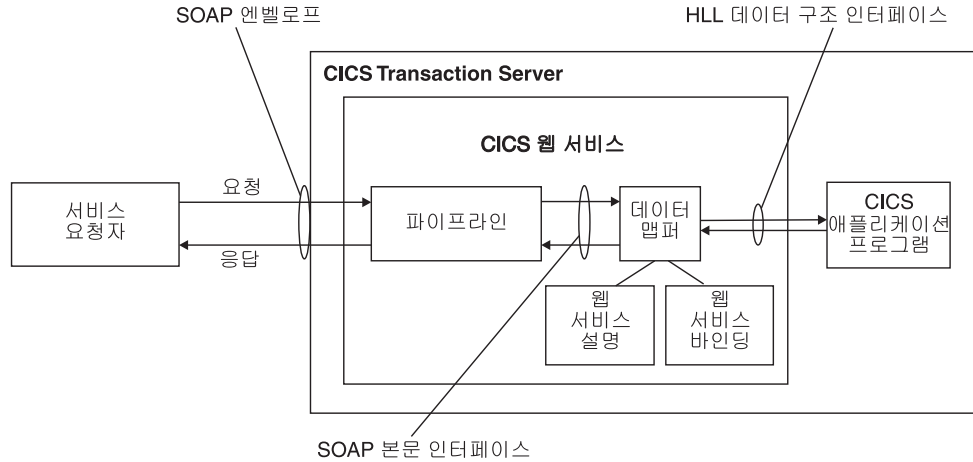


그림 11. 서비스 제공자의 애플리케이션 데이터 구조로 SOAP 본문 맵핑

파이프라인의 메시지 핸들러(일반적으로, CICS 제공 SOAP 메시지 핸들러)는 인바운드 요청으로부터 SOAP 엔벨로프를 제거하고 SOAP 본문을 데이터 매퍼 기능으로 전달합니다. 이는 웹 서비스 바인딩 파일을 사용하여 애플리케이션의 데이터 구조에 SOAP 본문의 콘텐츠를 맵핑하십시오. SOAP 메시지의 전체 유효성 검증이 활성화되면 SOAP 본문이 웹 서비스 설명에 대해 유효성 검증됩니다. 아웃바운드 응답이 있는 경우, 프로세스가 반대로 됩니다.

33 페이지의 그림 12에서는 이러한 파일이 서비스 요청자에서 사용되는 위치를 표시합니다.

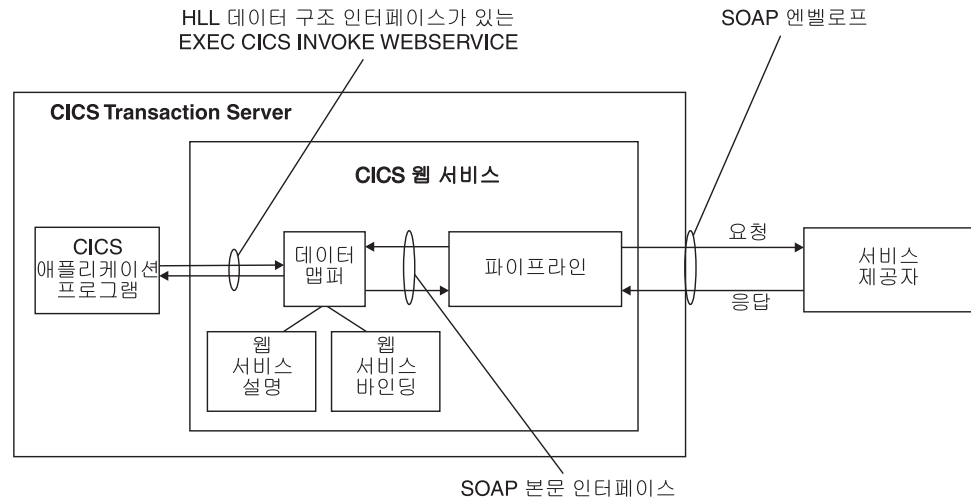


그림 12. 서비스 요청자의 애플리케이션 데이터 구조로 SOAP 본문 맵핑

아웃바운드 요청의 경우, 데이터 매퍼 기능은 웹 서비스 바인딩 파일로부터 정보를 사용하여 애플리케이션의 데이터 구조에서 SOAP 본문을 구성합니다. 파이프라인의 메시지 핸들러(일반적으로, CICS 제공 SOAP 메시지 핸들러)는 SOAP 엔벨로프를 추가합니다. 인바운드 응답이 있으면 프로세스가 반대로 됩니다. SOAP 메시지의 전체 유효성 검증이 활성화되면 인바운드 SOAP 본문이 웹 서비스 설명에 대해 유효성 검증됩니다.

둘 다의 경우에서, 특별한 CICS 애플리케이션 프로그램이 웹 서비스 설정에서 작동할 수 있도록 허용하는 실행 환경이 세 개의 오브젝트에 의해 정의됩니다. 이는 파이프라인, 웹 서비스 바인딩 파일 및 웹 서비스 설명입니다. 세 개의 오브젝트가 WEBSERVICE 자원 정의의 속성으로 CICS에 정의됩니다.

SOAP 메시지를 사용 중이라고 해도 CICS 웹 서비스 지원이 생성하는 변환을 사용할 수 없는 일부 상황이 있습니다.

- 동일한 데이터가 SOAP 메시지 및 고급 언어로 표시될 수 없을 때.

CICS가 지원하는 모든 고급 언어 및 XML 스키마는 다양한 다른 데이터 유형을 지원합니다. 그러나, 고급 언어에서 사용되는 데이터 유형과 XML 스키마에서 사용되는 데이터 유형 사이에 일대일 응답이 없고 데이터가 하나로 표시될 수 있지만, 다른 것에는 표시될 수 없는 경우가 있습니다. 이 상황에서 다음 중 하나를 고려해야 합니다.

- 애플리케이션 데이터 구조를 변경하십시오. 이는 애플리케이션 프로그램 자체에 대한 변경사항을 포함할 수 있으므로 실행할 수 없습니다.
- CICS가 SOAP 메시지 본문으로 변환할 수 있는 양식으로 애플리케이션 데이터를 변환하는 래퍼 프로그램을 구성하십시오. 이를 수행하는 경우, 애플리케이션 프

로그래를 변경하지 않을 수 있습니다. 이 경우, CICS 웹 서비스 지원은 랩퍼 프로그램과 직접적으로 상호작용하고 애플리케이션 프로그램과 간접적으로만 상호작용합니다.

- 애플리케이션 프로그램이 CICS 웹 서비스 지원에 의해 지원되지 않는 언어에 있을 때.

이 상황에서, 다음 중 하나를 고려해야 합니다.

- CICS 웹 서비스 지원은 자원(COBOL, PL/I, C 또는 C++)을 수행하는 언어 중 하나로 작성되는 랩퍼 프로그램을 구성하십시오.
- CICS 웹 서비스 지원을 사용하는 대신 SOAP 메시지와 애플리케이션 프로그램의 데이터 구조 사이에 맵핑을 수행하도록 사용자 자체 프로그램을 쓰십시오.

WSDL 및 애플리케이션 데이터 구조

웹 서비스 설명에는 서비스가 사용하는 입력 및 출력 메시지에 대한 요약 표현이 포함됩니다. CICS는 웹 서비스 설명을 사용하여 애플리케이션 프로그램이 사용하는 데이터 구조를 구성합니다. 런타임 시 CICS는 애플리케이션 데이터 구조와 메시지 사이에 맵핑을 수행합니다.

웹 서비스에 대한 설명이 다른 것들 사이에 포함됩니다.

- 하나 이상의 조작
- 각 조작에 대한 입력 메시지 및 선택적 출력 메시지
- 각 메시지에 대해 XML 데이터 유형으로 정의되는 메시지 구조. 메시지에 사용되는 복잡한 데이터 유형은 웹 서비스 설명 내의 <types> 요소에 포함되는 XML 스키마에 정의됩니다. 단순 메시지는 <types> 요소를 사용하지 않고 설명될 수 있습니다.

WSDL에는 조작의 요약 정의 및 연관된 메시지가 포함됩니다. 이는 애플리케이션 프로그램에서 직접적으로 사용될 수 없습니다. 조작을 구현하려면 서비스 제공자는 다음을 수행해야 합니다.

- 메시지의 구조를 이해하기 위해 WSDL을 구문 분석해야 합니다
- 각 입력 메시지를 구문 분석하고 출력 메시지를 구성해야 합니다.
- 입/출력 메시지의 콘텐츠와 애플리케이션 프로그램에 사용되는 데이터 구조 사이의 맵핑을 수행해야 합니다.

서비스 요청자는 조작을 호출하기 위해 동일한 작업을 수행해야 합니다.

CICS 웹 서비스 지원을 사용할 때, 많은 것이 사용자를 위해 수행되고, WSDL 또는 입출력 메시지가 구성되는 방법에 대한 상세한 이해 없이 사용자의 애플리케이션 프로그램을 쓸 수 있습니다.

CICS 웹 서비스 지원은 두 개의 유틸리티 프로그램으로 구성됩니다.

DFHWS2LS

이 유틸리티 프로그램은 웹 서비스 설명을 출발점으로 취합니다. 이는 메시지의 설명 및 해당 메시지에서 사용되는 데이터 유형을 사용하여 애플리케이션 프로그램에서 사용할 수 있는 고급 언어 데이터 구조를 구성합니다.

DFHLS2WS

이 유틸리티 프로그램은 시작점으로 고급 언어 데이터 구조를 사용합니다. 이는 구조를 사용하여 메시지에 대한 설명을 포함하는 웹 서비스 설명 및 언어 구조에서 파생된 해당 메시지에서 사용되는 데이터 유형을 구성합니다.

유틸리티 프로그램 둘 다 런타임 시 CICS가 애플리케이션 프로그램의 데이터 구조와 SOAP 메시지 사이에 매핑을 수행하기 위해 사용하는 웹 서비스 바인딩 파일을 생성합니다.

WSDL 맵핑에 대한 COBOL 예제

이 예제에서는 CICS 웹 서비스 지원이 생성하는 웹 서비스 설명에 COBOL 프로그램에서 사용되는 데이터 구조가 표시되는 방법을 보여줍니다.

그림 13에서는 단순 COBOL 데이터 구조가 표시됩니다.

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

그림 13. WSDL에서 정의된 입력 메시지의 COBOL 레코드 정의

웹 서비스 설명의 해당 단편에 있는 핵심요소가 36 페이지의 그림 14에 표시됩니다.

```

<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>

```

그림 14. COBOL 데이터 구조로부터 파생되는 WSDL 단편

WSDL 및 메시지 교환 패턴

WSDL 2.0 문서에는 웹 서비스 요청자와 웹 서비스 제공자 간에 SOAP 1.2 메시지가 교환되는 방식을 정의하는 메시지 교환 패턴(MEP)이 포함됩니다.

CICS는 서비스 제공자 및 서비스 요청자 애플리케이션에 대해 모두 *WSDL 2.0* 파트 2: 부록 스펙과 *WSDL 2.0* 파트 2: 추가 MEP 스펙에 정의된 여덟 가지 메시지 교환 패턴 중 네 가지를 지원합니다. 지원되는 MEP는 다음과 같습니다.

In-Only

요청 메시지가 웹 서비스 제공자로 전송되지만 제공자가 웹 서비스 요청자에 대한 응답 유형을 보낼 수 없습니다.

- 제공자 모드의 경우 CICS가 In-Only MEP를 사용하는 웹 서비스에서 요청 메시지를 수신하면 응답 메시지를 리턴하지 않습니다. DFHNORESPONSE 컨테이너는 SOAP 핸들러 채널에 배치되어 파이프라인이 응답 메시지를 보낼 수 없음을 나타냅니다.
- 요청자 모드의 경우 CICS가 웹 서비스 제공자로 요청 메시지를 보내며 응답을 대기하지 않습니다.

In-Out

요청 메시지가 웹 서비스 제공자로 전송되고 응답 메시지는 웹 서비스 요청자로 리턴됩니다. 응답 메시지는 정상 SOAP 메시지 또는 SOAP 결함입니다.

- 제공자 모드의 경우 CICS가 In-Out MEP를 사용하는 웹 서비스에서 요청 메시지를 수신할 때 요청자로 응답 메시지를 리턴합니다.
- 요청자 모드의 경우 CICS가 요청 메시지를 보내고 응답을 대기합니다. 이 응답은 정상 응답 메시지 또는 SOAP 결함 메시지입니다. CICS가 응답을 대기하는 시간은 파이프라인에서 구성되며 해당 파이프라인을 사용하는 모든 웹 서비스에 적용됩니다. CICS가 응답을 수신하기 전에 요청 제한시간이 초과되면 서비스 요청자 애플리케이션으로 오류가 리턴됩니다.

In-Optional-Out

요청 메시지가 웹 서비스 제공자로 전송되고 응답 메시지는 선택적으로 웹 서비스 요청자로 리턴됩니다. 응답이 있는 경우 정상 SOAP 메시지 또는 SOAP 결함입니다.

- 제공자 모드의 경우 SOAP 응답 메시지, SOAP 결함을 리턴할지 또는 응답을 리턴하지 않을지 여부를 런타임에 결정하며 이 결정은 서비스 제공자 애플리케이션 로직에 따라 다릅니다. CICS가 웹 서비스 요청자로 응답을 보내지 않으면 DFHNORESPONSE 컨테이너가 SOAP 핸들러 채널에 배치되어 파이프라인이 응답 메시지를 보낼 수 없음을 나타냅니다. 메시지를 보내지 않는 경우 서비스 제공자 애플리케이션이 채널에서 DFHWS-DATA 컨테이너를 삭제해야 합니다.
- 요청자 모드의 경우 CICS가 요청 메시지를 보내고 웹 서비스 요청자로부터 응답을 대기합니다. 응답이 수신되기 전에 요청 제한시간이 초과되면 CICS가 메시지가 수신되었으며 제공자가 응답을 보내지 않아도 된다고 간주합니다. CICS가 응답을 대기하는 시간은 파이프라인에서 구성되며 해당 파이프라인을 사용하는 모든 웹 서비스에 적용됩니다.

Robust In-Only

웹 서비스 제공자로 요청 메시지가 전송되며 오류가 발생하는 경우에만 웹 서비스 요청자로 응답 메시지가 리턴됩니다. 오류가 있으면 SOAP 결함 메시지가 요청자로 전송됩니다.

- 제공자 모드의 경우 파이프라인이 요청 메시지를 애플리케이션으로 전달하면 DFHNORESPONSE 컨테이너가 SOAP 핸들러 채널에 배치되어 파이프라

인이 응답 메시지를 보낼 수 없음을 나타냅니다. 파이프라인에서 오류가 발생하면 요청자로 SOAP 결함 메시지가 리턴됩니다.

- 요청자 모드의 경우 CICS가 웹 서비스 제공자로 요청 메시지를 보내며 제한시간이 초과되기 전 지정된 기간 동안 대기합니다. CICS가 응답을 대기하는 시간은 파이프라인에서 구성되며 해당 파이프라인을 사용하는 모든 웹 서비스에 적용됩니다. 제한시간이 있는 경우 CICS는 요청 메시지가 수신된 것으로 가정합니다.

WSDL 2.0의 메시지 교환 패턴에 대한 자세한 정보는 다음 W3C 스펙을 참조하십시오.

- WSDL 2.0 파트 2: 부록:
- WSDL 2.0 파트 2: 추가 MEP:

관련 개념:

426 페이지의 『메시지 교환』

웹 서비스 주소 지정(WS-Addressing)은 다음의 메시지 교환을 지원합니다. 단방향, 양방향요청-응답, 동기 요청-응답 및 비동기 요청-응답.

웹 서비스 바인딩 파일

웹 서비스 바인딩 파일에는 CICS가 입력 및 출력 메시지와 애플리케이션 데이터 구조 간에 데이터를 맵핑하기 위해 사용하는 정보가 포함됩니다.

웹 서비스 설명에는 서비스가 사용하는 입력 및 출력 메시지에 대한 요약 표현이 포함됩니다. 서비스 제공자 또는 서비스 요청자 애플리케이션이 실행되면 메시지 콘텐츠가 애플리케이션이 사용하는 데이터 구조에 맵핑되는 방식에 대한 정보가 CICS에 필요합니다. 이 정보는 웹 서비스 바인딩 파일에 저장됩니다.

웹 서비스 바인딩 파일은 다음 프로그램으로 작성됩니다.

- 언어 구조가 WSDL에서 생성되는 경우 유틸리티 프로그램 DFHWS2LS로
- WSDL이 언어 구조에서 생성되는 경우 유틸리티 프로그램 DFHLS2WS로

런타임에 CICS는 웹 서비스 바인딩 파일의 정보를 사용하여 애플리케이션 데이터 구조와 SOAP 메시지 간의 맵핑을 수행합니다. 웹 서비스 바인딩 파일은 WEBSERVICE 자원의 WSBIND 속성에서 CICS에 정의됩니다.

관련 정보:

WEBSERVICE 자원 정의

외부 표준

웹 서비스를 위한 CICS 지원은 수많은 산업 표준과 스펙에 따릅니다.

SAML

SAML은 보안 정보에 대해 설명하고 교환하기 위한 XML 기반 프레임워크입니다.

SAML(Security Assertion Markup Language)은 온라인 비즈니스 파트너 사이에 보안 정보를 설명하고 교환하기 위한 XML 기반 프레임워크입니다. 이 보안 정보는 보안 도메인 경계를 통해 작업하는 애플리케이션이 신뢰할 수 있는 휴대용 SAML 어설션의 양식으로 표현됩니다. OASIS SAML 표준은 이러한 SAML 어설션을 요청하고 작성하며 통신하고 이를 사용하기 위한 소중한 구문 및 규칙을 정의합니다.

SAML에서는 많은 문제점에 대한 솔루션을 제공합니다.

- 이는 연합된 ID로 알려진 서비스 제공자 사이에 보안 정보를 교환하기 위한 개방형 표준을 제공합니다.
- 이는 엔드투엔드 감사에 대한 수단을 제공합니다.
- 이는 사용자 역할 또는 권한 기반 정보에 대한 공통 소스를 제공합니다.

CICS는 SAMLCore1.1 및 SAMLCore2.0 표준을 지원합니다. 해당 표준에서 설명되는 프로토콜을 지원하지 않습니다. SAMLCore 1.0을 지원하지 않습니다.

SOAP 1.1 및 1.2

SOAP는 분권화된 분산 환경에 있는 정보의 교환을 위한 XML 기반 경량 프로토콜입니다.

프로토콜은 세 부분으로 구성됩니다.

- 메시지에 있는 것과 이를 처리하는 방법에 대해 설명하기 위해 프레임워크를 정의하는 엔벨로프.
- 애플리케이션 정의 데이터 유형의 인스턴스를 표시하기 위한 인코딩 규칙 세트
- 원격 프로시저 호출(RPC) 및 응답 표시 규칙

SOAP는 HTTP와 같은 다른 프로토콜과 함께 사용할 수 있습니다.

SOAP에 대한 스펙은 W3C(World Wide Web Consortium)에 의해 공개됩니다. SOAP 1.1에 대한 스펙은 <http://www.w3.org/TR/SOAP>에서 노트로서 설명됩니다. 이 스펙은 W3C에 의해 승인되지 않지만, SOAP 1.2 스펙에 대한 기초를 형성합니다.

SOAP 1.2는 W3C 권장사항이고 두 개의 파트로 공개됩니다.

- 파트 1: 메시징 프레임워크는 <http://www.w3.org/TR/soap12-part1>에서 공개됩니다.
- 파트 2: 부속물은 <http://www.w3.org/TR/soap12-part2>에서 공개됩니다.

스펙은 사용법 시나리오를 포함하여, SOAP 버전 1.2 스펙의 기능에 학습서를 제공하려고 하는 입문서도 포함합니다. 입문서는 <http://www.w3.org/TR/soap12-part0>에서 공개됩니다.

MTOM 1.0에 대한 SOAP 1.1 바인딩

MTOM 1.0에 대한 SOAP 1.1 바인딩은 SOAP 1.1로 SOAP 메시지 전송 최적화 메커니즘(MTOM) 및 XML-2진 최적화 패키징(XOP) 스펙을 사용하는 방법에 대해 설명하는 스펙입니다.

이 스펙의 목적은 이러한 기능이 SOAP 1.1과 상호적으로 사용되고 SOAP 1.2 MTOM/XOP 구현을 주로 재사용할 수 있도록 MTOM 및 XOP에 대한 최소 변경사항을 정의하는 것입니다.

MTOM 1.0에 대한 SOAP 1.1 바인딩 스펙은 <http://www.w3.org/Submission/soap11mtom10/>에서 W3C(World Wide Web Consortium)에 의한 형식적 제출로 공개됩니다.

SOAP 메시지 전송 최적화 메커니즘(MTOM)

SOAP 메시지 전송 최적화 메커니즘(MTOM)은 SOAP 메시지의 전송 및 형식을 최적화하는 방법을 개념적으로 정의하는 스펙의 관련 쌍 중 하나입니다.

MTOM은 다음을 정의합니다.

1. 요약 용어로 SOAP 메시지에서 base64binary 데이터의 전송을 최적화하는 방법
2. XOP를 사용하여 바인딩 독립 방법으로 SOAP 메시지의 최적화된 MIME 다중 부분 직렬화를 구현하는 방법

MTOM의 구현은 관련 XML-2진 최적화 패키징(XOP) 스펙에 의존합니다. 이러한 두 개의 스펙이 밀접하게 링크된 것처럼 보통 MTOM/XOP로서 언급됩니다.

스펙은 <http://www.w3.org/TR/soap12-mtom/>에서 W3C 권장사항으로 W3C(World Wide Web Consortium)에 의해 공개됩니다.

Web Services Addressing 1.0

Web Services Addressing 1.0(WS-Addressing)은 웹 서비스 사이에 메시징 정보를 전달하기 위한 전송 독립 메커니즘을 정의하는 스펙입니다.

WS-Addressing 스펙은 전송 프로토콜과 메시징 시스템에 의해 일반적으로 제공되는 정보를 정상화하는 두 개의 구성(메시지 주소 지정 특성 및 엔드포인트 참조)을 정의합니다.

스펙은 W3C 권장사항으로 W3C(World Wide Web Consortium)에 의해 공개되고 세 개의 파트로 공개됩니다.

- WS-Addressing 1.0 - 코어
- WS-Addressing 1.0 - SOAP 바인딩
- WS-Addressing 1.0 - 메타데이터

CICS를 가진 WS-Addressing을 사용할 때 이러한 W3C 스펙을 따르도록 권장됩니다.

상호 운용성의 경우, CICS는 네임스페이스가 <http://schemas.xmlsoap.org/ws/2004/08/addressing>으로 설정될 때에만 W3C WS-Addressing 제출 스펙을 묵인합니다.

CICS API 명령은 WS-Addressing 권장사항 스펙을 따르는 MAP 및 EPR을 지원합니다. 그러나, API 명령은 WS-Addressing 제출 스펙을 따르는 MAP 및 EPR을 지원하지 않습니다.

주소 지정 컨텍스트는 권장사항 스펙의 레벨에서 모든 MAP를 유지보수합니다. 필요한 경우, 이러한 MAP는 SOAP 메시지에 적용되거나 해당 메시지에서 추출될 때 제출 스펙 레벨로 변환되거나 제출 스펙으로 변환될 수 있습니다.

Web Services Atomic Transaction 버전 1.0

Web Services Atomic Transaction 버전 1.0(또는 WS-AtomicTransaction)은 짧은 지속 기간의 트랜잭션에 대한 원자적 트랜잭션 조정 유형을 정의하는 프로토콜입니다. 이는 Web Services Coordination 버전 1.0 (또는 WS-Coordination) 스펙에서 설명된 확장 가능한 조정 프레임워크로 사용됩니다.

WS-AtomicTransaction 스펙 및 WS-Coordination 스펙은 트랜잭션 처리 시스템이 웹 서비스 환경에서 상호실행할 수 있도록 하는 단기간 트랜잭션에 대한 프로토콜을 정의합니다. WS-AtomicTransaction을 사용하는 트랜잭션에는 ACID 특성인 원자성, 일관성, 격리 및 내구성이 있습니다.

WS-AtomicTransaction에 대한 스펙은 <http://www.ibm.com/developerworks/library/specification/ws-tx/>에서 공개됩니다.

Web Services Coordination 버전 1.0

Web Services Coordination 버전 1.0(또는 WS-Coordination)은 분산 애플리케이션의 조치를 조정하는 프로토콜을 제공하기 위한 확장 가능한 프레임워크입니다. 이러한 조정 프로토콜은 분배 활동의 결과에 대한 일치 계약에 도달해야 하는 것을 포함하여 수많은 애플리케이션을 지원하기 위해 사용됩니다.

프레임워크는 애플리케이션 서비스가 기타 서비스에 활동을 전파하고 조정 프로토콜에 등록하기 위해 필요한 컨텍스트를 작성할 수 있도록 합니다. 프레임워크는 기존 트랜잭션 처리, 워크플로우 및 조정을 위한 기타 시스템이 배타적 프로토콜을 숨기고 이기종 환경에서 작동할 수 있도록 합니다.

WS-Coordination에 대한 스펙은 <http://www.ibm.com/developerworks/library/specification/ws-tx/>에서 공개됩니다.

Web Services Description Language 버전 1.1 및 2.0

웹 서비스 설명 언어(WSDL)는 문서 기반 또는 프로시저 기반 정보를 포함하는 메시지에서 작동하는 엔드포인트 세트로서 네트워크 서비스를 설명하기 위한 XML 형식입니다.

조작 및 메시지가 추상적으로 설명되고 엔드포인트를 정의하기 위해 분명한 네트워크 프로토콜 및 메시지 형식으로 바인딩됩니다. 관련된 분명한 엔드포인트는 추상적 엔드포인트(서비스)에 결합됩니다.

WSDL은 통신하기 위해 사용되는 메시지 형식 또는 네트워크 프로토콜과 상관없이 엔드포인트 및 해당 메시지의 설명을 허용하기 위해 확장 가능할 수 있습니다. WSDL 1.1 스펙은 SOAP 1.1, HTTP GET와 POST 및 MIME과 함께 WSDL을 사용하는 방법에 대해 설명하는 바인딩만을 정의합니다.

WSDL 2.0에서는 웹 서비스를 설명하기 위해 XML 형식 뿐만 아니라 모델을 제공합니다. 이는 사용자가 서비스에 의해 제공되는 추상적 기능에 대한 설명을 서비스 설명에 대한 명백한 세부사항(예: 기능이 제공되는 "방법" 및 "위치")에서 분리할 수 있도록 합니다. 이는 SOAP 1.2 및 HTTP에 대한 분명한 세부사항과 같이 설명을 위한 언어, SOAP 모듈 및 메시지 교환 패턴에 대한 확장자에 대해서도 설명합니다. WSDL 2.0 스펙은 WSDL 1.1에 있는 많은 기술적 문제와 제한사항도 해결합니다.

WSDL 1.1에 대한 스펙은 <http://www.w3.org/TR/wsdl>에서 W3C 노트로서 W3C(World Wide Web Consortium)에 의해 공개됩니다.

WSDL 2.0에 대한 스펙은 <http://www.w3.org/TR/wsdl20>에서 W3C 권장사항으로서 공개됩니다.

웹 서비스 보안: SOAP 메시지 보안

웹 서비스 보안(WSS): SOAP 메시지 보안은 메시지 무결성 및 기밀성을 제공하는 SOAP 메시징에 대한 일련의 개선사항입니다. WSS: SOAP 메시지 보안은 확장 가능하고 다양한 보안 모델 및 암호 기술을 수용할 수 있습니다.

WSS: SOAP 메시지 보안에서는 개별적으로 또는 함께 사용될 수 있는 세 개의 주요 메커니즘을 제공합니다. 다음과 같습니다.

- 메시지의 일부로서, 그리고 메시지 콘텐츠와 보안 토큰을 연관시키기 위해 보안 토큰을 전송하기 위한 기능
- 권한이 없고 발견되지 않은 수정(메시지 무결성)으로부터 메시지의 콘텐츠를 보호하기 위한 기능
- 권한이 없는 폭로(메시지 무결성)로부터 메시지의 콘텐츠를 보호하기 위한 기능.

WSS: SOAP 메시지 보안은 다양한 보안 요구사항을 충족시키기 위해 기타 웹 서비스 확장자 및 애플리케이션 스펙 프로토콜과 함께 사용될 수 있습니다.

스펙은 OASIS에 의해 공개됩니다. <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>.

웹 서비스 신뢰 언어

웹 서비스 신뢰 언어(또는 WS-Trust)는 보안 토큰을 요청하고 실행하기 위한 프레임워크를 제공하고 신뢰 관계를 중개하기 위해 웹 서비스 보안에 빌드하는 확장자를 정의합니다.

WS-Trust에서는 다음을 설명합니다.

1. 보안 토큰을 실행하고 갱신하고 유효성 검증하기 위한 메소드.
2. 신뢰 관계를 확립하고 이에 액세스하며 이를 중개하는 방법.

CICS는 OASIS WS-Trust v1.4 Standard에서 공개되는 스펙의 2012년 4월 버전을 지원합니다.

SOAP 1.2에 대한 WSDL 1.1 바인딩 확장

SOAP 1.2에 대한 WSDL 1.1 바인딩 확장은 웹 서비스 메시지가 SOAP 1.2 프로토콜에 바인딩되는 것을 표시하기 위해 필요한 바인딩 확장을 정의하는 스펙입니다.

이 스펙의 목적은 SOAP 1.1에 대한 바인딩과 비교 가능한 기능을 제공하는 것입니다.

이 스펙은 <http://www.w3.org/Submission/wsdl11soap12/>에 있는 W3C(World Wide Web Consortium)에 의한 형식적 제출 요청으로 공개됩니다.

WS-I 기본 프로파일 버전 1.1

WS-I 기본 프로파일 버전 1.1 (WS-I BP 1.1)은 해당 스펙에 대한 설명 및 개정 조항과 함께 비전매 특허의 웹 서비스 스펙 세트입니다. 이는 웹 서비스의 다른 구현 사이에 상호 운용성을 함께 승격시킵니다.

WS-I BP 1.1은 메시지의 표시 및 엔벨로프의 직렬화와 관련되는 요구사항을 개별화하고 공개된 정오표를 통합하여 기본 프로파일 버전 1.0으로부터 파생됩니다. 이러한 요구사항은 단순 SOAP 바인딩 프로파일 버전 1.0의 일부입니다.

요약하기 위해 WS-I 기본 프로파일 버전 1.0이 두 개의 공개된 프로파일로 개별적으로 분할됩니다. 다음과 같습니다.

- WS-I 기본 프로파일 버전 1.1
- WS-I 단순 SOAP 바인딩 프로파일 버전 1.0

함께, 이러한 두 개의 프로파일은 WS-I 기본 프로파일 버전 1.0을 대신합니다.

이러한 분리에 대한 이유는 기본 프로파일 1.1이 단순 SOAP 바인딩 프로파일 1.0을 포함하여 엔벨로프 직렬화를 지정하는 프로파일로 구성될 수 있도록 하는 것입니다.

WS-I BP 1.1에 대한 스펙은 웹 서비스 상호 운용성 조직(WS-I)에 의해 공개되고 <http://www.ws-i.org/Profiles/BasicProfile-1.1.html>에서 찾을 수 있습니다.

WS-I 단순 SOAP 바인딩 프로파일 버전 1.0

WS-I 단순 SOAP 바인딩 프로파일 버전 1.0(SSBP 1.0)은 상호 운용성을 촉진하는 해당 스펙에 대한 설명 및 개정 조항과 함께 비전매 특허의 웹 서비스 스펙 세트입니다.

SSBP 1.0은 메시지에서 엔벨로프 및 해당 표시에 대한 직렬화와 관련되는 WS-I 기본 프로파일 1.0 요구사항으로부터 파생됩니다.

WS-I 기본 프로파일 1.0은 두 개의 개별적으로 공개된 프로파일로 나뉩니다. 다음과 같습니다.

- WS-I 기본 프로파일 버전 1.1
- WS-I 단순 SOAP 바인딩 프로파일 버전 1.0

함께, 이러한 두 개의 프로파일은 WS-I 기본 프로파일 버전 1.0을 대신합니다.

SSBP 1.0에 대한 스펙은 웹 서비스 상호 운용성 조직(WS-I)에 의해 공개되고 <http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html>에서 찾을 수 있습니다.

XML(Extensible Markup Language) 버전 1.0

XML(*Extensible Markup Language*) 1.0은 SGML의 서브세트입니다. 해당 목적은 현재 HTML로 가능한 방법으로 일반 SGML이 웹에서 제공되고 수신되며 처리될 수 있도록 하는 것입니다.

XML은 용이한 구현 및 SGML과 HTML 모두와의 상호 운용성을 위해 디자인되었습니다.

XML 1.0에 대한 스펙 및 해당 정오표는 <http://www.w3.org/TR/REC-xml>에서 W3C 권장사항으로 W3C(World Wide Web Consortium)에 의해 공개됩니다.

XML-2진 최적화 패키징(XOP)

XML-2진 최적화 패키징(XOP)은 특정 콘텐츠 유형을 가지는 XML Infoset를 효과적으로 직렬화하는 방법을 정의하는 관련된 스펙 쌍 중 하나입니다.

XOP는 다음으로 이를 수행합니다.

1. 일부 형식으로 XML 패키징. XOP 패키지라고 합니다. 스펙은 MIME Multipart/Related에 대해 언급하지만 이 형식으로 제한하지 않습니다.
2. 해당 크기를 줄이기 위해 모든 또는 일부 base64binary 콘텐츠 재인코딩.
3. 패키지의 다른 곳에 base64binary 콘텐츠를 위치시키고 이를 참조하는 XML로 인코딩된 콘텐츠 대체.

XOP는 MTOM 스펙의 구현으로 사용되며, SOAP 메시지의 최적화를 정의합니다. 이러한 두 개의 스펙이 밀접하게 링크된 것처럼 보통 MTOM/XOP로서 언급됩니다.

스펙은 <http://www.w3.org/TR/xop10/>에서 W3C 권장사항으로 W3C(World Wide Web Consortium)에 의해 공개됩니다.

XML 암호화 구문 및 처리

XML 암호화 구문 및 처리는 XML에서 결과를 표시하고 데이터를 암호화하기 위한 프로세스를 지정합니다. 데이터는 임의의 데이터(XML 문서 포함), XML 요소 또는 XML 요소 콘텐츠일 수 있습니다. 데이터 암호화의 결과는 암호 데이터를 포함하거나 참조하는 XML 암호화 요소입니다.

XML 암호화 구문 및 처리는 W3C(World Wide Web Consortium)의 권장사항이고 <http://www.w3.org/TR/xmlenc-core/>에서 공개됩니다.

XML-서명 구문 및 처리

XML-서명 구문 및 처리는 XML 디지털 서명에 대한 처리 규칙 및 구문을 지정합니다.

서명을 포함하는 XML 내 또는 그 외의 지역에 위치하든지, XML 디지털 서명에서는 무결성, 메시지 인증 및 데이터 유형에 대한 서명자 인증 서비스를 제공합니다.

XML-서명에 대한 스펙은 <http://www.w3.org/TR/xmldsig-core/>에 있는 W3C(World Wide Web Consortium)에 의해 공개됩니다.

CICS의 웹 서비스 표준 준수

준수하는 웹 서비스를 생성하고 배치할 수 있도록 허용한다는 점에서 CICS는 지원되는 웹 서비스 표준 및 스펙을 준수합니다.

CICS가 이 준수를 실행하지 않는다는 것에 주목해야 합니다. 예를 들어, WS-I 기본 프로파일 1.1 스펙에 대한 지원의 경우 CICS를 사용하면 추가 서비스 품질(QoS)을 이 프로파일에서 요약된 상호 운용성을 중단할 수 있는 웹 서비스에 적용할 수 있습니다.

CICS의 WS-Addressing 준수 방법

CICS는 WS-Addressing 스펙의 코어와 SOAP 바인딩 파트로 컴파일됩니다. CICS는 한 예외를 가진 스펙의 메타데이터 파트로 컴파일됩니다.

CICS가 WS-Addressing 결함을 실행할 때, 스펙을 따르지 않습니다. CICS는 WS-Addressing 결함을 빌드할 때 기본 조치에 대해 메타데이터 스펙에 설명된 형식을 따르지 만, 최종 구분 기호 및 결함 이름을 포함하지 않습니다.

WSDL 1.1의 경우, 스펙에 따른 기본 조치는 다음과 같습니다.

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

그러나, CICS는 결함 이름을 생략하고 다음과 같이 기본 조치를 빌드합니다.

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

WSDL 2.0의 경우, 스펙에 따른 기본 조치는 다음과 같습니다.

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

그러나, CICS는 결함 이름을 생략하고 다음과 같이 기본 조치를 빌드합니다.

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

CICS의 WSDL 2.0 준수 방법

CICS는 조건적으로 WSDL 2.0으로 컴파일하고 지원은 다음 제한사항을 따릅니다.

필수 요구사항

- 메시지 교환 패턴(in-only, in-out, robust in-only 및 in-optional-out)만 WSDL에서 사용될 수 있습니다.
- 하나의 엔드포인트만 각 서비스에 허용됩니다.
- 최소 하나의 조작이 있어야 합니다.
- 엔드포인트는 URI로만 지정될 수 있습니다.
- SOAP 바인딩이 있어야 합니다.
- XML 스키마 유형 시스템이 사용되어야 합니다.

용인되는 측면

- 다음 HTTP 바인딩 특성이 무시됩니다.
 - whttp:location
 - whttp:header
 - whttp:transferCodingDefault
 - whttp:transferCoding
 - whttp:cookies
 - whttp:authenticationType
 - whttp:authenticationRealm
- SOAP 헤더 정보는 DFHWS2LS에 의해 무시됩니다. 그러나, 인바운드 및 아웃바운드 메시지에 대한 필수 SOAP 헤더 정보를 작성하고 처리하도록 파이프라인에 사용자 자체 메시지 핸들러를 추가할 수 있습니다.

지원되지 않는 측면

- #any 및 #other 메시지 콘텐츠 모델.
- out-only, robust-out-only, out-in 및 out-optional-in 메시지 교환 패턴.
- 엔드포인트에 대한 WS-Addressing.

- HTTP GET은 지원되지 않습니다. 이는 WSDL 문서에서 SOAP 응답 메시지 교환 패턴을 사용하여 정의됩니다. WSDL이 이 메시지 교환 패턴을 정의하는 경우, DFHWS2LS는 오류 메시지를 실행합니다.

CICS의 웹 서비스 보안 스펙 준수 방법

CICS는 조건적으로 다음 양상을 지원하여 웹 서비스 보안: SOAP 메시지 보안 및 관련 스펙을 규제 준수합니다.

웹 서비스 보안: SOAP 메시지 보안 규제 준수

보안 헤더

<wsse:Security> 헤더에서는 SOAP 액터 또는 역할의 양식으로 특정 수신인을 대상으로 하는 보안 관련 정보를 첨부하기 위한 메커니즘을 제공합니다. 이는 메시지 또는 매개체의 궁극적인 수신인이 될 수 있습니다. 다음 속성은 CICS에서 지원됩니다.

- S11:actor(매개체에 대한)
- S11:mustUnderstand
- S12:role(매개체에 대한)
- S12:mustUnderstand

보안 토큰

다음 보안 토큰은 보안 헤더에서 지원됩니다.

- 사용자 이름 및 비밀번호
- 2진 보안 토큰(X.509 인증서)
- SAML 토큰
- Kerberos 토큰

토큰 참조

보안 토큰은 일련의 청구를 전달합니다. 가끔 이러한 청구는 어디에든지 상주하고 수신 애플리케이션에 의해 액세스되어야 합니다.

<wsse:SecurityTokenReference> 요소에서는 보안 토큰을 참조하기 위한 확장 가능 메커니즘을 제공합니다. 다음 메커니즘이 지원됩니다.

- 직접 참조
- 키 ID
- 키 이름
- 임베디드 참조

서명 알고리즘

이 스펙이 XML 서명에서 빌드되므로 XML 서명 스펙에 지정된 것과 동일한 알고리즘 요구사항을 가집니다. CICS는 다음을 지원합니다.

알고리즘 유형	알고리즘	URI
요약	SHA1	http://www.w3.org/2000/09/xmldsig#sha1
서명	SHA1을 가진 DSA(유효성 검증 전용)	http://www.w3.org/2000/09/xmldsig#dsa-sha1
서명	SHA1을 가진 RSA	http://www.w3.org/2000/09/xmldsig#rsa-sha1
정규화	독점 XML 정규화(주석 제외)	http://www.w3.org/2001/10/xml-exc-c14n#

서명된 서명 파트

CICS를 사용하면 다음 SOAP 요소가 서명될 수 있습니다.

- SOAP 메시지 본문
- ID 토큰(보안 토큰 유형), 이는 확실한 ID로 사용됩니다

암호화 알고리즘

다음 데이터 암호화 알고리즘이 지원됩니다.

알고리즘	URI
3-DES(Data Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmlenc#tripledes-cbc
키 길이가 128비트인 AES(Advanced Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmlenc#aes128-cbc
키 길이가 192비트인 AES(Advanced Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmlenc#aes192-cbc
키 길이가 256비트인 AES(Advanced Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmlenc#aes256-cbc

다음 키 암호화 알고리즘이 지원됩니다.

알고리즘	URI
키 전송(공용 키 암호) RSA 버전 1.5:	http://www.w3.org/2001/04/xmlenc#rsa-1_5

암호화 메시지 파트

CICS를 사용하면 다음 SOAP 요소가 암호화될 수 있습니다.

- SOAP 본문

시간소인

<wsu:Timestamp> 요소에서는 메시지에서 보안 시맨틱의 작성 및 만기 시간을 표현하기 위한 메커니즘을 제공합니다. CICS는 인바운드 SOAP 메시지에서 웹 서비스 보안 헤더 내에 시간소인의 사용을 용인합니다.

오류 핸들링

CICS는 스펙에 나열된 응답 코드의 표준 목록을 사용하여 SOAP 결함 메시지를 생성합니다.

웹 서비스 보안: UsernameToken 프로파일 1.0의 규제 준수

이 스펙의 다음 양상이 지원됩니다.

비밀번호 유형

텍스트

토큰 참조

직접 참조

웹 서비스 보안: X.509 인증서 토큰 프로파일 1.0의 규제 준수

이 스펙의 다음 양상이 지원됩니다.

토큰 유형

- X.509 버전 3: 단일 인증서. 다음을 참조하십시오. <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf>.
- X.509 버전 3: 인증서 폐기 목록(CRL)을 포함하지 않는 X509PKIPathv1. <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf>의 내용을 참조하십시오.
- X.509 버전 3: CRL 포함 또는 포함하지 않는 PKCS7. IBM 소프트웨어 개발 키트(SDK)은 둘 다를 지원합니다.

토큰 참조

- 키 ID - 주제 키 ID
- 직접 참조
- 사용자 정의 참조 - 발급자 이름과 일련 번호

지원되지 않는 측면

다음 항목은 CICS에서 지원되지 않습니다.

- 신선미를 위한 시간소인의 유효성 검사
- 난스(nonce)
- SOAP 첨부 파일을 위한 웹 서비스 보안
- <wsse:KeyIdentifier>를 사용하여 <wsse:SecurityTokenReference>로부터 X509 인증서로의 참조
- SAML(Security Assertion Markup Language) 토큰 프로파일, WS-SecurityKerberos 토큰 프로파일 및 XrML 토큰 프로파일
- 웹 서비스 상호 운용성(WS-I) 기본 보안 프로파일

- XML 엔벨로프 디지털 서명
- XML 엔벨로프 디지털 암호화
- 디지털 서명에 대한 다음 전송 알고리즘이 지원되지 않습니다.
 - XSLT: <http://www.w3.org/TR/1999/REC-xslt-19991116>
 - SOAP 메시지 정규화. 자세한 정보는 <http://www.w3.org/TR/2003/NOTE-soap12-n11n-20031008/>의 내용을 참조하십시오
- 암호화를 위한 Diffie-Hellman 키 계약 알고리즘은 지원되지 않습니다. 자세한 정보는 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/Overview.html#sec-DHKeyValue>의 내용을 참조하십시오.
- XML 암호화 스펙에서 선택적인 암호화에 대한 다음 정규화 알고리즘은 지원되지 않습니다.
 - 정규 XML(주석 포함 또는 제외)
 - 독점 XML 정규화(주석 포함 또는 제외)
- 사용자 이름 토큰 버전 1.0 프로파일 스펙에서 요약 비밀번호 유형이 지원되지 않습니다.

CICS의 WS-Trust 준수 방법

CICS는 조건적으로 WS-Trust로 컴파일하고 지원은 다음 제한사항을 따릅니다.

지원되는 측면

- 유효성 검사 바인딩
- 하나의 토큰이 리턴되는 발행 바인딩
- 발행 바인딩의 AppliesTo

용인되는 측면

- 요청되는 참조
- 키 및 엔트로피
- 계산된 키 리턴

지원되지 않는 측면

- 여러 보안 토큰 리턴
- 헤더에서 보안 토큰 리턴
- 갱신 바인딩
- 바인딩 취소
- 교섭 및 인증 확인 확장기능
- 키 및 토큰 매개변수 확장기능
- 키 교환 토큰 바인딩

CICS의 WS-I 기본 프로파일 1.1 준수 방법

모든 *MUST* 레벨 요구사항을 고수한다는 점에서 CICS는 WS-I 기본 프로파일 1.1로 조건적으로 컴파일합니다. 그러나, CICS는 특히 UDDI 레지스트리에 대한 지원을 구현하지 않습니다. 따라서, 스펙에서 이와 관련되는 위치가 무시됩니다. 특정 스키마 요소를 매핑하는 지원에 제한사항이 있으므로 웹 서비스 지원 작업 및 연관된 런타임 환경은 이 프로파일을 완전히 따르는 것은 아닙니다.

지원되지 않는 스키마 요소의 목록은 고급 언어 및 XML 스키마 매핑을 참조하십시오.

준수 대상은 요구사항이 적용하는 아티팩트(예: SOAP 메시지, WSDL 설명) 또는 파티(예: SOAP 프로세서, 사용자)를 식별합니다. CICS가 지원하는 준수 대상은 다음과 같습니다.

MESSAGE

ENVELOPE(예: HTTP를 통한 SOAP 메시지)를 전송하는 프로토콜 요소.

ENVELOPE

soap:Envelope 요소 및 해당 콘텐츠의 직렬화.

DESCRIPTION

유형에 대한 설명, 메시지, 인터페이스와 프로토콜 및 데이터 형식 바인딩, 웹 서비스와 연관된 네트워크 액세스 위치(예: WSDL 설명).

INSTANCE

wsdl:port를 구현하는 소프트웨어.

CONSUMER

INSTANCE를 호출하는 소프트웨어.

SENDER

이와 연관된 프로토콜에 따라 메시지를 생성하는 소프트웨어

RECEIVER

이와 연관되는 프로토콜에 따라 메시지를 이용하는 소프트웨어.

제 5 장 SOAP 웹 서비스 시작하기

CICS에서 SOAP 웹 서비스로 시작하는 여러 가지 방법이 있습니다. 가장 적절한 방식은 SOAP 웹 서비스에 대해 사용자가 이미 알고 있는 정도 및 그에 대한 계획의 진전 정도에 따라 달라집니다.

이 태스크 정보

여기에는 CICS의 SOAP 웹 서비스에 대한 일부 시작점이 있습니다.

프로시저

- 예제 애플리케이션을 설치하십시오. CICS는 웹 서비스 제공자로 사용 설정될 수 있는 카탈로그 관리 애플리케이션의 예제를 제공합니다. 이 예제에는 CICS에서 애플리케이션이 최소 작업량으로 작동하게 하기 위해 필요한 모든 코드 및 자원 정의가 포함되어 있습니다. 여기에는 다수의 공통 웹 서비스 클라이언트에서 실행되는 서비스와 상호작용하기 위한 코드도 포함됩니다.

CICS의 웹 서비스를 배치할 수 있는 빠른 개념 증명 실증을 원하거나 CICS의 웹 서비스에 대해 학습하기 위한 수동 방식을 원하는 경우 예제 애플리케이션을 사용하십시오.

예제 애플리케이션은 717 페이지의 제 16 장 『CICS 카탈로그 관리자 예제 애플리케이션』에 설명되어 있습니다.

- 서비스 제공자 또는 요청자로의 애플리케이션 배치를 계획하십시오. CICS의 웹 서비스를 사용하는 방식에 대해 애플리케이션 및 관련 인프라 계획을 시작할 수 있을 정도로 이미 충분히 알고 있을 수 있습니다.

SOAP 웹 서비스 사용 계획

CICS에서 SOAP 웹 서비스 사용을 계획하려면 먼저 각 애플리케이션에 대한 이러한 질문을 고려해야 합니다.

시작하기 전에

CICS 애플리케이션을 서비스 제공자 역할로 또는 서비스 요청자 역할로 배치할 계획입니까?

웹 서비스에 대한 CICS 지원을 사용하여 연결하려고 하는 한 쌍의 애플리케이션이 있을 수 있습니다. 이 경우 한 애플리케이션은 서비스 제공자가 되고 다른 한 애플리케이션은 서비스 요청자가 됩니다.

기존 애플리케이션 프로그램을 사용할 계획입니까? 아니면 새 애플리케이션 프로그램을 작성할 계획입니까?

기존 애플리케이션이 비즈니스 로직에 대해 잘 정의된 인터페이스로 디자인된 경우 웹 서비스 설정에서 그러한 애플리케이션을 서비스 제공자 또는 서비스 요청자로 사용할 수도 있습니다. 그러나 대부분의 경우 비즈니스 로직을 웹 서비스 로직에 연결하는 랩퍼 프로그램을 작성해야 합니다.

새 애플리케이션을 작성할 계획인 경우 비즈니스 로직이 웹 서비스 조직과 분리된 상태로 유지되도록 해야 하고 다시 한번 이 분리를 제공하기 위해 결국 랩퍼 프로그램을 작성해야 합니다. 그러나 애플리케이션이 웹 서비스를 고려한 상태에서 디자인되는 경우 랩퍼 작성이 더 단순할 수 있습니다.

SOAP 메시지를 사용할 예정입니까?

SOAP는 웹 서비스 아키텍처에 대한 기초이고 CICS에서 제공되는 대부분의 지원에서는 SOAP을 사용할 것으로 가정합니다. 그러나 다른 메시지 형식을 사용해야 하는 상황이 있을 수도 있습니다. 예를 들어, CICS 웹 서비스 인프라와 함께 배치하려는 자체 메시지 형식을 개발했을 수도 있습니다. 이러한 작업은 CICS에서 수행할 수 있지만 CICS가 제공하는 기능 중 일부(예: 웹 서비스 지원 및 SOAP 메시지 핸들러)를 사용하지 못할 수 있습니다.

SOAP을 사용하지 않기로 결정한 경우 애플리케이션 프로그램이 인바운드 메시지 구문 분석 및 아웃바운드 메시지 구성을 담당합니다.

데이터 구조와 SOAP 메시지 간 맵핑을 생성하기 위해 CICS 웹 서비스 지원을 사용할 계획입니까?

지원은 약간의 추가 프로그래밍 작업 또는 추가 프로그래밍 작업 없이 웹 서비스 설정에 대한 여러 애플리케이션의 빠른 배치를 제공합니다. 그리고 추가 프로그래밍이 필요한 경우 이는 보통 단순한 작업이며 기존 비즈니스 로직을 변경하지 않고도 수행할 수 있습니다.

그러나 웹 서비스 지원을 사용하지 않고 처리하는 것이 나은 경우도 있습니다. 예를 들어, 데이터 구조를 SOAP 메시지로 맵핑하는 기존 코드가 있는 경우 웹 서비스 지원을 사용하여 애플리케이션을 리엔지니어링하는 것은 별다른 장점이 없습니다.

CICS 웹 서비스 지원이 가장 일반적인 데이터 유형 및 구조를 지원하기는 하지만 지원되지 않는 항목도 있습니다. 이 상황에서는 해당 언어에 대해 지원되지 않는 데이터 유형 및 구조의 목록을 확인하고 지원이 지원할 수 있는 형식으로 애플리케이션 데이터를 맵핑하는 프로그램 계층 제공을 고려해야 합니다. 이러한 작업이 불가능한 경우 메시지를 직접 구문 분석해야 합니다. 지원이 지원할 수 있고 지원할 수 없는 항목에 대한 세부사항은 고급 언어 및 XML 스키마 맵핑을 참조하십시오.

CICS 웹 서비스 지원을 사용하지 않기로 결정한 경우 z System z®용 Rational® Developer과 같은 도구를 사용하여 필요한 아티팩트를 작성한 다음 인바운드

메시지 구문 분석 및 아웃바운드 메시지 구성을 위한 자체 코드를 제공할 수 있습니다. 제공된 공급업체 인터페이스 API를 사용할 수도 있습니다.

기존 서비스 설명을 사용하시겠습니까 아니면 새 서비스 설명을 작성하시겠습니까?

일부 상황에서 기존 서비스 설명을 시작점으로 사용해야 합니다. 예를 들어, 다음과 같습니다.

- 애플리케이션이 서비스 요청자이고 기존 웹 서비스를 호출하도록 디자인되어 있습니다.
- 애플리케이션이 서비스 제공자이고 이 애플리케이션은 기존 산업 표준 서비스 설명을 준수해야 합니다.

다른 상황에서 애플리케이션에 대해 새 서비스 설명을 작성해야 할 수도 있습니다.

다음에 수행할 작업

관련 정보:

CICS 카탈로그 관리자 예제 애플리케이션

CICS 카탈로그 관리자 예제 애플리케이션은 CICS 애플리케이션을 외부 클라이언트와 서버에 연결할 때 우수 사례를 표시하도록 설계되는 작업 COBOL 애플리케이션입니다.

SOAP 서비스 제공자 애플리케이션 계획

일반적으로 CICS 애플리케이션은 비즈니스 로직과 통신 로직이 분리될 수 있도록 구성되어야 합니다. 이 사례를 따를 경우 웹 서비스 제공자에 간단한 방식으로 새 애플리케이션 및 기존 애플리케이션을 배치하는 데 도움이 됩니다. 일부 상황에서는 애플리케이션 프로그램과 CICS 웹 서비스 지원 사이에 단순한 랩퍼 프로그램을 삽입해야 합니다.

그림 15는 통신 로직과 비즈니스 로직의 분리를 위해 파티션되는 일반적인 애플리케이션을 표시합니다.

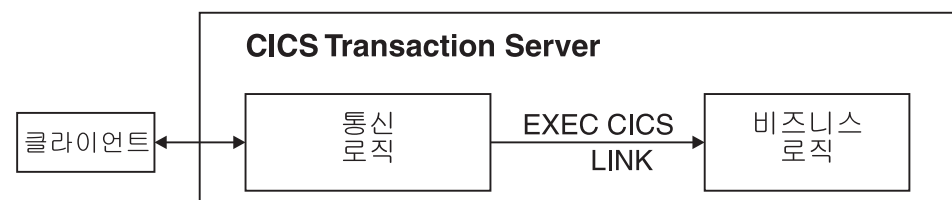


그림 15. 통신 및 비즈니스 로직으로 파티션된 애플리케이션

대부분의 경우 비즈니스 로직을 서비스 제공자 애플리케이션으로 직접 배치할 수 있습니다. 이는 56 페이지의 그림 16에 설명되어 있습니다.

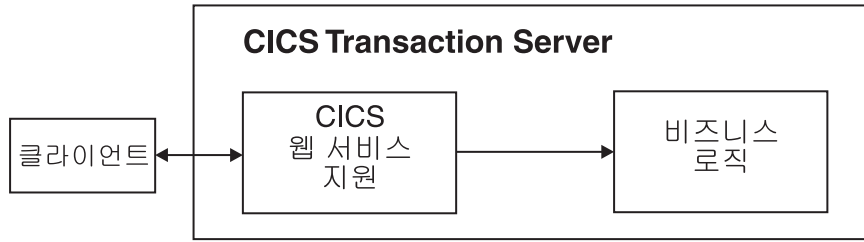


그림 16. CICS 애플리케이션을 웹 서비스 제공자로 단순 배치

이 단순 모델을 사용하려면 다음 조건을 적용하십시오.

CICS 웹 서비스 지원을 사용하여 SOAP 메시지와 애플리케이션 데이터 구조 간 매핑을 생성하는 경우:

비즈니스 로직에 대한 인터페이스에서 사용되는 데이터 유형은 CICS 웹 서비스 지원에 의해 지원되어야 합니다. 그렇지 않은 경우 CICS 웹 서비스 지원과 비즈니스 로직 사이에 래퍼 프로그램을 삽입해야 합니다.

기존 웹 서비스 설명에 부합하는 서비스를 제공하기 위해 기존 프로그램을 배치할 때 래퍼 프로그램도 필요하게 됩니다. 지원을 사용하여 웹 서비스 설명을 처리할 경우 결과적으로 생성되는 데이터 구조는 비즈니스 로직에 대한 인터페이스와 일치할 가능성이 매우 낮습니다.

CICS 웹 서비스 지원을 사용하지 않는 경우:

서비스 제공자 파이프라인의 메시지 핸들러가 비즈니스 로직과 직접 상호작용해야 합니다.

래퍼 프로그램 사용

CICS 웹 서비스 지원이 비즈니스 로직과 직접 상호작용하기 위한 코드를 생성할 수 없는 경우에 래퍼 프로그램을 사용하십시오. 예를 들어, CICS 웹 서비스 지원이 SOAP 메시지로 직접 매핑할 수 없는 데이터 구조를 비즈니스 로직에 대한 인터페이스가 사용할 수 있습니다. 이 상황에서 래퍼 프로그램을 사용하면 필요한 추가 데이터 조작을 제공할 수 있습니다.

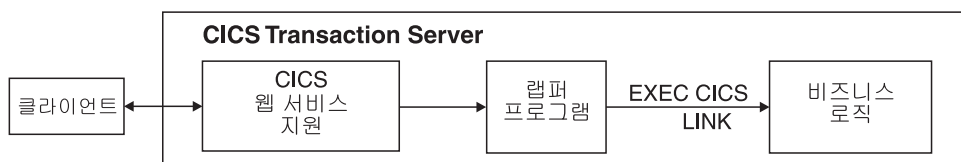


그림 17. 래퍼 프로그램을 사용하여 CICS 애플리케이션을 웹 서비스 제공자로 배치

지원이 지원하고 랩퍼 프로그램에 대한 인터페이스로 사용할 수 있는 두 번째 데이터 구조를 디자인해야 합니다. 그러면 랩퍼 프로그램에 두 가지 단순 기능이 생겨 다음을 수행합니다.

- 두 데이터 구조간 데이터 이동
- 기존 인터페이스를 사용하여 비즈니스 로직 호출

오류 핸들링

CICS 웹 서비스 지원을 사용할 계획인 경우 오류가 발생할 때 변경 롤백을 처리하는 방법에 대해서도 고려해야 합니다. 서비스 요청자에서 SOAP 요청 메시지를 수신하면 이 SOAP 메시지는 애플리케이션 프로그램에 전달되기 바로 전에 CICS에 의해 변환됩니다. 이 변환 중 오류가 발생하면 CICS는 메시지에 대해 수행한 모든 작업을 자동으로 롤백하지 않습니다. 예를 들어, 파이프라인에서 핸들러를 사용하여 SOAP 메시지에 대한 일부 추가 처리를 추가할 계획인 경우 이미 수행된 복구 가능한 변경사항을 롤백해야 하는지 여부를 결정해야 합니다.

예를 들어, 아웃바운드 SOAP 메시지에서 서비스 제공자 애플리케이션 프로그램이 서비스 요청자에게 응답 메시지를 보내는 중에 응답 SOAP 메시지 생성 중 CICS에 오류가 발생하면 애플리케이션 프로그램에 의해 작성된 모든 복구 가능 변경사항이 자동으로 취소됩니다. 동기화 지점을 추가하는 것이 애플리케이션 프로그램에 적절한지 여부를 고려해야 합니다.

제공자 애플리케이션에서 웹 서비스 원자적 트랜잭션을 사용할 계획이고 웹 서비스 요청자도 원자적 트랜잭션을 지원하는 경우 CICS가 트랜잭션을 롤백하도록 하는 모든 오류는 원격 요청자가 변경사항을 롤백하도록 하는 원인도 됩니다.

SOAP 서비스 요청자 애플리케이션 계획

일반적으로 CICS 애플리케이션은 비즈니스 로직과 통신 로직이 분리될 수 있도록 구성되어야 합니다. 이 사례를 따를 경우 웹 서비스 요청자에 간단한 방식으로 새 애플리케이션 및 기존 애플리케이션을 배치하는 데 도움이 됩니다. 거의 모든 상황에서 애플리케이션 프로그램과 CICS 웹 서비스 지원 사이에 단순한 랩퍼 프로그램을 삽입해야 합니다.

58 페이지의 그림 18는 통신 로직과 비즈니스 로직 사이의 분리를 위해 파티션되는 일반적인 애플리케이션을 표시합니다. 애플리케이션은 웹 서비스 요청자에서 비즈니스 로직의 재사용을 위해 적절하게 구성됩니다.

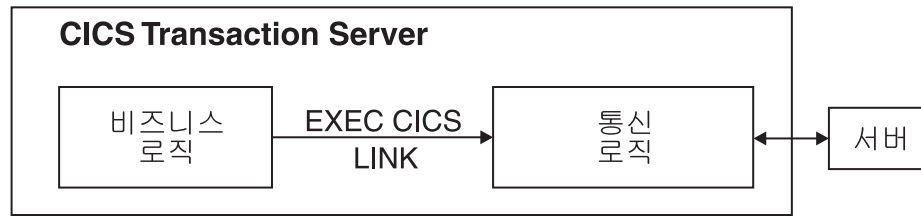


그림 18. 통신 및 비즈니스 로직으로 파티션된 애플리케이션

이 상황에서는 CICS 웹 서비스 지원을 호출하기 위해 기존 **EXEC CICS LINK** 명령을 사용할 수 없습니다.

- CICS 웹 서비스 지원을 사용하여 SOAP 메시지와 애플리케이션 데이터 구조 간 맵핑을 생성하는 경우 **EXEC CICS INVOKE SERVICE** 명령을 사용하고 애플리케이션의 데이터 구조를 CICS 웹 서비스 지원에 전달해야 합니다. 또한 비즈니스 로직에 대한 인터페이스에서 사용되는 데이터 유형은 CICS 웹 서비스 지원에 의해 지원되어야 합니다.

그러나 애플리케이션 프로그램이 호출하는 대상 **WEBSERVICE**가 제공자 모드인 경우 즉, 값이 **PROGRAM** 속성에 대해 정의된 경우 CICS는 **EXEC CICS LINK** 명령을 사용하여 요청을 자동으로 최적화합니다.

- CICS 웹 서비스 지원을 사용하지 않는 경우 자체 메시지를 구성하고 프로그램 **DFHPIRT**에 링크해야 합니다.

어느 방식이든 프로그램을 변경할 준비가 되지 않은 경우에는 비즈니스 로직이 웹 서비스를 직접 호출할 수 없음을 따릅니다. 웹 서비스 지원의 경우 이 옵션은 그림 19에 표시되지만 어느 경우든 이 옵션은 권장되지 않습니다.

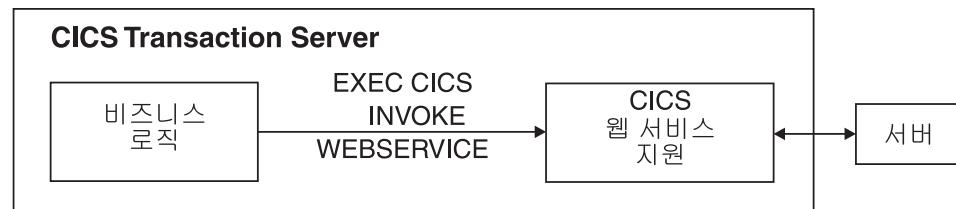


그림 19. CICS 애플리케이션을 웹 서비스 요청자로 단순 배치

래퍼 프로그램 사용

래퍼 프로그램 사용은 비즈니스 로직을 거의 변경하지 않은 상태로 유지하는 더 나은 솔루션입니다. 이 경우 래퍼는 다음과 같은 두 가지 목적을 가집니다.

- 비즈니스 로직을 대신하여 **EXEC CICS INVOKE SERVICE** 명령 또는 **EXEC CICS LINK PROGRAM(DFHPIRT)**을 발행합니다. 비즈니스 로직의 유일한 변경사항은 비즈니스 로직이 링크하는 프로그램의 이름입니다.
- 필요하다면 CICS 웹 서비스 지원이 SOAP 메시지로 직접 맵핑할 수 없는 데이터 구조를 애플리케이션이 사용하는 경우 필요한 데이터 조작을 제공할 수 있습니다.

웹 서비스 지원이 사용되는 경우의 이 구조는 그림 20에 설명되어 있습니다.

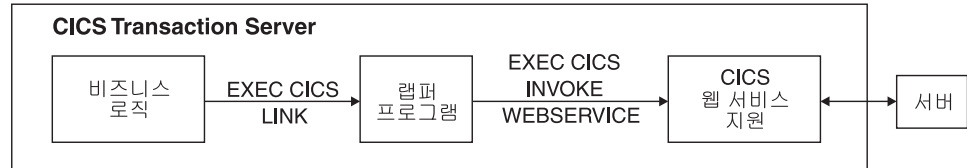


그림 20. 래퍼 프로그램을 사용하여 CICS 애플리케이션을 웹 서비스 요청자로 배치

오류 핸들링

CICS 웹 서비스 지원을 사용할 계획인 경우 오류가 발생할 때 변경 롤백을 처리하는 방법에 대해서도 고려해야 합니다. 서비스 요청자 애플리케이션이 서비스 제공자로부터 SOAP 결함 메시지를 수신하면 애플리케이션 프로그램이 결함 메시지를 처리하는 방식을 결정해야 합니다. SOAP 결함 메시지를 수신하면 CICS는 변경사항을 자동으로 롤백하지 않습니다.

요청자 애플리케이션 프로그램에서 웹 서비스 원자적 트랜잭션을 구현할 계획인 경우 오류 핸들링은 달라집니다. 원격 서비스 제공자에게 오류가 발생하고 변경사항을 롤백하는 경우 SOAP 결함 메시지가 리턴되고 CICS의 로컬 트랜잭션도 롤백됩니다. 로컬 최적화가 적용되면 서비스 요청자 또는 제공자는 같은 트랜잭션을 사용합니다. 제공자에게 오류가 발생하면 요청자의 트랜잭션에서 작성된 모든 변경사항도 롤백됩니다.

제 6 장 웹 서비스 인프라 작성

웹 서비스를 CICS에 배치하려면 필요한 전송 인프라를 작성하고 웹 서비스 요청을 처리할 한 개 이상의 파이프라인을 정의해야 합니다. 일반적으로 한 파이프라인은 다양한 웹 서비스에 대한 요청을 처리할 수 있고 새 웹 서비스를 CICS 시스템에 배치할 때 기존 파이프라인 사용을 선택할 수 있습니다.

웹 서비스를 위한 CICS 시스템 구성

웹 서비스를 사용하려면 먼저 CICS 시스템이 올바르게 구성되어야 합니다.

프로시저

1. PL/I용 Language Environment® 지원을 설치했는지 확인하십시오. 자세한 정보는 Language Environment 지원 설치를 참조하십시오.
2. 유니코드에 대한 z/OS 지원을 활성화하십시오. z/OS 변환 서비스를 사용으로 설정하고 CICS가 SOAP 메시지와 애플리케이션 프로그램 간에 수행할 데이터 변환을 지정하는 변환 이미지를 설치해야 합니다. 자세한 정보는 z/OS Unicode Services User's Guide and Reference의 내용을 참조하십시오.

SOAP 웹 서비스 작성

기존 CICS 애플리케이션을 SOAP 웹 서비스로 노출하여 SOAP 웹 서비스 제공자 또는 요청자 역할을 할 새 CICS 애플리케이션을 작성할 수 있습니다.

시작하기 전에

SOAP 웹 서비스 작성을 시작하기 전에 다음 태스크를 수행하십시오.

1. 웹 서비스를 지원하도록 CICS 시스템을 구성합니다. 『웹 서비스를 위한 CICS 시스템 구성』의 내용을 참조하십시오.
2. 웹 서비스의 배치를 지원하기 위해 필요한 인프라를 작성합니다. 제 6 장 『웹 서비스 인프라 작성』의 내용을 참조하십시오.
3. 웹 서비스 지원을 사용할지 여부를 결정합니다. 53 페이지의 『SOAP 웹 서비스 사용 계획』의 내용을 참조하십시오.

이 태스크 정보

CICS 웹 서비스 지원은 사용자가 새 SOAP 웹 서비스 제공자 또는 서비스 요청자 애플리케이션에 필요한 아티팩트를 작성하고 기존 애플리케이션을 웹 서비스 제공자로 사용할 수 있도록 돕는 제공 유틸리티입니다.

CICS 웹 서비스 지원은 단순 언어 구조에서 WSDL 문서를 작성하거나 기존 WSDL 문서에서 언어 구조를 작성할 수 있으며 COBOL, C/C++ 및 PL/I를 지원합니다. 또한 SOAP 메시지를 컨테이너 및 COMMAREA로 또는 그 반대로 자동 런타임 변환을 가능하게 하기 위해 사용되는 정보를 생성합니다. 이 정보는 파이프라인 처리 중 CICS 웹 서비스 지원에서 사용됩니다.

다음 프로시저에서 설명하는 대로 웹 서비스를 작성하고 작성한 웹 서비스가 올바르게 작동하는지 유효성 검증하십시오.

프로시저

1. 다음 네 가지 방식 중 하나로 SOAP 웹 서비스를 작성하십시오.
 - 웹 서비스 지원을 사용하여 웹 서비스 설명 또는 언어 구조를 작성하고 작성한 항목을 CICS에 배치합니다. **PIPELINE SCAN** 명령을 사용하여 필수 CICS 자원을 자동 작성합니다.
 - z Systems용 Rational Developer 또는 Java API를 사용하여 웹 서비스 설명 또는 언어 구조를 작성하고 작성한 항목을 CICS에 배치합니다. **PIPELINE SCAN** 명령을 사용하여 필수 CICS 자원을 자동 작성합니다.
 - 데이터 변환을 포함하여 인바운드 및 아웃바운드 메시지에서 XML을 처리하기 위해 애플리케이션 프로그램을 작성하거나 변경하고 파이프라인에 올바른 컨테이너를 채웁니다. 필수 CICS 자원을 수동으로 작성해야 합니다.
 - Axis2 애플리케이션을 웹 서비스로 배치합니다.
2. 웹 서비스를 시작하여 웹 서비스가 의도한 대로 작동하는지 테스트하십시오. 웹 서비스 지원을 사용하여 웹 서비스를 배치하는 경우 **SET WEBSERVICE** 명령을 사용하여 유효성 검증을 켤 수 있습니다. 이 유효성 검증에서는 데이터가 올바르게 변환되는지 확인합니다.

다음에 수행할 작업

이러한 단계는 다음 주제에서 보다 자세히 설명됩니다.

CICS 웹 서비스 지원자

CICS 웹 서비스 지원은 사용자가 기존 CICS 애플리케이션을 웹 서비스로 변환시키도록 도움을 줄 수 있고, CICS 애플리케이션이 외부 제공자가 제공하는 웹 서비스를 사용할 수 있도록 도움을 줄 수 있는 일련의 일괄처리 유틸리티입니다. 지원은 최소 프로그래밍 노력으로 서비스 제공자 및 서비스 요청자에서 사용하기 위한 CICS 애플리케이션의 빠른 배치를 지원합니다.

CICS에 웹 서비스 지원을 사용할 때, 인바운드 메시지를 구문 분석하고 아웃바운드 메시지를 구성하기 위한 사용자 자체 코드를 작성할 필요가 없습니다. CICS는 SOAP 메시지와 애플리케이션 프로그램의 데이터 구조 사이의 데이터를 맵핑합니다.

지원은 단순 언어 구조에서 WSDL 문서를 작성하거나 기존 WSDL 문서에서 언어 구조를 작성할 수 있고, COBOL, C/C++ 및 PL/I를 지원합니다. 이는 컨테이너 및 COMMAREA에 대한 SOAP 메시지의 자동화 런타임 변환을 가능하게 하거나 또는 그 반대에 사용되는 정보를 생성합니다.

CICS 웹 서비스 지원은 두 개의 유틸리티 프로그램으로 구성됩니다.

DFHLS2WS

언어 구조에서 웹 서비스 바인딩 파일을 생성합니다. 이 유틸리티는 웹 서비스 설명도 생성합니다.

DFHWS2LS

웹 서비스 설명에서 웹 서비스 바인딩 파일을 생성합니다. 이 유틸리티는 애플리케이션 프로그램에서 사용할 수 있는 언어 구조도 생성합니다.

양쪽 모두의 프로그램을 실행하기 위한 JCL 프로시저는 *hlq.XDFHINST* 라이브러리에 있습니다.

웹 서비스 지원의 유틸리티 프로그램 및 데이터 맵핑에 대한 자세한 정보는 다음 주제를 참조하십시오.

DFHLS2WS: 고급 언어 대 WSDL 변환:

DFHLS2WS 프로시저는 고급 언어 데이터 구조에서 웹 서비스 바인딩 파일과 웹 서비스 설명을 생성합니다. DFHLS2WS는 CICS 애플리케이션 프로그램을 서비스 제공자로 표시할 때 사용할 수 있습니다.

DFHLS2WS와 기호 매개변수, 입력 매개변수와 설명, 작업 예제에 대한 작업 제어 명령문은 이 프로시저를 사용하는 데 도움을 줍니다.

DFHLS2WS의 작업 제어 명령문

JOB 작업을 시작합니다.

EXEC

프로시저 이름(DFHLS2WS)을 지정합니다.

INPUT.SYSUT1 DD

입력을 지정합니다. 입력 매개변수는 일반적으로 입력 스트림에 지정됩니다. 그러나 데이터 세트 또는 파티션된 데이터 세트의 멤버에 정의될 수 있습니다.

기호 매개변수

DFHLS2WS에서는 다음 기호 매개변수가 정의됩니다.

JAVADIR=path

DFHLS2WS에서 사용되는 Java 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/에 추가되어 /usr/lpp/path의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

PATHPREF=prefix

다른 매개변수에서 사용되는 z/OS UNIX 디렉토리 경로를 확장하는 선택적 접두부를 지정합니다. 기본값은 비어 있는 문자열입니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

SERVICE=value

이 매개변수는 IBM 지원 센터에서 지시하는 경우에만 사용합니다.

TMPDIR=tmpdir

DFHLS2WS가 z/OS UNIX에서 임시 작업공간으로 사용하는 디렉토리의 위치를 지정합니다. 작업이 실행되는 사용자 ID는 이 디렉토리에 대한 읽기 및 쓰기 권한이 있어야 합니다.

기본값은 /tmp입니다.

TMPFILE=tmpprefix

DFHLS2WS가 임시 작업공간 파일의 이름을 구성하기 위해 사용하는 접두부를 지정합니다.

기본값은 LS2WS입니다.

USSDIR=path

UNIX 시스템 서비스 파일 시스템에서 CICS TS 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/cicsts/에 추가되어 /usr/lpp/cicsts/path의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **USSDIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

임시 작업공간

DFHLS2WS는 런타임에 다음 세 개 임시 파일을 작성합니다.

`tmpdir/tmpprefix.in`

`tmpdir/tmpprefix.out`

`tmpdir/tmpprefix.err`

여기서,

`tmpdir`은 **TMPDIR** 매개변수에 지정된 값입니다.

`tmpprefix`는 **TMPFILE** 매개변수에 지정된 값입니다.

TMPDIR 및 **TMPFILE**이 지정되지 않은 경우 파일의 기본 이름은 다음과 같습니다.

`/tmp/LS2WS.in`

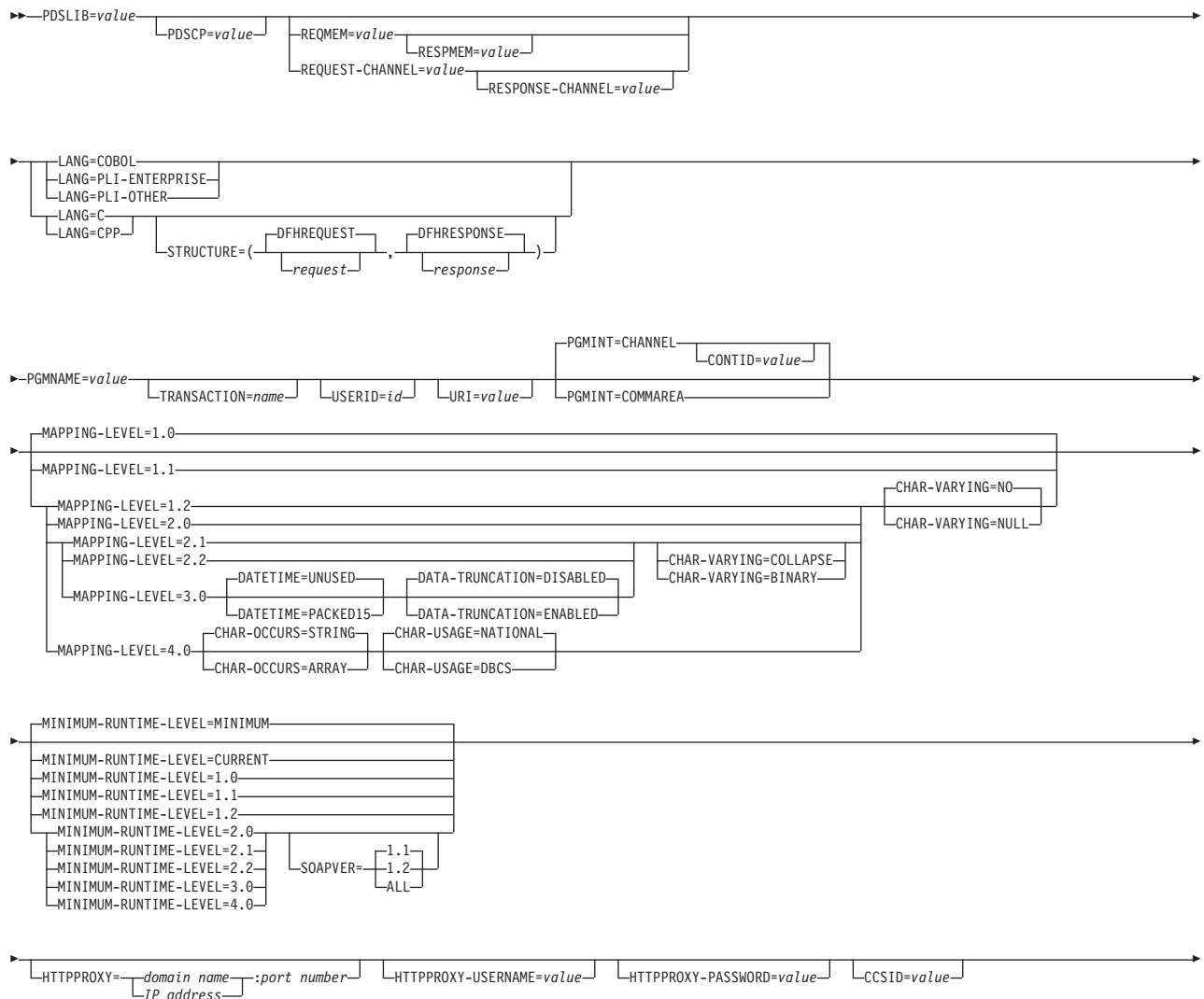
/tmp/LS2WS.out

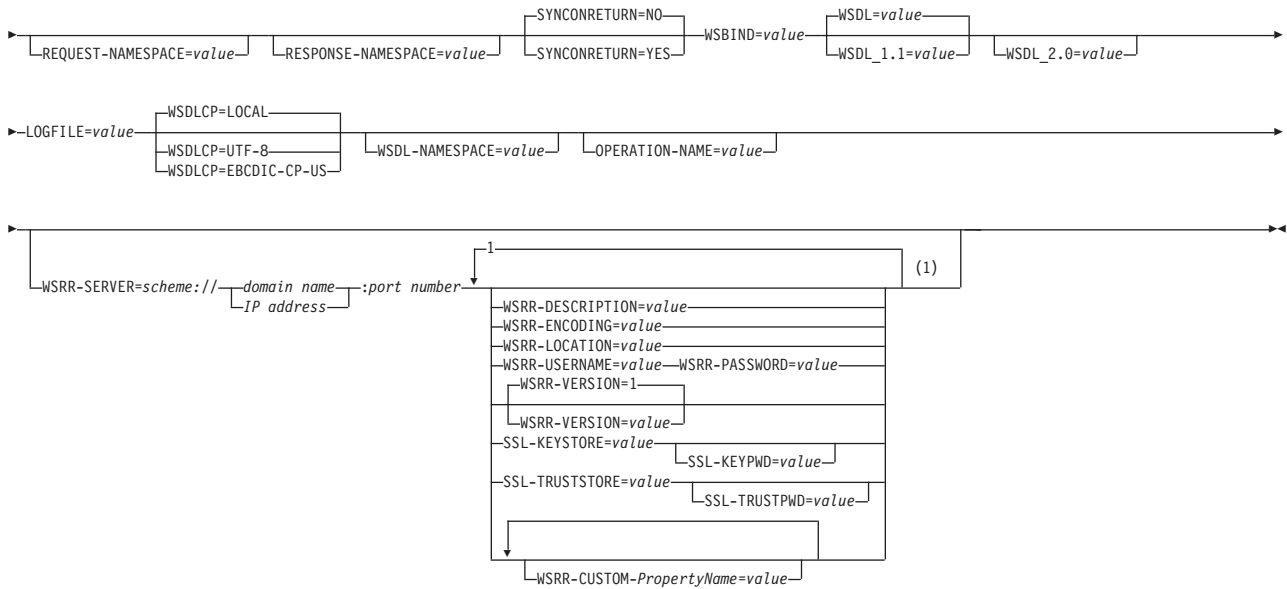
/tmp/LS2WS.err

중요사항: DFHLS2WS는 z/OS UNIX 파일 또는 데이터 세트 멤버에 대한 액세스를 차단하지 않습니다. 따라서 두 개 이상의 DFHLS2WS 인스턴스가 동시에 실행되고 동일한 임시 작업공간 파일을 사용하는 경우, 다른 작업이 작업공간 파일을 사용하는 동안 특정 작업이 해당 파일을 겹쳐쓸 수 있어 예상치 못한 실패를 경험할 수 있습니다.

따라서 이러한 상황을 방지하는 이름 지정 규칙과 작동 프로시저를 고안해야 합니다. 예를 들어, 시스템 기호 매개변수 **SYSUID**를 사용하여 개별 사용자에게 고유한 작업공간 파일 이름을 생성할 수 있습니다. 이러한 임시 파일은 작업이 종료되기 전에 삭제됩니다.

DFHLS2WS의 입력 매개변수





참고:

- 1 **WSRR-SERVER** 매개변수가 설정될 때 지정할 수 있는 각 **WSRR** 매개변수는 한 번만 지정할 수 있습니다. 이 규칙의 예외는 **WSRR-CUSTOM** 매개변수로 최대 255번까지 지정할 수 있습니다.

매개변수 사용

- 입력 매개변수는 순서에 관계 없이 지정할 수 있습니다.
- 각 매개변수는 행을 바꿔 시작해야 합니다.
- 매개변수는 사용 시 해당 연속 문자와 함께 열 72를 넘어가서는 안됩니다. 열 73 - 80은 공백이어야 합니다.
- 매개변수가 너무 길어 한 행에 맞지 않는 경우에는 행 끝에서 별표(*) 문자를 사용하여 매개변수가 다음 행에서 연속됨을 표시해야 합니다. 별표 앞의 모든 내용은 공백을 포함하여 매개변수의 일부로 간주됩니다. 예를 들어, 다음과 같습니다.

```
WSBIND=wsbinddir*
/app1
```

구문은 다음과 같습니다.

```
WSBIND=wsbinddir/app1
```

- 행에서 첫 번째 문자 위치의 # 문자는 주석 문자입니다. 이 행은 무시됩니다.
- 행에서 마지막 문자 위치의 쉼표는 선택적 선 구분 기호이며 무시됩니다.

매개변수 설명

CCSID=value

런타임에 애플리케이션 데이터 구조에서 문자 데이터를 인코딩하는 데 사용되는 CCSID를 지정합니다. 이 매개변수의 값은 **LOCALCCSID** 시스템 초기화 매개변수의 값을 대체합니다. value는 Java 및 z/OS 변환 서비스가 지원하는 EBCDIC CCSID

여야 합니다(z/OS Unicode Services User's Guide and Reference 참조). 이 매개변수를 지정하지 않으면 시스템 초기화 매개변수에 지정된 CCSID를 사용하여 애플리케이션 데이터 구조가 인코딩됩니다.

이 매개변수는 임의 매핑 레벨에 사용할 수 있습니다. 그러나 생성된 파일을 CICS TS 3.1 리전에 배치하려면 APAR PK23547을 적용하여 웹 서비스 바인딩 파일을 설치할 수 있는 코드의 최소 런타임 레벨을 달성해야 합니다.

CHAR-VARYING={NO|NULL|COLLAPSE|BINARY}

매핑 레벨이 1.2 이상일 때 언어 구조의 문자 필드가 매핑되는 방식을 지정합니다. COBOL의 문자 필드가 X 유형의 Picture 절인 경우(예: PIC(X) 10) C/C++의 문자 필드는 문자 배열입니다. 다음 옵션을 선택할 수 있습니다.

NO 문자 필드가 <xsd:string>에 매핑되고 고정 길이 필드로 처리됩니다. 데이터의 최대 길이는 필드 길이와 같습니다. 매핑 레벨이 2.0 이하인 COBOL 및 PL/I의 경우 **CHAR-VARYING** 매개변수의 기본값은 NO입니다.

이 값은 엔터프라이즈 및 기타 PL/I 언어 구조에는 적용되지 않습니다.

NULL

문자 필드가 <xsd:string>에 매핑되고 널 종료 문자열로 처리됩니다. CICS는 SOAP 메시지에서 변환될 때 종료 널 문자를 추가합니다. 문자열의 최대 길이는 언어 구조에 표시된 길이보다 한 문자 적은 길이로 계산됩니다. C/C++의 경우 **CHAR-VARYING** 매개변수의 기본값은 NULL입니다.

이 값은 엔터프라이즈 및 기타 PL/I 언어 구조에는 적용되지 않습니다.

COLLAPSE

문자 필드가 <xsd:string>에 매핑됩니다. 필드에서 후미 공백은 SOAP 메시지에 포함되지 않습니다. 인바운드 SOAP 메시지는 선두, 후미, 임베디드 공백을 모두 제거하기 위해 구문 분석됩니다. 매핑 레벨이 2.1 이상인 COBOL 및 PL/I의 경우 **CHAR-VARYING** 매개변수의 기본값은 COLLAPSE입니다.

가변 길이 값 및 공백에 대한 자세한 정보는 가변 길이 값 및 공백에 대한 지원을 참조하십시오.

BINARY

문자 필드가 <xsd:base64binary>에 매핑되고 고정 길이 필드로 처리됩니다. **CHAR-VARYING** 매개변수의 BINARY 값은 매핑 레벨이 2.1 이상인 경우에만 사용할 수 있습니다.

CHAR-OCCURS={STRING|ARRAY}

매핑 레벨이 4.0 이상일 때 언어 구조의 문자 배열이 매핑되는 방식을 지정합니다 (예: PIC X OCCURS 20). 이 매개변수는 COBOL 언어만 사용할 수 있습니다.

ARRAY

문자 배열이 XML 배열에 맵핑됩니다. 이는 모든 문자가 개별 XML 요소에 맵핑됨을 의미합니다. 맵핑 레벨이 3.0 이하인 경우의 동작이기도 합니다.

STRING

문자 배열이 XML 문자열에 맵핑됩니다. 이는 전체 COBOL 배열이 단일 XML 요소에 맵핑됨을 의미합니다.

CHAR-USAGE={NATIONAL|DBCS}

COBOL에서 국가적 데이터 유형인 PIC N은 UTF-16 또는 DBCS 데이터에 사용될 수 있습니다. 이 설정은 NSYMBOL 컴파일러 옵션에 의해 제어됩니다. 데이터가 적절하게 처리되는지 확인하려면 보조의 **CHAR-USAGE** 매개변수를 NSYMBOL 컴파일러 옵션과 동일한 값으로 설정해야 합니다. 이는 주로 UTF-16을 사용할 때 CHAR-USAGE=NATIONAL로 설정됩니다.

DBCS PIC (*n*) 필드의 데이터는 UTF-16 인코딩 데이터로 처리됩니다.

NATIONAL

PIC (*n*) 필드의 데이터는 DBCS 인코딩 데이터로 처리됩니다.

CONTID=*value*

서비스 제공자에서, SOAP 메시지를 나타내는 데 사용되는 상위 레벨 데이터 구조를 보유하는 컨테이너의 이름을 지정합니다.

CICS가 대상 애플리케이션 프로그램으로 전달하는 컨테이너의 길이는 요청 컨테이너의 길이와 응답 컨테이너의 길이 중 긴 값입니다.

DATA-TRUNCATION={DISABLED|ENABLED}

가변 길이 데이터가 고정 길이 필드 구조에서 용인되는지 여부를 지정합니다.

DISABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 거부하고 오류 메시지를 발행합니다.

ENABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 용인하고 누락된 데이터를 널값으로 처리합니다.

DATETIME={UNUSED|PACKED15}

고급 언어 구조의 잠재적 ABSTIME 필드가 시간소인으로 맵핑되는지 여부를 지정합니다.

PACKED15

길이가 15(8바이트)인 팩형 10진수 필드가 CICS ABSTIME 필드로 처리되고 시간소인으로 맵핑됩니다.

UNUSED

길이가 15(8바이트)인 팩형 10진수 필드가 시간소인으로 처리되지 않습니다.

이 매개변수는 맵핑 레벨 3.0에서 설정할 수 있습니다.

HTTPPROXY={domain name:port number|IP address:port number}

WSDL에 인터넷에 있는 다른 WSDL 파일에 대한 참조를 포함하고 DFHLS2WS를 실행하는 시스템이 인터넷에 액세스하기 위해 프록시 서버를 사용하는 경우, 프록시 서버의 포트 번호와 도메인 이름 또는 IP 주소를 지정하십시오. 예를 들어, 다음과 같습니다.

HTTPPROXY=proxy.example.com:8080

다른 경우에는 이 매개변수가 필요하지 않습니다.

HTTPPROXY-PASSWORD=value

DFHLS2WS를 실행하는 시스템이 인터넷에 액세스하기 위해 HTTP 프록시 서버를 사용하고 HTTP 프록시 서버가 기본 인증을 사용하는 경우 **HTTPPROXY-USERNAME**과 함께 사용해야 하는 HTTP 프록시 비밀번호를 지정합니다. **HTTPPROXY**도 지정하는 경우에만 이 매개변수를 사용할 수 있습니다.

HTTPPROXY-USERNAME=value

DFHLS2WS를 실행하는 시스템이 인터넷에 액세스하기 위해 HTTP 프록시 서버를 사용하고 HTTP 프록시 서버가 기본 인증을 사용하는 경우 **HTTPPROXY-PASSWORD**와 함께 사용해야 하는 HTTP 프록시 사용자 이름을 지정합니다. **HTTPPROXY**도 지정하는 경우에만 이 매개변수를 사용할 수 있습니다.

LANG=COBOL

고급 언어 구조의 프로그래밍 언어가 COBOL인 것으로 지정합니다.

LANG=PLI-ENTERPRISE

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I인 것으로 지정합니다.

LANG=PLI-OTHER

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I 이외의 PL/I 레벨인 것으로 지정합니다.

LANG=C

고급 언어 구조의 프로그래밍 언어가 C인 것으로 지정합니다.

LANG=CPP

고급 언어 구조의 프로그래밍 언어가 C++인 것으로 지정합니다.

LOGFILE=value

DFHLS2WS가 해당 활동 로그 및 추적 정보를 기록하는 파일의 완전한 z/OS UNIX 이름입니다. DFHLS2WS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다.

이 파일은 일반적으로 사용하지 않지만 DFHLS2WS 관련 문제점이 발생하는 경우 IBM 서비스 조직이 요청할 수 있습니다.

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0}

웹 서비스 바인딩 파일과 웹 서비스 설명을 생성할 때 DFHLS2WS가 사용하는 맵핑 레벨을 지정합니다. 다음 옵션을 선택할 수 있습니다.

- 1.0** 이 맵핑 레벨은 기본값입니다. 웹 서비스 바인딩 파일이 CICS TS 3.1 맵핑 레벨을 사용하여 생성됨을 나타냅니다.
- 1.1** 이 특정 레벨에서 바인딩 파일을 다시 생성하려면 이 맵핑을 사용합니다.
- 1.2** 이 맵핑 레벨에서는 **CHAR-VARYING** 매개변수를 사용하여 런타임에 문자 배열이 처리되는 방식을 제어할 수 있습니다. **VARYING** 및 **VARYINGZ** 배열은 PL/I에서도 지원됩니다.
- 2.0** 이 맵핑 레벨은 CICS TS 3.2 리전 이상에서 언어 구조와 웹 서비스 바인딩 파일 간 맵핑에 대한 개선사항을 이용하기 위해 사용합니다.
- 2.1** APAR PK59794가 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2보다 상위 리전에는 이 맵핑 레벨을 사용합니다. 이 맵핑 레벨에서는 **CHAR-VARYING** 매개변수에 새 값, **COLLAPSE** 및 **BINARY**를 사용할 수 있습니다. COBOL의 **FILLER** 필드와 PL/I의 * 필드는 이 맵핑 레벨에서 시스템적으로 무시되며 생성된 WSDL 문서에 이 필드가 나타나지 않고 런타임에 데이터 구조에서 적절한 값이 유지됩니다.
- 2.2** APAR PK69738이 적용된 CICS TS 3.2 리전이나 CICS TS 3.2보다 상위 리전에 이 맵핑 레벨을 사용하면 DFHWS2LS 사용 시 맵핑 개선사항을 이용할 수 있습니다.
- 3.0** 이 맵핑 레벨은 CICS TS 4.1 리전에 사용합니다. 이 맵핑 레벨에서는 **REQUEST-CHANNEL** 및 **RESPONSE-CHANNEL** 매개변수를 설정하여 해당 인터페이스에서 많은 컨테이너를 사용하는 애플리케이션에서 웹 서비스를 작성할 수 있습니다. **DATETIME** 매개변수를 설정하여 **dateTime** 필드를 XML 시간소인에 맵핑할 수도 있습니다.
- 4.0** 이 맵핑 레벨은 CICS TS 5.2 리전에 사용합니다. 이 맵핑 레벨에서는 COBOL **OCCURS DEPENDING ON** 필드와 **CHAR-OCCURS** 매개변수를 사용할 수 있습니다.

맵핑 레벨에 대한 자세한 정보는 CICS 지원에 대한 맵핑 레벨을 참조하십시오.

MINIMUM - RUNTIME -

LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0|CURRENT}

웹 서비스 바인딩 파일이 배치될 수 있는 최소 CICS 런타임 환경을 지정합니다. 지정한 다른 매개변수와 일치하지 않는 레벨을 선택하면 오류 메시지를 받게 됩니다. 다음 옵션을 선택할 수 있습니다.

MINIMUM

지정된 매개변수의 경우 CICS의 런타임 레벨 중 가장 낮은 레벨이 자동으로 할당됩니다.

- 1.0** 생성된 웹 서비스 바인딩 파일이 APAR PK15904 및 PK23547이 적용되지 않은 CICS TS 3.1 리전에 배치됩니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 1.1** 생성된 웹 서비스 바인딩 파일이 최소 APAR PK15904가 적용된 CICS TS 3.1 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 1.1 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 1.2** 생성된 웹 서비스 바인딩 파일이 APAR PK15904 및 PK23547이 모두 적용된 CICS TS 3.1 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 1.2 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 3.2 이상 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 2.0 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.1** 생성된 웹 서비스 바인딩 파일이 APAR PK59794가 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2보다 상위 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 2.1 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.2** 생성된 웹 서비스 바인딩 파일이 APAR PK69738이 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2보다 상위 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 2.2 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 3.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 4.1 이상 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 3.0 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 4.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 5.2 이상 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 4.0 이하 맵핑 레벨을 사용할 수 있습니다. 이 레벨에서는 선택적 매개변수를 사용할 수 있습니다.

CURRENT

생성된 웹 서비스 바인딩 파일이 웹 서비스 바인딩 파일을 생성하기 위해 사용하는 런타임 레벨과 동일한 런타임 레벨에서 CICS 리전에 배치됩니다.

OPERATION-NAME=value

생성된 WSDL 문서에서 사용되는 조작 이름을 지정합니다. 값을 제공하지 않으면 **PGMNAME** 매개변수의 값과 그 뒤의 **operation** 값을 사용하여 기본 이름이 생성됩니다.

PDSLIB=value

처리할 고급 언어 데이터 구조를 포함하는 파티션된 데이터 세트의 이름을 지정합니다. 요청 및 응답에 사용되는 데이터 세트 멤버는 각각 **REQMEM** 및 **RESPMEM** 매개변수에 지정됩니다.

제한사항: 파티션된 데이터 세트 내 레코드는 80바이트의 고정 길이를 갖습니다.

PDSCP=value

REQMEM 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에서 사용되는 코드 페이지를 지정합니다. 여기서 *value*는 CCSID 번호 또는 Java 코드 페이지 번호입니다. 이 매개변수가 지정되지 않으면 z/OS UNIX System Services 코드 페이지를 사용합니다. 예를 들어, **PDSCP=037**을 지정할 수 있습니다.

PGMINT={CHANNEL|COMMAREA}

서비스 제공자의 경우, CICS가 대상 애플리케이션 프로그램으로 데이터를 전달하는 방식을 지정합니다.

CHANNEL

CICS가 채널 인터페이스를 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

- 맵핑 레벨이 3.0 이하인 경우에는 입출력에 모두 사용되는 하나의 컨테이너만 채널에 포함될 수 있습니다. 컨테이너의 이름을 지정하려면 **CONTID** 매개변수를 사용하십시오. 기본 이름은 DFHWS-DATA입니다.
- 맵핑 레벨 3.0에서는 채널에 여러 컨테이너가 포함될 수 있습니다. **REQUEST-CHANNEL** 및 **RESPONSE-CHANNEL** 매개변수를 사용하십시오. **PDSLIB**, **REQMEM** 또는 **RESPMEM**은 지정하지 마십시오.

COMMAREA

CICS가 통신 영역(COMMAREA)을 사용하여 대상 애플리케이션 프로그램으로 데이터를 전달합니다.

대상 애플리케이션 프로그램이 요청을 처리한 경우에는 동일한 메커니즘으로 응답을 리턴해야 합니다. 즉, 요청이 통신 영역에서 수신된 경우에는 응답도 통신 영역에서 리턴되어야 하고 요청이 컨테이너에서 수신된 경우에는 응답도 컨테이너에서 리턴되어야 합니다. CICS가 대상 애플리케이션 프로그램으로 전달하는 통신 영역 또는 컨테이너의 길이는 요청 통신 영역 또는 컨테이너의 길이와 응답 통신 영역 또는 컨테이너의 길이 중 긴 값입니다.

PGMNAME=value

웹 서비스로 노출될 대상 애플리케이션 프로그램에 대한 CICS PROGRAM 자원의 이름을 지정합니다. CICS 웹 서비스 지원은 이 프로그램에 링크됩니다.

REQMEM=value

웹 서비스 요청에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 지정합니다. 서비스 제공자의 경우 웹 서비스 요청은 애플리케이션 프로그램에 입력됩니다.

REQUEST-CHANNEL=value

채널 설명 문서의 이름과 위치를 지정합니다. 채널 설명은 웹 서비스 요청자로부터 SOAP 메시지를 수신할 때 웹 서비스 제공자 애플리케이션이 해당 인터페이스에서 사용할 수 있는 컨테이너를 설명합니다. 채널 설명은 CICS 제공 채널 스키마를 준수해야 하는 XML 문서입니다.

이 매개변수는 맵핑 레벨 3.0에서만 사용할 수 있습니다.

REQUEST-NAMESPACE=value

생성된 웹 서비스 설명에서 요청 메시지에 대한 XML 스키마의 네임스페이스를 지정합니다. 이 매개변수를 지정하지 않으면 CICS가 네임스페이스를 자동으로 생성합니다.

RESPMEM=value

웹 서비스 응답에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 지정합니다. 서비스 제공자의 경우 웹 서비스 응답은 애플리케이션 프로그램에서 출력됩니다.

응답과 관련이 없으면 즉, 단방향 메시지인 경우에는 이 매개변수를 생략하십시오.

RESPONSE-CHANNEL=value

채널 설명 문서의 이름과 위치를 지정합니다. 채널 설명은 웹 서비스 요청자로 SOAP 응답 메시지를 보낼 때 웹 서비스 제공자 애플리케이션이 해당 인터페이스에서 사용할 수 있는 컨테이너를 설명합니다. 채널 설명은 CICS 제공 채널 스키마를 준수해야 하는 XML 문서입니다.

이 매개변수는 맵핑 레벨 3.0에서만 사용할 수 있습니다.

RESPONSE-NAMESPACE=value

생성된 웹 서비스 설명에서 응답 메시지에 대한 XML 스키마의 네임스페이스를 지정합니다. 이 매개변수를 지정하지 않으면 CICS가 네임스페이스를 자동으로 생성합니다.

SOAPVER={1.1|1.2|ALL}

생성된 웹 서비스 설명에서 사용할 SOAP 레벨을 지정합니다. 이 매개변수는 **MINIMUM-RUNTIME-LEVEL**이 2.0 이상으로 설정된 경우에만 사용할 수 있습니다.

1.1 SOAP 1.1 프로토콜이 웹 서비스 설명에 대한 바인딩으로 사용됩니다.

1.2 SOAP 1.2 프로토콜이 웹 서비스 설명에 대한 바인딩으로 사용됩니다.

ALL SOAP 1.1 또는 1.2 프로토콜을 웹 서비스 설명에 대한 바인딩으로 사용할 수 있습니다.

이 매개변수의 값을 지정하지 않는 경우 기본값은 다음과 같이 작성하려는 WSDL 버전에 따라 다릅니다.

- WSDL 1.1만 필요한 경우에는 SOAP 1.1 바인딩이 사용됩니다.
- WSDL 2.0만 필요한 경우에는 SOAP 1.2 바인딩이 사용됩니다.
- WSDL 1.1과 WSDL 2.0이 모두 필요한 경우에는 각 웹 서비스 설명에 SOAP 1.1 및 1.2 바인딩이 모두 사용됩니다.

SSL-KEYSTORE=*value*

이 선택적 매개변수는 키 저장소 파일의 완전한 위치를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-KEYPWD=*value*

이 선택적 매개변수는 키 저장소의 비밀번호를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-TRUSTSTORE=*value*

이 선택적 매개변수는 신뢰 저장소 파일의 완전한 위치를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-TRUSTPWD=*value*

이 선택적 매개변수는 신뢰 저장소의 비밀번호를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

STRUCTURE=(*request,response*)

C 및 C++의 경우에만, **REQMEM** 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에 포함된 상위 레벨 구조의 이름을 지정합니다.

request

REQMEM 매개변수가 지정되는 경우 요청을 포함하는 상위 레벨 구조의 이름을 지정합니다. 기본값은 DFHREQUEST입니다.

파티션된 데이터 세트 멤버에는 사용자가 지정하는 이름을 갖는 상위 레벨 구조 또는 이름을 지정하지 않는 경우 DFHREQUEST 구조가 포함되어야 합니다.

response

RESPMEM 매개변수가 지정되는 경우 응답을 포함하는 상위 레벨 구조의 이름을 지정합니다. 기본값은 DFHRESPONSE입니다.

값을 지정하는 경우, 파티션된 데이터 세트 멤버에는 사용자가 지정하는 이름을 갖는 상위 레벨 구조 또는 DFHRESPONSE 구조(이름을 지정하지 않는 경우)가 포함되어야 합니다.

SYNCONRETURN={NO|YES}

원격 웹 서비스가 동기점을 발행할 수 있는지 여부를 지정합니다.

NO 원격 웹 서비스가 동기점을 발행할 수 없습니다. 이 값은 기본값입니다. 원격 웹 서비스가 동기점을 발행하면 ADPL 이상 종료와 함께 실패합니다.

YES 원격 웹 서비스가 동기점을 발행할 수 있습니다. YES를 선택하면 원격 웹 서비스에서 제어가 리턴되는 경우 원격 태스크가 개별 작업 단위로 커미트 됩니다. 원격 웹 서비스가 복구 가능 자원을 갱신하고 리턴 후 실패하면 해당 자원에 대한 갱신을 백아웃할 수 없습니다.

TRANSACTION=name

서비스 제공자에서 이 매개변수는 파이프라인을 시작할 수 있는 별명 트랜잭션의 1 - 4자 이름을 지정합니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 TRANSACTION 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ # _ < >

URI=value

이 매개변수는 클라이언트가 웹 서비스에 액세스하기 위해 사용할 상대 또는 절대 URI를 지정합니다. CICS는 DFHLS2WS로 작성된 웹 서비스 바인딩 파일에서 URIMAP 자원을 생성할 때 지정되는 값을 사용합니다. 이 매개변수는 URIMAP 정의가 적용되는 URI의 경로 컴포넌트를 지정합니다.

USERID=id

서비스 제공자에서 이 매개변수는 웹 클라이언트가 사용할 수 있는 1 - 8자 사용자 ID를 지정합니다. 애플리케이션 생성 응답 또는 웹 서비스의 경우 이 사용자 ID 아래에 별명 트랜잭션이 첨부됩니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 USERID 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ #

WSBIND=*value*

웹 서비스 바인딩 파일의 완전한 z/OS UNIX 이름입니다. DFHLS2WS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 파일 확장자는 .wsbind입니다.

WSDL=*value*

웹 서비스 설명이 작성되는 파일의 완전한 z/OS UNIX 이름입니다. 웹 서비스 설명은 WSDL 1.1 스펙을 준수합니다. DFHLS2WS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 파일 확장자는 .wsdl입니다.

WSDL_1.1=*value*

웹 서비스 설명이 작성되는 파일의 완전한 z/OS UNIX 이름입니다. 웹 서비스 설명은 WSDL 1.1 스펙을 준수합니다. DFHLS2WS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 파일 확장자는 .wsdl입니다. 이 매개변수는 **WSDL** 매개변수와 동일한 결과를 생성하므로 둘 중 하나만 지정할 수 있습니다.

WSDL_2.0=*value*

웹 서비스 설명이 작성되는 파일의 완전한 z/OS UNIX 이름입니다. 웹 서비스 설명은 WSDL 2.0 스펙을 준수합니다. DFHLS2WS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 파일 확장자는 .wsdl입니다. 이 매개변수는 **WSDL** 또는 **WSDL_1.1** 매개변수와 함께 사용할 수 있습니다. 이 매개변수는 **MINIMUM-RUNTIME-LEVEL**이 2.0 이상으로 설정된 경우에만 사용할 수 있습니다.

WSDLCP={LOCAL|UTF-8|EBCDIC-CP-US}

WSDL 문서를 생성하는 데 사용되는 코드 페이지를 지정합니다.

LOCAL

WSDL 문서가 로컬 코드 페이지를 사용하여 생성되며 WSDL 문서에서는 인코딩 태그가 생성되지 않는 것으로 지정합니다.

UTF-8

WSDL 문서가 UTF-8 코드 페이지를 사용하여 생성되는 것으로 지정합니다. WSDL 문서에는 인코딩 태그가 생성됩니다. 이 옵션을 지정하는 경우 다른 플랫폼 간에 WSDL 문서를 복사할 때 인코딩이 올바르게 유지되는지 확인해야 합니다.

EBCDIC-CP-US

이 값은 WSDL 문서가 US EBCDIC 코드 페이지를 사용하여 생성되는 것으로 지정합니다. WSDL 문서에는 인코딩 태그가 생성됩니다.

WSDL-NAMESPACE=value

생성된 WSDL 문서에서 사용할 CICS의 네임스페이스를 지정합니다.

이 매개변수를 지정하지 않으면 CICS가 네임스페이스를 자동으로 생성합니다.

WSRR-CUSTOM-PropertyName=value

WSRR에서 WSDL 문서에 사용자 정의 메타데이터를 추가하려면 이 선택적 매개변수를 사용하십시오. **WSRR-CUSTOM-PropertyName=value** 쌍은 WSDL 문서에 추가되고 WSRR에 WSSR-CUSTOM 접두부 없이 나타납니다.

최대 255개 사용자 정의 **PropertyName=value** 쌍을 지정할 수 있습니다. **PropertyName=value** 쌍은 중복되거나 공백이어서는 안 됩니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-DESCRIPTION=value

공개되는 WSDL 문서를 설명하는 메타데이터를 지정하려면 이 선택적 매개변수를 사용하십시오.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-ENCODING=value

WSDL 문서의 문자 세트 인코딩을 지정하려면 이 선택적 매개변수를 사용하십시오. **WSRR-ENCODING** 매개변수가 지정되지 않으면 WSRR이 WSDL 문서에 지정된 값을 사용합니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-LOCATION=value

WSDL 문서의 위치를 식별하는 URI를 지정하려면 이 선택적 매개변수를 사용하십시오. 이 매개변수가 지정되지 않는 경우 URI의 기본값은 **WSDL** 매개변수에 지정된 파일 이름입니다. 예를 들어, **WSDL** 매개변수의 값이 `wsrr/example.wsdl`인 경우 **WSRR-LOCATION** 매개변수의 기본값은 `example.wsdl`입니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-PASSWORD=value

WSRR에 액세스하기 위해 비밀번호가 필요한 경우에는 이 선택적 매개변수를 사용하십시오.

WSRR-USERNAME 매개변수가 지정되는 경우 이 매개변수도 지정해야 합니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-SERVER={domain name:port number|IP address:port number}

WSRR(IBM WebSphere Service Registry and Repository) 서버의 위치를 지정하려면 이 매개변수를 사용하십시오. 이 매개변수가 지정되면 **WSRR** 매개변수 유효성 검증이 사용됩니다.

WSRR-USERNAME=value

WSRR에 액세스하기 위해 사용자 이름을 지정해야 하는 경우 이 선택적 매개변수를 사용하십시오. 이 사용자 이름은 WSRR이 소유자 특성을 설정하기 위해 사용합니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-VERSION={1|value}

WSRR에서 WSDL 문서의 버전 특성을 설정하려면 이 매개변수를 사용하십시오.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

기타 정보

- DFHLS2SC를 실행하는 사용자 ID는 UNIX System Services를 사용하도록 구성되어야 합니다. 사용자 ID는 CICS z/OS UNIX 파일 구조 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND**, **WSDL** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다.
- JCL의 최대 매개변수 길이는 100자입니다. 이 길이는 **STDPRM**문을 사용하여 늘릴 수 있습니다. 자세한 정보는 *z/OS UNIX System Services User Guide*를 참조하십시오.

예제

```
//LS2WS JOB 'accounting information',name,MSGCLASS=A
// SET QT='''
//JAVAPROG EXEC DFHLS2WS,
// TMPFILE=&QT.&SYSUID.&QT
//INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
REQMEM=DFH0XCP4
RESPMEM=DFH0XCP4
LANG=COBOL
LOGFILE=/u/exampleapp/wsbinding/example.log
MINIMUM-RUNTIME-LEVEL=2.1
MAPPING-LEVEL=2.1
CHAR-VARYING=COLLAPSE
PGMNAME=DFH0XCMN
URI=http://myserver.example.org:8080/exampleApp/example
PGMINT=COMMAREA
SOAPVER=1.1
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbinding/example.wsbinding
WSDL=/u/exampleapp/wsd1/example.wsd1
WSDL_2.0=/u/exampleapp/wsd1/example_20.wsd1
WSDLCP=LOCAL
WSDL-NAMESPACE=http://mywsdlnamespace
/*
```

DFHWS2LS: WSDL을 고급 언어로 변환:

DFHWS2LS 프로시저는 웹 서비스 설명으로부터 고급 언어 데이터 구조 및 웹 서비스 바인딩 파일을 생성합니다. CICS 애플리케이션 프로그램을 서비스 제공자로 노출하거나 서비스 요청자를 구성할 때 DFHWS2LS를 사용할 수 있습니다.

DFHWS2LS에 대한 작업 제어 명령문, 해당 기호 매개변수, 해당 입력 매개변수와 그에 대한 설명 및 예제 작업은 이 프로시저를 사용할 수 있도록 도움을 줍니다.

DFHWS2LS에 대한 작업 제어 명령문

JOB 작업을 시작합니다.

EXEC

프로시저 이름(DFHWS2LS)을 지정합니다.

INPUT.SYSUT1 DD

입력을 지정합니다. 입력 매개변수는 일반적으로 입력 스트림에 지정됩니다. 그러나 데이터 세트 또는 파티션된 데이터 세트의 멤버에 정의될 수 있습니다.

기호 매개변수

다음 기호 매개변수가 DFHWS2LS에서 정의됩니다.

JAVADIR=*path*

DFHWS2LS에서 사용되는 Java 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/에 추가되어 /usr/lpp/*path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

PATHPREFIX=*prefix*

다른 매개변수에서 사용되는 z/OS UNIX 디렉토리 경로를 확장하는 선택적 접두부를 지정합니다. 기본값은 비어 있는 문자열입니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

TMPDIR=*tmpdir*

DFHWS2LS가 임시 작업공간으로 사용하는 z/OS UNIX의 디렉토리 위치를 지정합니다. 작업이 실행되는 사용자 ID는 이 디렉토리에 대한 읽기 및 쓰기 권한이 있어야 합니다.

기본값은 /tmp입니다.

TMPPFILE=*tmpprefix*

DFHWS2LS가 임시 작업공간 파일의 이름을 구성하는 데 사용하는 접두부를 지정합니다.

기본값은 WS2LS입니다.

USSDIR=*path*

UNIX 시스템 서비스 파일 시스템에서 CICS TS 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/cicsts/에 추가되어 /usr/lpp/cicsts/*path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **USSDIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

SERVICE=*value*

이 매개변수는 IBM 지원 센터에서 지시하는 경우에만 사용합니다.

임시 작업공간

DFHWS2LS는 런타임 시 다음과 같은 세 개의 임시 파일을 작성합니다.

tmpdir/tmpprefix.in
tmpdir/tmpprefix.out
tmpdir/tmpprefix.err

여기서,

*tmpdir*은 **TMPDIR** 매개변수에 지정된 값입니다.

*tmpprefix*는 **TMPFILE** 매개변수에 지정된 값입니다.

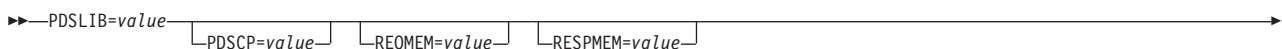
TMPDIR 및 **TMPFILE**이 지정되지 않은 경우 파일의 기본 이름은 다음과 같습니다.

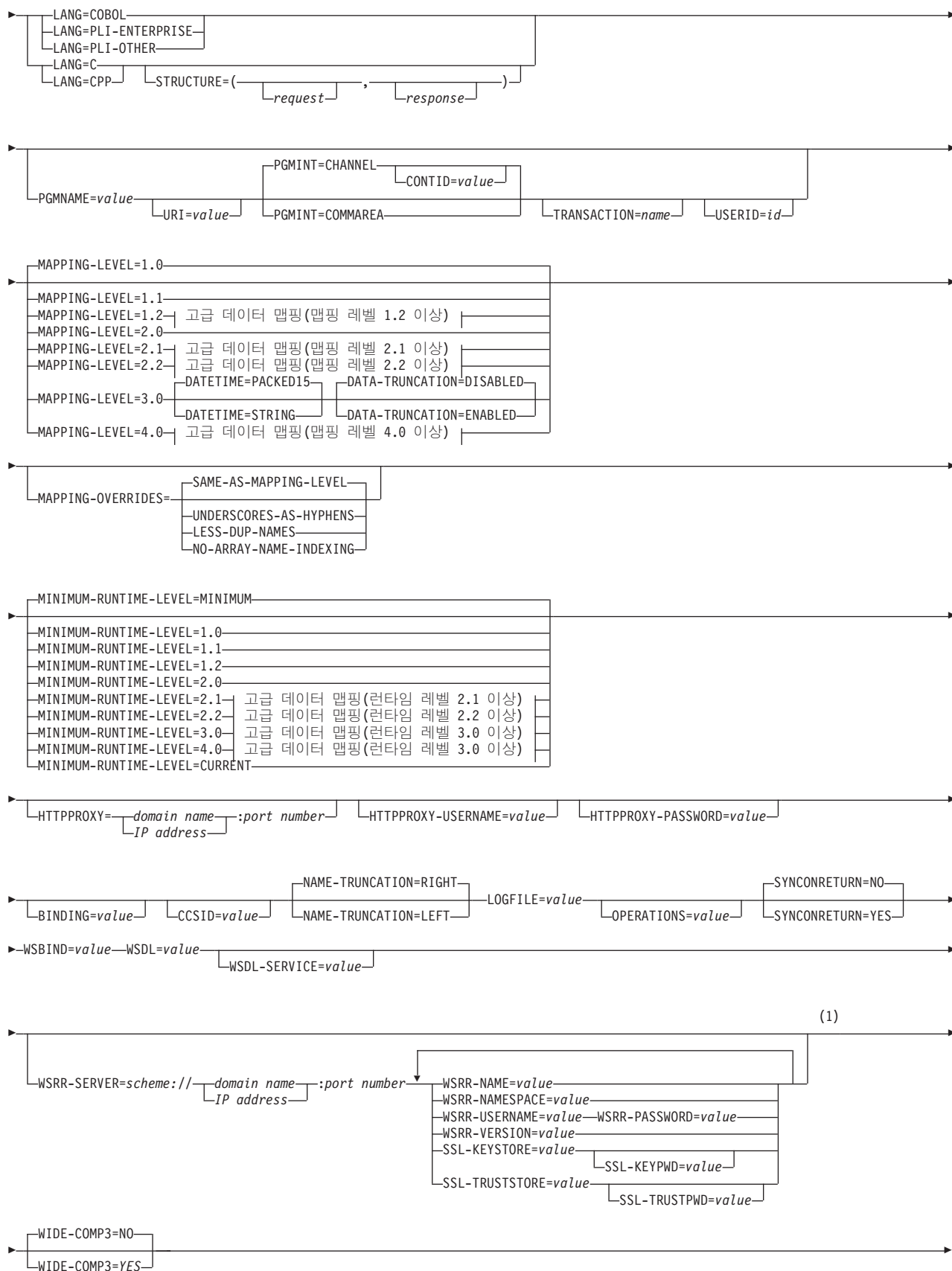
/tmp/WS2LS.in
/tmp/WS2LS.out
/tmp/WS2LS.err

중요사항: DFHWS2LS는 z/OS UNIX 파일 또는 데이터 세트 멤버에 대한 액세스를 잠그지 않습니다. 따라서 둘 이상의 DFHWS2LS 인스턴스가 동시에 실행되고 같은 임시 작업공간 파일을 사용하는 경우 한 작업이 사용 중인 작업공간 파일을 다른 작업이 겹쳐쓰는 것을 막을 수 없기 때문에 예상치 못한 장애가 발생할 수 있습니다.

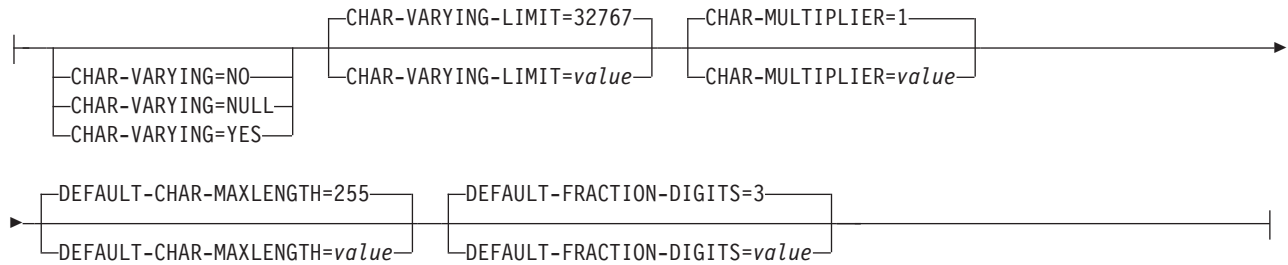
따라서 이러한 상황을 방지하는 이름 지정 규칙과 작동 프로시저를 고안해야 합니다. 예를 들어, 시스템 기호 매개변수 **SYSUID**를 사용하여 개별 사용자에게 고유한 작업공간 파일 이름을 생성할 수 있습니다. 이러한 임시 파일은 작업이 종료되기 전에 삭제됩니다.

DFHWS2LS의 입력 매개변수





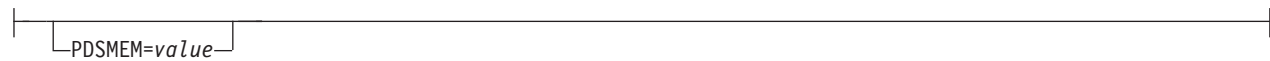
고급 데이터 맵핑(맵핑 레벨 1.2 이상):



고급 데이터 맵핑(맵핑 레벨 2.1 이상):



고급 데이터 맵핑(맵핑 레벨 2.2 이상):



고급 데이터 맵핑(런타임 레벨 2.1 이상):



고급 데이터 맵핑(런타임 레벨 3.0 이상):



참고:

- 1 **WSRR-SERVER** 매개변수가 설정될 때 지정할 수 있는 각 **WSRR** 매개변수는 한 번만 지정할 수 있습니다.

매개변수 사용

- 입력 매개변수는 순서에 관계 없이 지정할 수 있습니다.
- 각 매개변수는 행을 바꿔 시작해야 합니다.
- 매개변수는 사용 시 해당 연속 문자와 함께 열 72를 넘어가서는 안됩니다. 열 73 - 80은 공백이어야 합니다.

- 매개변수가 너무 길어 한 행에 맞지 않는 경우에는 행 끝에서 별표(*) 문자를 사용하여 매개변수가 다음 행에서 연속됨을 표시해야 합니다. 별표 앞의 모든 내용은 공백을 포함하여 매개변수의 일부로 간주됩니다. 예를 들어, 다음과 같습니다.

```
WSBIND=wsbinddir*
/app1
```

구문은 다음과 같습니다.

```
WSBIND=wsbinddir/app1
```

- 행에서 첫 번째 문자 위치의 # 문자는 주석 문자입니다. 이 행은 무시됩니다.
- 행에서 마지막 문자 위치의 쉼표는 선택적 선 구분 기호이며 무시됩니다.

매개변수 설명

BINDING=*value*

웹 서비스 설명에 한 개 이상의 <wsdl:바인딩> 요소가 포함된 경우, 이 매개변수를 사용하여 언어 구조 및 웹 서비스 바인딩 파일을 생성하는 데 사용할 요소를 지정하십시오. 웹 서비스 설명의 <wsdl:바인딩> 요소에 대해 사용되는 name 속성의 값을 지정하십시오.

CCSID=*value*

런타임에 애플리케이션 데이터 구조에서 문자 데이터를 인코딩하는 데 사용되는 CCSID를 지정합니다. 이 매개변수의 값은 **LOCALCCSID** 시스템 초기화 매개변수의 값을 대체합니다. *value*는 Java 및 z/OS 변환 서비스가 지원하는 EBCDIC CCSID 여야 합니다(z/OS Unicode Services User's Guide and Reference 참조). 이 매개변수를 지정하지 않으면 시스템 초기화 매개변수에 지정된 CCSID를 사용하여 애플리케이션 데이터 구조가 인코딩됩니다.

이 매개변수는 임의 맵핑 레벨에 사용할 수 있습니다. 그러나 생성된 파일을 CICS TS 3.1 리전에 배치하려면 APAR PK23547을 적용하여 웹 서비스 바인딩 파일을 설치할 수 있는 코드의 최소 런타임 레벨을 달성해야 합니다.

CHAR-MULTIPLIER={1**|***value***}**

맵핑 레벨이 1.2 이상인 경우 각 문자에 허용되는 바이트 수를 지정합니다. 이 매개변수의 *value*는 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다. 모든 비숫자 문자 기반 맵핑은 이 승수가 적용됩니다. 2진, 숫자, 존형 및 팩형 10진수 필드는 이 승수의 대상이 아닙니다.

이 매개변수는 예를 들어, DBCS 문자를 사용하려는 경우에 유용할 수 있습니다. 이 경우 런타임 시 모든 2바이트 문자 앞뒤로 잠재적인 SO 및 SI 문자를 위한 공간을 허용하기 위해 승수 3을 선택할 수 있습니다.

CCSID=1200(UTF-16 표시)을 설정하면 **CHAR-MULTIPLIER**에 대한 유일한 올바른 값은 2 또는 4입니다. UTF-16을 사용할 때 기본값은 2입니다. 애플리케이션 데이터가 1개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우

CHAR-MULTIPLIER=2를 사용하십시오. 애플리케이션 데이터가 2개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우 **CHAR-MULTIPLIER=4**를 사용하십시오.

참고: **CHAR-MULTIPLIER**를 1로 설정하면 DBCS 문자의 사용이 배제되지 않으며 이를 2로 설정하면 UTF-16 대리 쌍의 사용을 배제하지 않습니다. 그러나 와이드 문자가 주기적으로 사용되면 일부 올바른 값이 할당된 필드에 맞지 않게 됩니다. **CHAR-MULTIPLIER**의 값으로 더 큰 값을 사용할 수록 XML에서 올바른 문자 수보다 더 많은 문자를 할당된 필드에 저장할 수 있습니다. 해당 범위 제한을 주의하여 따라야 합니다.

CHAR-VARYING={NO|NULL|YES}

맵핑 레벨이 1.2 이상인 경우 가변 길이 문자 데이터가 맵핑되는 방식을 지정합니다. 가변 길이 2진 데이터 유형은 항상 컨테이너 또는 가변 구조에 맵핑됩니다. 이 매개변수를 지정하지 않은 경우 지정한 언어에 따라 기본 맵핑이 달라집니다. 다음 옵션을 선택할 수 있습니다.

아니오 가변 길이 문자 데이터가 고정 길이 문자열로 맵핑됩니다.

널 가변 길이 문자 데이터가 널 종료 문자열로 맵핑됩니다.

예 PL/I의 경우 가변 길이 문자 데이터가 CHAR VARYING 데이터 유형에 맵핑됩니다. COBOL, C, C++ 언어에서는 가변 길이 문자 데이터가 두 가지 관련 요소(데이터-길이 및 데이터)로 구성되는 해당 표시에 맵핑됩니다.

CHAR-VARYING-LIMIT={32767|value}

맵핑 레벨이 1.2 이상인 경우 언어 구조에 맵핑되는 가변 길이 문자 데이터와 2진 데이터의 최대 크기를 지정합니다. 문자 또는 2진 데이터가 이 매개변수에 지정된 값보다 큰 경우에는 컨테이너에 맵핑되고 생성된 언어 구조에서 컨테이너 이름이 사용됩니다. 값의 범위는 0 - 32,767바이트(기본값)입니다.

CONTID=value

서비스 제공자에서, SOAP 메시지를 나타내는 데 사용되는 상위 레벨 데이터 구조를 보유하는 컨테이너의 이름을 지정합니다.

CICS가 대상 애플리케이션 프로그램으로 전달하는 컨테이너의 길이는 요청 컨테이너의 길이와 응답 컨테이너의 길이 중 긴 값입니다.

DATA-TRUNCATION={DISABLED|ENABLED}

가변 길이 데이터가 고정 길이 필드 구조에서 용인되는지 여부를 지정합니다.

DISABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 거부하고 오류 메시지를 발행합니다.

ENABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 용인하고 누락된 데이터를 널값으로 처리합니다.

DATETIME={PACKED15|STRING}

<xsd:dateTime> 요소가 언어 구조로 매핑되는 방식을 지정합니다.

PACKED15

기본값은 모든 <xsd:dateTime> 요소가 시간소인으로 처리되고 CICS ABSTIME 형식으로 매핑되는 것입니다.

STRING

<xsd:dateTime> 요소가 텍스트로 처리됩니다.

DEFAULT-CHAR-MAXLENGTH={255|value}

매핑 레벨이 1.2 이상이고 웹 서비스 설명 문서에 길이가 나타나지 않는 경우 매핑 문자 내 문자 데이터의 기본 배열 길이를 지정합니다. 이 매개변수의 값은 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다.

DEFAULT-FRACTION-DIGITS={3|value}

XML 10진수 스키마 유형에 대해 사용할 기본 소수 자리수를 지정합니다. 기본값은 3입니다. COBOL의 경우 유효한 값은 0 - 17이고 매개변수 **WIDE-COMP3**를 사용 중인 경우 0 - 30입니다. C 또는 PLI의 경우 유효 범위는 0 - 30입니다.

HTTPPROXY={domain name:port number|IP address:port number}

인터넷에 있는 다른 WSDL 파일에 대한 참조가 WSDL에 포함되고 DFHWS2LS를 실행하고 있는 시스템이 프록시 서버를 사용하여 인터넷에 액세스하는 경우 도메인 이름 또는 IP 주소 및 프록시 서버의 포트 번호를 지정합니다. 예를 들어, 다음과 같습니다.

HTTPPROXY=proxy.example.com:8080

다른 경우에는 이 매개변수가 필요하지 않습니다.

HTTPPROXY-PASSWORD=value

DFHWS2LS를 실행하고 있는 시스템이 HTTP 프록시 서버를 사용하여 인터넷에 액세스하고 이 HTTP 프록시 서버가 기본 인증을 사용하는 경우에는 **HTTPPROXY-USERNAME**에 사용해야 하는 HTTP 프록시 비밀번호를 지정합니다. **HTTPPROXY**도 지정하는 경우에만 이 매개변수를 사용할 수 있습니다.

HTTPPROXY-USERNAME=value

DFHWS2LS를 실행하고 있는 시스템이 HTTP 프록시 서버를 사용하여 인터넷에 액세스하고 이 HTTP 프록시 서버가 기본 인증을 사용하는 경우에는 **HTTPPROXY-PASSWORD**에 사용해야 하는 HTTP 프록시 사용자 이름을 지정합니다. **HTTPPROXY**도 지정하는 경우에만 이 매개변수를 사용할 수 있습니다.

INLINE-MAXOCCURS-LIMIT={1|value}

maxOccurs 속성에 따라 인라인 가변 반복 콘텐츠가 사용되는지 여부를 지정합니다. 인라인으로 맵핑되는 가변 반복 콘텐츠는 생성된 언어 구조의 나머지와 함께 현재 컨테이너에 배치됩니다. 가변 반복 콘텐츠는 두 파트로, 즉 데이터 발생 횟수를 저장하는 카운터와 데이터의 각 발생을 저장하는 배열로 저장됩니다. 가변 반복 콘텐츠의 대체 맵핑은 컨테이너 기반 맵핑으로, 데이터 발생 횟수와 데이터가 배치되는 컨테이너의 이름을 저장합니다. 개별 컨테이너에 데이터를 저장하는 경우 성능에 영향을 주어 인라인 맵핑이 선호될 수 있습니다.

INLINE-MAXOCCURS-LIMIT 매개변수는 맵핑 레벨 2.1 이상에서만 사용할 수 있습니다. **INLINE-MAXOCCURS-LIMIT** 값은 0 - 32,767 범위의 양의 정수일 수 있습니다. 값이 0이면 인라인 맵핑이 사용되지 않음을 나타냅니다. 값이 1이면 선택적 요소가 인라인으로 맵핑됩니다. maxOccurs 속성의 value가

INLINE-MAXOCCURS-LIMIT의 value보다 크면 컨테이너 기반 맵핑을 사용하고 그렇지 않으면 인라인 맵핑을 사용합니다.

가변적으로 반복되는 목록이 인라인 맵핑되도록 할지 선택할 때 반복 데이터의 단일 항목 길이를 고려하십시오. 길이가 긴 인스턴스가 거의 없는 경우에는 컨테이너 기반 맵핑이 선호됩니다. 길이가 짧은 인스턴스가 많은 경우에는 인라인 맵핑이 선호됩니다.

LANG=COBOL

고급 언어 구조의 프로그래밍 언어가 COBOL인 것으로 지정합니다.

LANG=PLI-ENTERPRISE

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I인 것으로 지정합니다.

LANG=PLI-OTHER

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I 이외의 PL/I 레벨인 것으로 지정합니다.

LANG=C

고급 언어 구조의 프로그래밍 언어가 C인 것으로 지정합니다.

LANG=CPP

고급 언어 구조의 프로그래밍 언어가 C++인 것으로 지정합니다.

LOGFILE=value

DFHWS2LS가 그 활동 로그 및 추적 정보를 쓰는 파일의 완전한 z/OS UNIX 이름입니다. DFHWS2LS는 파일을 작성하지만 이미 존재하지 않는 경우 디렉토리 구조는 작성하지 않습니다.

일반적으로 이 파일을 사용하지는 않지만 DFHWS2LS와 관련된 문제점이 발생하는 경우 IBM 서비스 조직에서 이 파일을 요청할 수 있습니다.

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0}

웹 서비스 바인딩 파일 및 언어 구조를 생성할 때 DFHWS2LS가 사용하는 맵핑 레벨을 지정합니다. 다음 옵션을 선택할 수 있습니다.

- 1.0** CICS TS 3.1 맵핑 레벨을 사용하여 웹 서비스 바인딩 파일 및 언어 구조가 생성됩니다.
- 1.1** XML 속성 및 <목록>과 <유니온> 데이터 유형이 언어 구조로 맵핑됩니다. 최대 길이가 32,767바이트를 초과하는 문자 및 2진 데이터는 컨테이너로 맵핑됩니다. 컨테이너 이름이 언어 구조에서 작성됩니다.
- 1.2** **CHAR-VARYING** 및 **CHAR-VARYING-LIMIT** 매개변수를 사용하여 런타임 시 문자 데이터가 맵핑되고 처리되는 방식을 제어합니다. 이들 매개변수 중 어느 매개변수도 지정하지 않으면 최대 길이가 32,768바이트 미만인 2진 및 문자 데이터는 C++을 제외한 모든 언어에 대해 **VARYING** 구조로 맵핑됩니다. C++에서는 문자 데이터가 널 종료 문자열로 맵핑됩니다.
- 2.0** 이 맵핑 레벨은 CICS TS 3.2 리전 이상에서 언어 구조와 웹 서비스 바인딩 파일 간 맵핑에 대한 개선사항을 이용하기 위해 사용합니다.
- 2.1** 이 맵핑 레벨은 <xsd:any> 및 xsd:anyType 지원, 가변적으로 반복되는 컨테츠를 **INLINE-MAXOCCURS-LIMIT** 매개변수로 인라인 맵핑하는 옵션 및 <xsd:sequence>, <xsd:choice> 및 <xsd:all>에서 minOccurs="0"에 대한 지원을 얻기 위해 APAR PK59794가 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2 이후의 모든 리전에서 사용합니다.
- 2.2** 이 맵핑 레벨은 APAR PK69738이 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2 이후의 모든 리전에서 사용합니다. 이 맵핑 레벨에서는 다음 지원을 제공합니다.
 - 고정값을 가지는 요소
 - <xsd:choice> 요소에 대한 개선된 지원
 - 추상 데이터 유형
 - 추상 요소
 - 대체 그룹
- 3.0** 이 맵핑 레벨은 CICS TS 4.1 또는 이후 리전에서 사용합니다. 이 맵핑 레벨에서 시간소인을 CICS ABSTIME 형식으로 변환할 수 있습니다.
- 4.0** UTF-16을 사용하려는 경우 CICS TS 5.2 리전에 이 맵핑 레벨을 사용하십시오.

맵핑 레벨에 대한 자세한 정보는 CICS 지원을 위한 맵핑 레벨의 내용을 참조하십시오.

MAPPING-OVERRIDES={SAME-AS-MAPPING-LEVEL|UNDERSCORES-AS-HYPHENS|LESS-DUP-NAMES|NO-ARRAY-NAME-INDEXING}

언어 구조를 생성할 때 지정된 맵핑 레벨에 대해 기본 동작이 대체되는지 여부를 지정합니다.

참고: 모든 하위 옵션은 쉼표로 구분된 목록으로 사용할 수 있습니다. 이러한 옵션은 상호 배타적이지 않고 조합되며 순서가 지정되지 않습니다.

LESS-DUP-NAMES

이 매개변수는 이름 끝에 `_value`가 있는 비구조적 구조 필드 이름을 생성하여 필드를 직접 참조할 수 있습니다. 예를 들어, 다음 PLI 언어 구조에서 **MAPPING-OVERRIDES=LESS-DUP-NAMES**가 지정되는 경우 레벨 12 필드 `streetName`에는 접미부 `_value`가 추가됩니다.

```
09 streetName,
   12 streetName                                CHAR(255) VARYING
   UNALIGNED,
   12 filler                                    BIT (7),
   12 attr_nil_streetName_value                BIT (1),
```

결과적으로 구조는 다음과 같습니다.

```
09 streetName,
   12 streetName_value                          CHAR(255) VARYING
   UNALIGNED,
   12 filler                                    BIT (7),
   12 attr_nil_streetName_value                BIT (1),
```

SAME-AS-MAPPING-LEVEL

이 매개변수는 맵핑 레벨과 동일한 스타일의 언어 구조를 생성합니다. 이는 기본값입니다.

UNDERSCORES-AS-HYPHENS

COBOL에서만 사용됩니다. 이 매개변수는 WSDL 문서의 밑줄을 문자 `X`가 아닌 하이픈으로 변환하여 생성된 COBOL 언어 구조의 가독성을 향상시킵니다. 필드 이름 충돌이 발생하면 필드가 고유할 수 있도록 필드에 번호가 지정됩니다. 자세한 정보는 애플리케이션 개발에서 XML 스키마 대 COBOL 맵핑의 내용을 참조하십시오.

NO-ARRAY-NAME-INDEXING

엔터프라이즈 PL/I에서만 사용됩니다. 배열 내의 필드 이름은 상위 레벨 구조의 범위 내에서만 고유합니다.

MINIMUM - RUNTIME -

LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0|CURRENT}

웹 서비스 바인딩 파일이 배치될 수 있는 최소 CICS 런타임 환경을 지정합니다. 지정한 다른 매개변수와 일치하지 않는 레벨을 선택하면 오류 메시지를 받게 됩니다. 다음 옵션을 선택할 수 있습니다.

MINIMUM

지정한 매개변수의 경우 CICS의 런타임 레벨 중 가장 낮은 레벨이 자동으로 할당됩니다.

- 1.0** 생성된 웹 서비스 바인딩 파일이 APAR PK15904 및 PK23547이 적용되지 않은 CICS TS 3.1 리전에 배치됩니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 1.1** 생성된 웹 서비스 바인딩 파일이 최소 APAR PK15904가 적용된 CICS TS 3.1 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 1.1 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 1.2** 생성된 웹 서비스 바인딩 파일이 APAR PK15904 및 PK23547이 모두 적용된 CICS TS 3.1 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 1.2 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 3.2 이상 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 2.0 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.1** 생성된 웹 서비스 바인딩 파일이 APAR PK59794가 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2보다 상위 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 2.1 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.2** 생성된 웹 서비스 바인딩 파일이 APAR PK69738이 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2보다 상위 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 2.2 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 3.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 4.1 이상 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 3.0 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 4.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 5.2 이상 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 4.0 이하 맵핑 레벨을 사용할 수 있습니다. 이 레벨에서는 선택적 매개변수를 사용할 수 있습니다.

CURRENT

생성된 웹 서비스 바인딩 파일이 웹 서비스 바인딩 파일을 생성하기 위해 사용하는 런타임 레벨과 동일한 런타임 레벨에서 CICS 리전에 배치됩니다.

NAME-TRUNCATION={LEFT|RIGHT}

XML 요소의 이름을 왼쪽부터 자를지 또는 오른쪽부터 자를지 지정합니다. CICS 웹 서비스 지원은 XML 요소 이름을 지정된 고급 언어에 맞는 길이로 자르며 기본적으로 오른쪽부터 자릅니다.

OPERATIONS=value

웹 서비스 요청자 애플리케이션의 경우 웹 서비스 바인딩 파일을 생성하는 데 사용되는 웹 서비스 설명에서 유효한 <wsdl:Operation> 요소의 서브세트를 지정합니다. 각 <wsdl:Operation> 요소는 간격으로 분리되며 필요한 경우 목록이 둘 이상의 행에 걸쳐 이어질 수 있습니다. WSDL 1.1 및 WSDL 2.0 문서 둘 다에서 이 매개변수를 사용할 수 있습니다.

PDSCP=value

REQMEM 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에서 사용되는 코드 페이지를 지정합니다. 여기서 *value*는 CCSID 번호 또는 Java 코드 페이지 번호입니다. 이 매개변수가 지정되지 않으면 z/OS UNIX System Services 코드 페이지를 사용합니다. 예를 들어, **PDSCP=037**을 지정할 수 있습니다.

PDSLIB=value

생성된 고급 언어를 포함하는 파티션된 데이터 세트의 이름을 지정합니다. 요청 및 응답에 사용되는 데이터 세트 멤버는 각각 **REQMEM** 및 **RESPMEM** 매개변수에 지정됩니다.

PDSMEM=value

추상 데이터 유형에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 DFHWS2LS가 사용하는 1-에서 6-문자 접두부를 지정합니다. 접두부에 2자리 숫자를 추가하여 멤버 이름을 생성합니다.

추상 데이터 유형과 연관된 언어 구조의 이름을 지정하려면 맵핑 레벨 2.2 이상에서 이 매개변수를 사용하십시오. **PDSMEM** 매개변수가 생략된 경우 추상 데이터 유형에 대한 언어 구조는 **REQMEM** 매개변수의 값을 사용하여 이름 지정됩니다.

PGMINT={CHANNEL|COMMAREA}

서비스 제공자의 경우, CICS가 대상 애플리케이션 프로그램으로 데이터를 전달하는 방식을 지정합니다.

CHANNEL

CICS가 채널 인터페이스를 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

COMMAREA

CICS가 통신 영역을 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

이 매개변수는 DFHWS2LS의 출력이 서비스 요청자에서 사용될 때 무시됩니다.

대상 애플리케이션 프로그램이 요청을 처리한 경우에는 동일한 메커니즘으로 응답을 리턴해야 합니다. 통신 영역에서 요청을 수신한 경우 통신 영역에서 응답이 리턴되어야 합니다. 컨테이너에서 요청을 수신한 경우에는 컨테이너에서 응답이 리턴되어야 합니다. CICS가 대상 애플리케이션 프로그램으로 전달하는 통신 영역 또는 컨테이너의 길이는 요청 통신 영역 또는 컨테이너의 길이와 응답 통신 영역 또는 컨테이너의 길이 중 긴 값입니다.

PGMNAME=value

CICS PROGRAM 자원의 이름을 지정합니다.

DFHWS2LS가 서비스 제공자에서 사용될 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우 이 매개변수를 제공해야 합니다. 이 매개변수는 웹 서비스로 노출되는 애플리케이션 프로그램의 자원 이름을 지정합니다.

DFHWS2LS가 서비스 요청자에서 사용될 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우 이 매개변수를 생략하십시오.

REQMEM=value

웹 서비스 요청에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 DFHWS2LS가 사용하는 1-에서 6-문자 접두부를 지정합니다.

- 서비스 제공자의 경우 웹 서비스 요청은 애플리케이션 프로그램에 입력됩니다.
- 서비스 요청자에서 웹 서비스 요청은 애플리케이션 프로그램으로부터의 출력입니다.

DFHWS2LS는 각 조작에 대해 파티션된 데이터 세트 멤버를 생성합니다. 접두부에 2자리 숫자를 추가하여 멤버 이름을 생성합니다.

이 매개변수는 선택사항이기는 하지만 웹 서비스 설명에 요청의 정의가 포함되어 있는 경우에는 이 매개변수를 지정해야 합니다.

RESPMEM=value

웹 서비스 응답에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 DFHWS2LS가 사용하는 1-에서 6-문자 접두부를 지정합니다.

- 서비스 제공자의 경우 웹 서비스 응답은 애플리케이션 프로그램에서 출력됩니다.
- 서비스 요청자에서 웹 서비스 응답은 애플리케이션 프로그램에 대한 입력입니다.

DFHWS2LS는 각 조작에 대해 파티션된 데이터 세트 멤버를 생성합니다. 접두부에 2자리 숫자를 추가하여 멤버 이름을 생성합니다.

응답과 관련이 없으면 즉, 단방향 메시지인 경우에는 이 매개변수를 생략하십시오.

SSL-KEYSTORE=value

이 선택적 매개변수는 키 저장소 파일의 완전한 위치를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-KEYPWD=*value*

이 선택적 매개변수는 키 저장소의 비밀번호를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-TRUSTSTORE=*value*

이 선택적 매개변수는 신뢰 저장소 파일의 완전한 위치를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-TRUSTPWD=*value*

이 선택적 매개변수는 신뢰 저장소의 비밀번호를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

STRUCTURE=(*request,response*)

C 및 C++의 경우에만, 요청 및 응답 구조의 이름이 생성되는 방식을 지정합니다.

생성된 요청 및 응답 구조는 *requestnn* 및 *responsenn* 형식의 이름이며 여기에서 *nn*은 각 조작에 대해 구조를 구별하기 위해 생성되는 숫자 접미부입니다.

하나 또는 두 이름이 모두 생략되면 구조는 사용자가 지정한 **REQMEM** 및 **RESPMEM** 매개변수에서 생성된 파티션된 데이터 세트 멤버 이름과 같은 이름을 가집니다.

SYNCONRETURN={NO|YES}

원격 웹 서비스가 동기점을 발행할 수 있는지 여부를 지정합니다.

NO 원격 웹 서비스가 동기점을 발행할 수 없습니다. 이 값은 기본값입니다. 원격 웹 서비스가 동기점을 발행하면 ADPL 이상 종료와 함께 실패합니다.

YES 원격 웹 서비스가 동기점을 발행할 수 있습니다. YES를 선택하면 원격 웹 서비스에서 제어가 리턴되는 경우 원격 태스크가 개별 작업 단위로 커밋됩니다. 원격 웹 서비스가 복구 가능 자원을 갱신하고 리턴 후 실패하면 해당 자원에 대한 갱신을 백아웃할 수 없습니다.

TRANSACTION=*name*

서비스 제공자에서 이 매개변수는 파이프라인을 시작할 수 있는 별명 트랜잭션의 1 - 4자 이름을 지정합니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 TRANSACTION 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ # _ < >

URI=value

서비스 제공자에서, 이 매개변수는 클라이언트가 웹 서비스에 액세스하기 위해 사용하는 상대 URI를 지정합니다. CICS는 DFHWS2LS에 의해 작성된 웹 서비스 바인딩 파일에서 URIMAP 자원을 생성할 때 지정한 값을 사용합니다. 이 매개변수는 URIMAP 정의가 적용되는 URI의 경로 컴포넌트를 지정합니다.

서비스 요청자에서 대상 웹 서비스의 URI는 이 매개변수로 지정되지 않습니다. CICS는 서비스 요청자에 대해 URIMAP 자원을 생성하지 않습니다. 서비스 요청자가 대상 웹 서비스의 URI에 대해 클라이언트 요청을 수행할 때 사용할 서비스 요청자에 대한 자체 URIMAP 자원을 정의할 수 있습니다. 서비스 요청자가 **INVOKE SERVICE** 명령을 발행하면 CICS는 웹 서비스 설명에서 지정한 `wsdl:port`에서(있는 경우) `soap:address` 위치를 사용합니다. 이를 대체하고 **INVOKE SERVICE** 명령에 대해 URIMAP 또는 URI 옵션을 사용하여 다른 URI를 지정할 수 있습니다.

USERID=id

서비스 제공자에서 이 매개변수는 웹 클라이언트가 사용할 수 있는 1 - 8자 사용자 ID를 지정합니다. 애플리케이션 생성 응답 또는 웹 서비스의 경우 이 사용자 ID 아래에 별명 트랜잭션이 첨부됩니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 USERID 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3={NO|YES}

COBOL에서만 사용됩니다. COBOL 언어 구조에서 팩형 10진수 가변 길이의 최대 크기를 제어합니다.

아니오 DFHWS2LS는 COBOL 언어 구조 유형 COMP-3을 생성할 때 팩형 10진수 가변 길이를 18로 제한합니다. 팩형 10진수 크기가 18보다 크면 지정된 유형은 총 18숫자로 제한됨을 나타내는 메시지 DFHPI9022W가 발행됩니다.

예 DFHWS2LS는 COBOL 언어 구조 유형 COMP-3을 생성할 때 최대 크기 31을 지원합니다.

WSADDR-EPR-ANY={TRUE|FALSE}

CICS가 WS-Addressing 엔드포인트 참조(EPR)를 언어 구조의 해당 컴포넌트 파

트로 변환하는지 또는 EPR을 <xsd:any> 유형으로 처리하는지 지정합니다. EPR을 <xsd:any> 유형으로 처리하는 것은 **WSACONTEXT BUILD** API가 EPR XML을 직접 사용할 수 있음을 의미합니다.

FALSE

DFHWS2LS가 일반적으로 작동하며 XML을 고급 언어 구조로 변환합니다.

TRUE

이 옵션을 TRUE로 설정하는 것은 런타임 시 CICS가 전체 EPR을 <xsd:any> 유형으로 처리하고 EPR XML을 애플리케이션이 참조할 수 있는 컨테이너에 배치함을 의미합니다. 애플리케이션은 **WSACONTEXT BUILD** API와 함께 EPR XML을 사용하여 주소 지정 컨텍스트에서 EPR을 구성할 수 있습니다.

이 매개변수는 런타임 레벨 3.0 이상에서만 사용할 수 있습니다.

WSBIND=*value*

웹 서비스 바인딩 파일의 완전한 z/OS UNIX 이름입니다. DFHWS2LS는 파일을 작성하지만 이미 존재하지 않는 경우 디렉토리 구조는 작성하지 않습니다. 기본 파일 확장자는 .wsbind입니다.

WSDL=*value*

웹 서비스 설명을 포함하는 파일의 완전한 z/OS UNIX 이름입니다. WSRR을 사용하여 WSDL 문서를 검색 중인 경우 이 매개변수는 파일 시스템에서 WSDL 문서의 로컬 사본이 작성될 위치를 지정합니다.

WSDL-SERVICE=*value*

한 개의 바인딩 요소에 대해 둘 이상의 서비스 요소가 웹 서비스 설명에 포함되어 있는 경우 사용되는 wsdl:Service 요소를 지정합니다. **BINDING** 매개변수에 대한 값을 지정하는 경우 이 매개변수에 대해 지정하는 서비스 요소는 지정된 바인딩 요소와 일치해야 합니다. WSDL 1.1 또는 WSDL 2.0 문서 둘 다에서 이 매개변수를 사용할 수 있습니다.

WSRR-NAME=*value*

WSRR에서 검색할 WSDL 문서의 이름을 지정합니다. 이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-NAMESPACE=*value*

WSRR에서 검색할 WSDL 문서의 네임스페이스를 지정합니다. **WSRR-SERVER** 매개변수가 지정되는 경우에 선택적으로 이 매개변수를 사용하여 **WSRR-NAME** 매개변수에 지정된 WSDL 문서 이름을 완전하게 확인할 수 있습니다.

WSRR-PASSWORD=*value*

WSRR에 액세스하기 위해 비밀번호가 필요한 경우에는 이 선택적 매개변수를 사용하십시오.

WSRR-USERNAME 매개변수가 지정되는 경우 이 매개변수도 지정해야 합니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-SERVER=*{domain name:port number|IP address:port number}*

WSRR(IBM WebSphere Service Registry and Repository) 서버의 위치를 지정하려면 이 매개변수를 사용하십시오. 이 매개변수가 지정되면 WSRR 매개변수 유효성 검증이 사용됩니다.

WSRR-USERNAME=*value*

WSRR에 액세스하기 위해 사용자 이름을 지정해야 하는 경우 이 선택적 매개변수를 사용하십시오. 이 사용자 이름은 WSRR이 소유자 특성을 설정하기 위해 사용됩니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-VERSION=*value*

WSRR에서 검색할 WSDL 문서의 버전을 지정합니다. 이 매개변수는 **WSRR-SERVER** 매개변수가 지정되는 경우에만 사용할 수 있습니다.

XML-ONLY=*{TRUE|FALSE}*

CICS가 SOAP 메시지의 XML을 애플리케이션 데이터로 변환하는지 여부를 지정합니다. **XML-ONLY** 매개변수를 사용하여 XML 자체를 처리하는 웹 서비스 애플리케이션을 작성하십시오.

TRUE

CICS는 XML으로의 어떠한 변환도 수행하지 않습니다. 이 서비스 요청자 또는 제공자 애플리케이션은 XML과 고급 언어 사이에서 데이터를 맵핑하기 위해 DFHWS-BODY 컨테이너의 콘텐츠를 사용하여 직접 작업해야 합니다.

FALSE

CICS가 XML을 고급 언어로 변환합니다.

이 매개변수는 런타임 레벨 2.1 이상에서만 사용할 수 있습니다.

기타 정보

- DFHLS2SC를 실행하는 사용자 ID는 UNIX System Services를 사용하도록 구성되어야 합니다. 사용자 ID는 CICS z/OS UNIX 파일 구조 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND**, **WSDL** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다.
- JCL의 최대 매개변수 길이는 100자입니다. 이 길이는 **STDPARM** 명령문을 사용하여 늘릴 수 있습니다. 자세한 정보는 z/OS UNIX System Services 사용자 안내서의 내용을 참조하십시오.

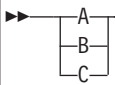
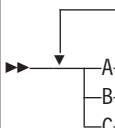
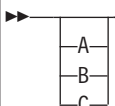
예제

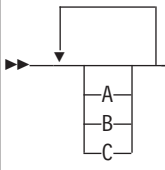
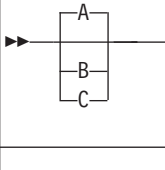
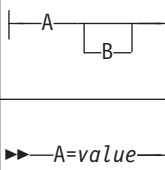
```
//WS2LS JOB  'accounting information',name,MSGCLASS=A
//  SET QT='''
//JAVAPROG EXEC DFHWS2LS,
// TMPFILE=&QT.&SYSUID.&QT
//INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
REQMEM=CPYBK1
RESPMEM=CPYBK2
LANG=COBOL
LOGFILE=/u/exampleapp/wsbind/example.log
MAPPING-LEVEL=3.0
MAPPING-OVERRIDES=UNDERSCORES-AS-HYPHENS
CHAR-VARYING=NULL
INLINE-MAXOCCURS-LIMIT=2
PGMNAME=DFH0XCMN
URI=exampleApp/example
PGMINT=COMMAREA
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbind/example.wsbind
WSDL=/u/exampleapp/wsd1/example.wsd1
/*
```

구문 표기법:

구문 표기법은 CICS 명령, 자원 정의 및 기타 여러 항목에 지정할 수 있는 속성 또는 옵션의 사용 가능한 조합을 지정합니다.

구문 표기법에서 사용되는 규칙:

표기법	설명
	필수 대체사항 세트를 선언합니다. 반드시 표시되는 값 중 하나를(하나만) 지정해야 합니다.
	필수 대체사항 세트를 선언합니다. 표시되는 값 중 최소한 하나를 지정해야 합니다. 하나 이상의 값을 어떤 순서로든 지정할 수 있습니다.
	선택적 대체사항 세트를 선언합니다. 표시되는 값 중 하나를 지정하거나 아무 것도 지정하지 않을 수 있습니다.

표기법	설명
	선택적 대체사항 세트를 선언합니다. 아무 값도 지정하지 않거나 표시되는 값 중 하나 또는 하나 이상을 어떤 순서로든 지정할 수 있습니다.
	선택적 대체사항 세트를 선언합니다. 표시되는 값 중 하나를 지정하거나 아무 것도 지정하지 않을 수 있습니다. A 는 아무 값도 지정하지 않은 경우에 사용되는 기본값입니다.
이름 이름: 	구문 표기법의 이름 지정된 섹션에 대한 참조입니다.
$A=value$	A는 표시된 대로 정확히 입력해야 하는 문자를 선언합니다. <i>value</i>는 적절한 값을 지정해야 하는 변수를 선언합니다.

CICS 지원에 대한 �핑 레벨:

�핑은 언어 구조와 XML 스키마 간에 정보가 변환되는 방법을 지정하는 규칙 세트입니다. 가장 정교한 �핑을 활용하려면 CICS 지원에서 **MAPPING-LEVEL** 매개변수를 최신 레벨로 설정하는 것이 좋습니다.

각 �핑 레벨은 이전 �핑의 기능을 상속하며 가장 높은 �핑 레벨이 가장 우수한 기능을 제공합니다. 가장 높은 �핑 레벨이 런타임에 데이터 변환을 보다 강력하게 제어하며 특정 데이터 유형 및 XML 요소 지원에 대한 제한사항을 제거합니다.

이전에 해당 레벨에서 사용한 애플리케이션을 다시 배치하려는 경우 **MAPPING-LEVEL** 매개변수를 이전 레벨로 설정할 수 있습니다.

�핑 레벨 3.0

�핑 레벨 3.0은 CICS TS 4.1 리전과 호환됩니다.

이 �핑 레벨은 다음 지원을 제공합니다.

- DFHSC2LS 및 DFHWS2LS가 xsd:dateTime 데이터 유형을 CICS ASKTIME 형식에 �핑합니다.

- DFHLS2WS가 하나의 컨테이너가 아닌 많은 컨테이너를 사용하는 애플리케이션에서 WSDL 문서와 웹 서비스 바인딩을 생성할 수 있습니다.
- 고정 길이 데이터 구조로 설명하는 잘린 데이터 허용. 이 동작은 CICS 지원에 **DATA-TRUNCATION** 매개변수를 사용하여 설정할 수 있습니다.

맵핑 레벨 2.2 이상

맵핑 레벨 2.2는 APAR PK69738 이상이 적용된 CICS TS 3.2 리전과 호환됩니다.

맵핑 레벨이 2.2 이상인 경우 DFHSC2LS 및 DFHWS2LS는 다음 XML 맵핑을 지원합니다.

- 요소의 고정 값
- 대체 그룹
- 추상 데이터 유형
- XML 스키마 <sequence> 요소는 <choice> 요소 내부에 중첩될 수 있습니다.

DFHSC2LS 및 DFHWS2LS는 다음 XML 맵핑에 대한 강화된 지원을 제공합니다.

- 추상 요소
- XML 스키마 <choice> 요소

맵핑 레벨 2.1 이상

맵핑 레벨 2.1은 APAR PK59794 이상이 적용된 CICS TS 3.2 리전과 호환됩니다.

이 맵핑 레벨에는 새 **INLINE-MAXOCCURS-LIMIT** 매개변수와 **CHAR-VARYING** 매개변수의 새 값으로 변수 콘텐츠를 처리하는 방법에 대한 강화된 제어가 포함됩니다.

맵핑 레벨 2.1 이상의 경우 DFHSC2LS 및 DFHWS2LS는 XML 맵핑에 대해 다음과 같은 새로운 또는 향상된 지원을 제공합니다.

- XML 스키마 <any> 요소
- xsd:anyType 유형
- 추상 요소 허용
- **INLINE-MAXOCCURS-LIMIT** 매개변수
- minOccurs 속성

INLINE-MAXOCCURS-LIMIT 매개변수는 가변 반복 목록이 인라인으로 맵핑되는지 여부를 지정합니다. 가변 반복 콘텐츠 인라인 맵핑에 대한 자세한 정보는 가변 요소 배열을 참조하십시오.

XML 스키마 <sequence>, <choice>, <all> 요소에서 minOccurs 속성에 대한 지원이 강화되었습니다. minOccurs="0"인 경우 CICS 지원은 이들 요소를 minOccurs="0" 속성이 모든 하위 요소의 속성인 것처럼 처리합니다.

맵핑 레벨이 2.1 이상인 경우 DFHLS2SC 및 DFHLS2WS는 다음 XML 맵핑을 지원합니다.

- COBOL 및 PL/I의 FILLER 필드는 무시됨
- **CHAR-VARYING** 매개변수에 대한 COLLAPSE 값
- **CHAR-VARYING** 매개변수에 대한 BINARY 값

COBOL 및 PL/I의 FILLER 필드는 무시됩니다. 즉, 생성된 XML 스키마에 나타나지 않으며 런타임에 데이터 구조에서 적절한 갭이 유지됩니다.

COLLAPSE로 인해 CICS가 텍스트 필드에서 후속 공백을 무시합니다.

BINARY는 2진 필드에 대한 지원을 제공합니다. 이 값은 COBOL을 XML 스키마로 변환할 때 유용합니다. 이 옵션은 SBCS 문자 배열에서만 사용할 수 있으며 `xsd:string` 필드가 아닌 고정 길이 `xsd:base64Binary` 필드로 배열이 맵핑될 수 있습니다.

맵핑 레벨 1.2 이상

맵핑 레벨 1.2는 CICS TS 3.1 이상 리전과 호환됩니다.

일괄처리 도구의 이 추가 매개변수로 런타임에 문자 및 2진 데이터가 변환되는 방식을 보다 강력하게 제어할 수 있습니다.

- **CHAR-VARYING**
- **CHAR-VARYING-LIMIT**
- **CHAR-MULTIPLIER**
- **DEFAULT-CHAR-MAXLENGTH**

DFHSC2LS 또는 DFHWS2LS에서 **CHAR-MULTIPLIER** 매개변수를 사용하려는 경우, 문자 데이터에 필요한 공백을 계산하기 위해 이 매개변수의 값을 사용한 후 다음 규칙이 적용됨에 유의하십시오.

- DFHSC2LS 및 DFHWS2LS는 다음 맵핑을 제공합니다.
 - 최대 길이가 32 767바이트보다 긴 가변 길이 문자 데이터 유형은 컨테이너에 맵핑됩니다. **CHAR-VARYING-LIMIT** 매개변수를 사용하여 낮은 한계를 설정할 수 있습니다. 언어 구조에서 컨테이너 이름을 저장하기 위한 16바이트 필드가 작성됩니다. 런타임에는 문자 데이터가 컨테이너에 저장되고 컨테이너 이름은 언어 구조에 배치됩니다.
 - 최대 길이가 32 768바이트 미만인 가변 길이 문자 데이터 유형은 C/C++ 및 엔터프라이즈 PL/I를 제외한 모든 언어의 경우 **VARYING** 구조에 맵핑됩니다. C/C++의 경우에는 이러한 데이터 유형이 널 종료 문자열에 맵핑되고 엔터프라이즈 PL/I의 경우에는 이러한 데이터 유형이 **VARYINGZ** 구조에 맵핑됩니다. **CHAR-VARYING** 매개변수를 사용하여 가변 길이 문자 데이터가 맵핑되는 방식을 선택할 수 있습니다.

- 최대 길이가 32 768바이트 미만인 가변 길이 2진 데이터가 모든 언어의 **VARYING** 구조에 맵핑됩니다. 최대 길이가 32 768바이트 이상인 경우에는 데이터가 컨테이너에 맵핑됩니다. 언어 구조에서 컨테이너 이름을 저장하기 위한 16 바이트 필드가 작성됩니다. 런타임에는 2진 데이터가 컨테이너에 저장되고 컨테이너 이름은 언어 구조에 배치됩니다.

XML 스키마에 연관된 길이가 없는 문자 데이터 유형이 있는 경우에는 DFHWS2LS 또는 DFHSC2LS에서 **DEFAULT-CHAR-MAXLENGTH** 매개변수를 사용하여 기본 길이를 지정할 수 있습니다.

DFHLS2SC 및 DFHLS2WS는 다음 맵핑을 제공합니다.

- 문자 필드는 **xsd:string** 데이터 유형에 맵핑되며 런타임에 고정 길이 필드 또는 널 종료 문자열로 처리될 수 있습니다. **CHAR-VARYING** 매개변수를 사용하여 PL/I를 제외한 모든 언어에 대해 런타임에 가변 길이 문자 데이터가 처리되는 방식을 선택할 수 있습니다.
- 데이터의 최대 길이가 32 767바이트보다 크거나 길이가 정의되지 않은 경우 **Base64Binary** 데이터 유형이 컨테이너에 맵핑됩니다. 데이터 길이가 32 767 이하인 경우에는 **base64Binary** 데이터 유형이 모든 언어의 **VARYING** 구조에 맵핑됩니다.

맵핑 레벨 1.1 이상:

맵핑 레벨 1.1은 CICS TS 3.1 이상 리전과 호환됩니다.

이 맵핑 레벨은 특히 **maxLength** 및 **minLength** 속성이 XML 스키마에서 다른 값으로 정의된 가변 길이의 데이터를 맵핑하는 경우 XML 문자 및 2진 데이터 유형의 향상된 맵핑을 제공합니다. 데이터는 다음과 같이 처리됩니다.

- 16 MB보다 큰 고정 길이의 문자 및 2진 데이터 유형은 PL/I를 제외한 모든 언어의 경우 컨테이너에 맵핑됩니다. PL/I의 경우에는 32 767바이트보다 큰 고정 길이 문자 및 2진 데이터 유형이 컨테이너에 맵핑됩니다. 언어 구조에서 컨테이너 이름을 저장하기 위한 16바이트 필드가 작성됩니다. 런타임에는 고정 길이 데이터가 컨테이너에 저장되고 컨테이너 이름은 언어 구조에 배치됩니다.

컨테이너는 길이가 가변적이므로 컨테이너에 맵핑되는 고정 길이 데이터가 XML 스키마 또는 웹 서비스 설명에 지정된 고정 길이와 일치시키기 위해 잘리거나 공백 또는 널로 채워지지 않습니다. 데이터 길이가 중요한 경우 길이를 확인하는 애플리케이션을 작성하거나 CICS 리전에서 유효성 검증을 활성화할 수 있습니다. SOAP 및 XML 유효성 검증 모두 성능에 중요한 영향을 미칩니다.

- XML 스키마 **<list>** 및 **<union>** 데이터 유형은 문자 필드에 맵핑됩니다.
- 스키마 정의 XML 속성은 무시되지 않고 맵핑됩니다. 각 XML 요소에 최대 255개 속성이 허용됩니다. 자세한 정보는 XML 속성에 대한 지원을 참조하십시오.

- xsi:nil 속성은 지원됩니다. 자세한 정보는 XML 속성에 대한 지원을 참조하십시오.

맵핑 레벨 1.1 전용

맵핑 레벨 1.1은 CICS TS 3.1 이상 리전과 호환됩니다.

이 맵핑 레벨은 특히 maxLength 및 minLength 속성이 XML 스키마에서 다른 값으로 정의된 가변 길이의 데이터를 맵핑하는 경우 XML 문자 및 2진 데이터 유형의 향상된 맵핑을 제공합니다. 데이터는 다음과 같이 처리됩니다.

- 가변 길이 2진 데이터 유형은 컨테이너에 맵핑됩니다. 언어 구조에서 컨테이너 이름을 저장하기 위한 16바이트 필드가 작성됩니다. 런타임에는 2진 데이터가 컨테이너에 저장되고 컨테이너 이름은 언어 구조에 배치됩니다.
- 최대 길이가 32 767바이트보다 긴 가변 길이 문자 데이터 유형은 컨테이너에 맵핑됩니다. 언어 구조에서 컨테이너 이름을 저장하기 위한 16바이트 필드가 작성됩니다. 런타임에는 문자 데이터가 컨테이너에 저장되고 컨테이너 이름은 언어 구조에 배치됩니다.
- 16MB 미만의 고정 길이를 갖는 문자 및 2진 데이터 유형은 PL/I를 제외한 모든 언어의 경우 고정 길이 필드에 맵핑됩니다. PL/I의 경우 32 767바이트 이하의 고정 길이 문자 및 2진 데이터 유형은 고정 길이 필드에 맵핑됩니다.
- CICS는 base64Binary 형식이 아닌 hexBinary 형식으로 데이터를 인코딩하고 디코딩합니다. XML 스키마의 Base64Binary 데이터 유형은 언어 구조에서 필드에 맵핑됩니다. 필드의 크기는 다음 공식을 사용하여 계산됩니다. $4 \times (\text{ceil}(z/3))$ 여기서,
 - z 는 XML 스키마에서 데이터 유형의 길이입니다.
 - $\text{ceil}(x)$ 은 x 보다 크거나 같은 가장 작은 정수입니다.

z 의 길이가 24 566바이트보다 큰 경우에는 결과 언어 구조가 컴파일되지 않습니다. 24 566바이트보다 큰 base64Binary 데이터가 있는 경우에는 맵핑 레벨 1.2를 사용하는 것이 좋습니다. 맵핑 레벨이 1.2인 경우 언어 구조에서 필드를 사용하지 않고 base64Binary 데이터를 컨테이너에 맵핑할 수 있습니다.

맵핑 레벨 1.0 전용

맵핑 레벨 1.0은 CICS TS 3.1 이상 리전과 호환됩니다.

다음 제한사항이 적용됩니다. 이 제한사항은 이후 맵핑 레벨에서 수정되었습니다.

- DFHSC2LS 및 DFHWS2LS는 XML 스키마의 문자 및 2진 데이터 유형을 언어 구조의 고정 길이 필드에 맵핑합니다. 다음 부분 XML 스키마를 예로 들 수 있습니다.

```
<xsd:element name="example">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
```

```

        <xsd:maxLength value="33000"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:element>

```

해당 부분 XML 스키마는 COBOL 언어 구조에 다음과 같이 나타납니다.

```
15 example      PIC X(33000)
```

- CICS는 base64Binary 형식이 아닌 hexBinary 형식으로 데이터를 인코딩하고 디코딩합니다. DFHSC2LS 및 DFHWS2LS는 Base64Binary 데이터를 고정 길이 문자 필드에 맵핑하며 해당 콘텐츠는 애플리케이션 프로그램이 인코딩 또는 디코딩해야 합니다.
- DFHSC2LS 및 DFHWS2LS는 처리 중에 XML 속성을 무시합니다.
- DFHLS2SC 및 DFHLS2WS는 언어 구조에서 문자 및 2진 필드를 고정 길이 필드로 해석하며 해당 필드를 maxLength 속성을 갖는 XML 요소에 맵핑합니다. 사용할 수 있는 데이터가 충분하지 않은 경우 런타임에 언어 구조의 필드가 공백 또는 널로 채워집니다.

고급 언어 및 XML 스키마 맵핑:

CICS 지원을 사용하여 고급 언어 구조와 XML 스키마 또는 WSDL 문서 사이에 맵핑을 생성하십시오. CICS 보조는 고급 언어 데이터 구조 또는 그 반대에서 XML 스키마 또는 WSDL 문서도 생성합니다.

DFHSC2LS 및 DFHLS2SC 유틸리티 프로그램은 총괄적으로 CICS XML 보조로 알려집니다. DFHWS2LS 및 DFHLS2WS 유틸리티 프로그램은 총괄적으로 CICS 웹 서비스 보조로 알려집니다.

- DFHLS2SC 및 DFHLS2WS는 고급 언어 구조를 XML 스키마 및 WSDL 문서에 각각 맵핑합니다.
- DFHSC2LS 및 DFHWS2LS는 XML 스키마 및 WSDL 문서를 고급 언어 구조에 맵핑합니다.

두 개의 맵핑이 대칭적이지 않습니다.

- DFHLS2SC 또는 DFHLS2WS를 가진 언어 데이터 구조를 처리하고 보완적 유틸리티 프로그램을 가진 결과로 생기는 XML 스키마 또는 WSDL 문서(각각 DFHSC2LS 또는 DFHWS2LS)를 처리하는 경우, 최종 데이터 구조가 원래의 구조와 같은 것으로 예상하지 마십시오. 그러나, 최종 데이터 구조는 논리적으로 원래의 구조에 해당합니다.
- DFHSC2LS 또는 DFHWS2LS를 가진 XML 스키마 또는 WSDL 문서를 처리하고 보완적 유틸리티 프로그램을 가진 결과로 생기는 언어 구조(각각 DFHLS2SC 또는 DFHLS2WS)를 처리하는 경우, 최종 XML 스키마 또는 WSDL 문서의 XML 스키마가 원래와 같은 것으로 예상하지 마십시오.

- 일부 경우에, DFHSC2LS 및 DFHWS2LS는 DFHLS2SC 및 DFHLS2WS가 지원하지 않는 언어 구조를 생성합니다.

CICS가 지원하는 언어 컴파일러에서 구현되는 것처럼 언어의 규칙에 따라 DFHLS2SC 및 DFHLS2WS가 처리하는 언어 구조를 코드화해야 합니다.

CICS 지원 사용 시 데이터 맵핑 제한사항:

CICS는 WSDL 버전 1.1 또는 2.0을 준수하는 XML 스키마 또는 WSDL 문서와 고급 언어 구조 간의 양방향 데이터 맵핑을 지원합니다. 그러나 여기에는 특정 제한사항이 적용됩니다. 이러한 제한사항은 DFHWS2LS 및 DFHSC2LS 도구에만 적용되며 맵핑 레벨에 따라 다릅니다.

모든 맵핑 레벨의 제한사항

- 리터럴 인코딩을 사용하는 SOAP 바인딩만 지원됩니다. 따라서 `use` 속성의 값을 `literal`로 설정해야 합니다. `use="encoded"`는 지원되지 않습니다.
- 데이터 유형 정의는 XML 스키마 정의 언어(XSD)를 사용하여 인코딩해야 합니다. 스키마에서, SOAP 메시지에서 사용되는 데이터 유형이 명시적으로 선언되어야 합니다.
- 웹 서비스 설명에서 일부 키워드의 길이가 제한됩니다. 예를 들어, 조작, 바인딩, 파트 이름은 255자로 제한됩니다. 경우에 따라 최대 조작 이름의 길이가 약간 짧을 수 있습니다.
- 웹 서비스 설명에서 정의된 SOAP 결함은 무시됩니다. 서비스 제공자 애플리케이션으로 SOAP 결함 메시지를 보내려면 **EXEC CICS SOAPFAULT** 명령을 사용하십시오.
- DFHWS2LS 및 DFHSC2LS는 특정 범위의 단일 `<xsd:any>` 요소만 지원합니다. 예를 들어, 다음 스키마 단편은 지원되지 않습니다.

```
<xsd:sequence>
  <xsd:any/>
  <xsd:any/>
</xsd:sequence>
```

여기서 `<xsd:any>`는 필요한 경우 `minOccurs` 및 `maxOccurs`를 지정할 수 있습니다. 예를 들어, 다음 스키마 단편이 지원됩니다.

```
<xsd:sequence>
  <xsd:any minOccurs="2" maxOccurs="2"/>
</xsd:sequence>
```

- 주기적 참조는 지원되지 않습니다. 예를 들어, 유형 A가 유형 B를 포함하고 유형 B가 다시 유형 A를 포함하는 경우입니다.
- `<xsd:choice>`, `<xsd:sequence>`, `<xsd:group>` 또는 `<xsd:all>` 요소와 같은 그룹 요소에서는 반복이 지원되지 않습니다. 예를 들어, 다음 스키마 단편은 지원되지 않습니다.

```

<xsd:choice maxOccurs="2">
  <xsd:element name="name1" type="string"/>
</xsd:choice>

```

예외는 맵핑 레벨이 2.1 이상인 경우입니다. 이 경우 이 요소에서 maxOccurs="1" 과 minOccurs="0"이 지원됩니다.

- DFHSC2LS 및 DFHWS2LS는 맵핑 레벨 2.2 이상에서 상위 요소 또는 유형이 추상으로 정의된 경우를 제외하고, xsi:type 속성 또는 대체 그룹에서 XML 스키마에서 있는 선언된 데이터 유형과 요소에서 파생된 SOAP 메시지의 데이터 유형과 요소를 지원하지 않습니다.
- <xsd:choice> 요소 내부에 임베드된 <xsd:sequence> 및 <xsd:group> 요소는 맵핑 레벨 2.2 이하에서는 지원되지 않습니다. <xsd:choice> 요소 내부에 임베드된 <xsd:choice> 및 <xsd:all> 요소는 지원되지 않습니다.

맵핑 레벨 1.1 이상의 지원 향상

맵핑 레벨이 1.1 이상인 경우 DFHWS2LS는 다음 XML 요소 및 요소 유형에 대한 지원을 제공합니다.

- <xsd:list> 요소
- <xsd:union> 요소
- xsd:anySimpleType 유형
- <xsd:attribute> 요소. 맵핑 레벨이 1.0인 경우에는 이 요소가 무시됩니다.

맵핑 레벨 2.1 이상의 지원 향상

맵핑 레벨이 2.1 이상인 경우 DFHWS2LS는 다음 XML 요소와 요소 속성을 지원합니다.

- <xsd:any> 요소
- xsd:anyType 유형
- 추상 요소. 이전 맵핑 레벨에서는 추상 요소가 상속 계층 구조의 비터미널 유형으로만 지원됩니다.
- <xsd:all>, <xsd:choice>, <xsd:sequence> 요소의 maxOccurs 및 minOccurs 속성 (maxOccurs="1" 및 minOccurs="0"인 경우만 해당)
- "FILLER" 필드(COBOL)와 "*" 필드(PL/I)가 억제됩니다. 이 필드는 생성된 WSDL에 나타나지 않으며 런타임에 데이터 구조에서 적절한 갭이 유지됩니다.

맵핑 레벨 2.2 이상의 지원 향상

맵핑 레벨이 2.2 이상인 경우 DFHSC2LS 및 DFHWS2LS는 <xsd:choice> 요소에 대한 향상된 지원을 제공합니다. 예를 들어, <xsd:choice> 요소에서 최대 255개 옵션을 지원합니다. <xsd:choice> 지원에 대한 자세한 정보는 141 페이지의 『<xsd:choice>에 대한 지원』을 참조하십시오.

맵핑 레벨이 2.2 이상인 경우 CICS 지원이 다음 XML 맵핑을 지원합니다.

- 대체 그룹
- 요소의 고정 값
- 추상 데이터 유형

맵핑 레벨이 2.2 이상인 경우 <xsd:choice> 요소에 임베드된 <xsd:sequence> 및 <xsd:group> 요소가 지원됩니다. 예를 들어, 다음 스키마 단편이 지원됩니다.

```
<xsd:choice>
  <xsd:element name="name1" type="string"/>
  <xsd:sequence/>
</xsd:choice>
```

SOAP 메시지의 상위 요소 또는 유형이 추상으로 정의되는 경우, DFHSC2LS 및 DFHWS2LS는 XML 스키마에서 선언된 데이터 유형 및 요소에서 파생되는 데이터 유형과 요소를 지원합니다.

맵핑 레벨 3.0 이상의 지원 향상

맵핑 레벨이 3.0 이상인 경우 CICS 지원이 다음 맵핑 개선사항을 지원합니다.

- DFHSC2LS 및 DFHWS2LS가 xsd:dateTime 데이터 유형을 CICS ASKTIME 형식에 맵핑합니다.
- DFHLS2WS가 하나의 컨테이너가 아닌 많은 컨테이너를 사용하는 애플리케이션에서 WSDL 문서와 웹 서비스 바인딩을 생성할 수 있습니다.
- 고정 길이 데이터 구조로 설명하는 잘린 데이터 허용. 이 동작은 CICS 지원에 **DATA-TRUNCATION** 매개변수를 사용하여 설정할 수 있습니다.

COBOL 대 XML 스키마 맵핑:

DFHLS2SC 및 DFHLS2WS 유틸리티 프로그램은 COBOL 데이터 구조와 XML 스키마 정의 간의 맵핑을 지원합니다.

COBOL 이름은 다음 규칙에 따라 XML 이름으로 변환됩니다.

1. 중복된 이름은 하나 이상의 숫자를 추가하여 고유한 이름이 됩니다.

예를 들어, year의 두 인스턴스는 year와 year1이 됩니다.

2. 하이픈은 밑줄 문자로 바뀝니다. 연속 하이픈 문자열은 연속 밑줄로 바뀝니다.

예를 들어, current-user--id는 current_user__id가 됩니다.

3. 하이픈으로 분리되고 대문자만 포함하는 이름의 세그먼트는 소문자로 변환됩니다.

예를 들어, CA-REQUEST-ID는 ca_request_id가 됩니다.

4. 숫자로 시작하는 이름에는 선행 밑줄 문자가 추가됩니다.

예를 들어, 9A-REQUEST-ID는 _9a_request_id가 됩니다.

CICS는 다음 표에 따라 COBOL 데이터 설명 요소를 스키마 요소로 맵핑합니다. 표에 없는 COBOL 데이터 설명 요소는 DFHLS2SC 또는 DFHLS2WS에서 지원되지 않습니다. 다음 제한사항도 적용됩니다.

- 레벨 번호가 66 및 77인 데이터 설명 항목은 지원되지 않습니다. 레벨 번호가 88인 데이터 설명 항목은 무시됩니다.
- 데이터 설명 항목에 대한 다음 절은 지원되지 않습니다.

OCCURS DEPENDING ON

OCCURS INDEXED BY

REDEFINES

RENAMES; 즉 레벨 66

DATE FORMAT

- 데이터 설명 항목에 대한 다음 절은 무시됩니다.

BLANK WHEN ZERO

JUSTIFIED

VALUE

- SIGN 절 SIGN TRAILING은 지원됩니다. SIGN 절 SIGN LEADING은 DFHLS2SC 또는 DFHLS2WS에서 지정된 맵핑 레벨이 1.2 이상인 경우에만 지원됩니다.
- SEPARATE CHARACTER는 SIGN TRAILING 및 SIGN LEADING 절 둘 다에 대해 1.2 이상의 맵핑 레벨에서 지원됩니다.
- USAGE 절에서 다음 구문이 지원되지 않습니다.

OBJECT REFERENCE

POINTER

FUNCTION-POINTER

PROCEDURE-POINTER

- USAGE 절에서 다음 구문은 1.2 이상의 맵핑 레벨에서 지원됩니다.

COMPUTATIONAL-1

COMPUTATIONAL-2

- DISPLAY 및 COMPUTATIONAL-5 데이터 설명 항목에 지원되는 유일한 PICTURE 문자는 9, S, Z입니다.
- PACKED-DECIMAL 데이터 설명 항목에 지원되는 PICTURE 문자는 9, S, V, Z입니다.
- 편집된 숫자 데이터 설명 항목에 지원되는 유일한 PICTURE 문자는 9와 Z입니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 NULL로 설정된 경우, 문자 배열은 xsd:string으로 맵핑되고 널(null) 종료 문자열로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 BINARY로 설정된 경우, 문자 배열은 xsd:base64Binary로 맵핑되고 2진 데이터로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 COLLAPSE로 설정된 경우, 문자열의 후행 공백이 무시됩니다.

COBOL 데이터 설명	스키마 simpleType
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> 여기서 $m = n$
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> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.
PIC S9(<i>z</i>) DISPLAY 여기서 $5 \leq z \leq 9$	<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> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.
PIC S9(<i>z</i>) DISPLAY 여기서 $9 < z$	<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> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.

COBOL 데이터 설명	스키마 simpleType
PIC 9 DISPLAY PIC 99 DISPLAY PIC 999 DISPLAY PIC 9999 DISPLAY	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="n"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.
PIC 9(z) DISPLAY 여기서 $5 \leq z \leq 9$	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="n"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.
PIC 9(z) DISPLAY 여기서 $9 < z$	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="n"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.
PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY 여기서 $n \leq 4$ 입니다.	<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 여기서 $5 \leq n \leq 9$ 입니다.	<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 여기서 $9 < n$ 입니다.	<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 여기서 $n \leq 4$ 입니다.	<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 여기서 $5 \leq n \leq 9$ 입니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>

COBOL 데이터 설명	스키마 simpleType
PIC 9(<i>n</i>) COMP PIC 9(<i>n</i>) COMP-4 PIC 9(<i>n</i>) COMP-5 PIC 9(<i>n</i>) BINARY 여기서 9 < <i>n</i> 입니다.	<xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType>
PIC S9(<i>m</i>)V9(<i>n</i>) COMP-3	<xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="p"/> <xsd:fractionDigits value="n"/> </xsd:restriction> </xsd:simpleType> 여기서 $p = m + n$ 입니다.
PIC 9(<i>m</i>)V9(<i>n</i>) COMP-3 PIC S9(<i>m</i>) COMP-3 DATETIME=PACKED15인 경우 맵핑 레벨 3.0에서 지원됨	<xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="p"/> <xsd:fractionDigits value="n"/> <xsd:minInclusive value="0"/> </xsd:restriction> </xsd:simpleType> 여기서 $p = m + n$ 입니다. <xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType> 시간소인 형식은 CICS ABSTIME입니다.
PIC S9(<i>m</i>)V9(<i>n</i>) DISPLAY 맵핑 레벨 1.2 이상에서 지원됨	<xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="p"/> <xsd:fractionDigits value="n"/> </xsd:restriction> </xsd:simpleType> 여기서 $p = m + n$ 입니다.
COMP-1 맵핑 레벨 1.2 이상에서 지원됨	<xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType>
COMP-2 맵핑 레벨 1.2 이상에서 지원됨	<xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType>

XML 스키마 대 COBOL 맵핑:

DFHSC2LS 및 DFHWS2LS 유틸리티 프로그램은 XML 스키마 정의와 COBOL 데이터 구조 간 맵핑을 지원합니다.

CICS 지원은 다음 규칙을 사용하여 스키마 요소 이름에서 COBOL 변수의 고유하고 올바른 이름을 생성합니다.

1. COBOL 예약어 앞에는 'X'가 붙습니다.

예를 들어, DISPLAY는 XDISPLAY가 됩니다.

2. A-Z, a-z, 0-9 또는 하이픈 이외의 문자는 'X'로 대체됩니다.

예를 들어, monthly_total은 monthlyXtotal이 됩니다. **MAPPING-OVERRIDES** 매개변수를 사용하여 다른 문자 처리 방법을 변경할 수 있습니다. 예를 들어, 값 **UNDERScores-AS-HYPHENS**를 설정하는 경우, XML의 모든 밑줄은 X가 아닌 하이픈으로 변환됩니다. 따라서 monthly_total은 monthly-total이 됩니다.

3. 마지막 문자가 하이픈이면 'X'로 대체됩니다.

예를 들어, ca-request-는 ca-requestX가 됩니다.

4. 스키마가 변수의 카디널리티가 가변적이고(즉, minOccurs 및 maxOccurs가 다른 값으로 xsd:element에 지정됨) 스키마 요소 이름이 23자보다 긴 것으로 지정하면 해당 길이에서 잘립니다.

스키마가 변수의 카디널리티가 고정적이고 스키마 요소 이름이 28자보다 긴 것으로 지정하면 해당 길이에서 잘립니다.

5. 동일한 범위의 중복된 이름은 이름의 두 번째 및 후속 인스턴스에 하나 또는 두 개 숫자를 추가하여 고유한 이름이 됩니다.

예를 들어, year의 세 가지 인스턴스는 year, year1, year2가 됩니다.

6. 문자열 _cont 또는 _num에는 다섯 개 문자가 예약되어 스키마가 변수의 카디널리티가 가변적인 것으로 지정하는 경우(즉, minOccurs 및 maxOccurs 속성이 다른 값으로 지정되는 경우) 사용됩니다.

자세한 정보는 132 페이지의 『요소의 가변 배열』을 참조하십시오.

7. 속성의 경우 이전 규칙이 요소 이름에 적용됩니다. 접두부 attr-이 요소 이름에 추가되고 그 다음에 -value 또는 -exist가 나타납니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다. 자세한 정보는 137 페이지의 『XML 속성에 대한 지원』을 참조하십시오.

nillable 속성은 특수 규칙을 갖습니다. 접두부 attr-이 추가되지만 요소 이름 처음에 nil-도 추가됩니다. 요소 이름 다음에는 -value가 나타납니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다.

결과 이름의 총 길이는 30자 이하입니다.

DFHSC2LS 및 DFHWS2LS는 다음 표에 따라 지정된 맵핑 레벨을 사용하여 COBOL 데이터 설명 요소에 스키마 유형을 맵핑합니다. 다음 사항에 주의하십시오.

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 NULL로 설정되면 가변 길이 문자 데이터가 널 종료 문자열에 맵핑되고 널 종료자에 추가 문자가 할당됩니다.

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 YES 로 설정되면 가변 길이 문자 데이터가 관련 두 요소(길이 피드와 데이터 필드)에 맵 핁됩니다. 예를 들어, 다음과 같습니다.

```
<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"/>
```

다음에 맵핑됩니다.

```
15 textString-length PIC S9999 COMP-5 SYNC
15 textString          PIC X(10000)
```

스키마 단순 유형	COBOL 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 2.0 이하: 지원되지 않음 맵핑 레벨 2.1: 지원됨
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: PIC X(255)
<pre><xsd:simpleType> <xsd:restriction base="xsd:type" <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. <pre>string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY hexBinary</pre>	모든 맵핑 레벨: PIC X(z)

스키마 단순 유형	COBOL 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:type" </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. duration date time gDay gMonth gYear gMonthDay gYearMonth	모든 맵핑 레벨: PIC X(32)
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.2 이하: PIC X(32) 맵핑 레벨 2.0 이상: PIC X(40) 맵핑 레벨 3.0 이상: PIC S9(15) COMP-3 형식은 CICS ABSTIME입니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. byte unsignedByte	모든 맵핑 레벨: PIC X DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC S9999 COMP-5 SYNC 또는 PIC S9999 DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC 9999 COMP-5 SYNC 또는 PIC 9999 DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC S9(18) COMP-3

스키마 단순 유형	COBOL 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC S9(9) COMP-5 SYNC 또는 PIC S9(9) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC 9(9) COMP-5 SYNC 또는 PIC 9(9) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC S9(18) COMP-5 SYNC 또는 PIC S9(18) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC 9(18) COMP-5 SYNC 또는 PIC 9(18) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="m"> <xsd:fractionDigits value="n"> </xsd:fractionDigits> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC 9(<i>p</i>)V9(<i>n</i>) COMP-3 여기서 $p = m - n$입니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC X DISPLAY 값 x'00'은 false를 의미하고 x'01'은 true를 의미합니다.
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: PIC X(255)
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: PIC X(255)

스키마 단순 유형	COBOL 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> 길이가 정의되지 않는 경우.	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1: PIC X(y) 여기서 $y = 4 \times (\text{ceil}(z/3))$. $\text{ceil}(x)$은 x보다 크거나 같은 가장 작은 정수입니다. 맵핑 레벨 1.2 이상: PIC X(z) 여기서 길이는 고정적입니다. PIC X(16) 길이가 정의되지 않은 경우. 이 필드는 2진 데이터를 저장하는 컨테이너의 16바이트 이름을 갖습니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: PIC X(32) 맵핑 레벨 1.2 이상: COMP-1
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: PIC X(32) 맵핑 레벨 1.2 이상: COMP-2

테이블에 표시되는 스키마 유형 중 일부는 minInclusive 및 maxInclusive 패킷에 지정되는 값(있는 경우)에 따라 COMP-5 SYNC 또는 DISPLAY의 COBOL 형식에 맵핑됩니다.

- 부호 있는 유형(short, int, long)의 경우 다음이 지정되면 DISPLAY가 사용됩니다.

```
<xsd:minInclusive value="-a"/>
<xsd:maxInclusive value="a"/>
```

여기서 a 는 '9' 문자열입니다.

- 부호 없는 유형(unsignedShort, unsignedInt, unsignedLong)의 경우 DISPLAY는 다음이 지정된 경우 사용됩니다.

```
<xsd:minInclusive value="0"/>
<xsd:maxInclusive value="a"/>
```

여기서 a 는 '9' 문자열입니다.

다른 모든 값이 지정되거나 값이 지정되지 않은 경우에는 COMP-5 SYNC가 사용됩니다.

C 및 C++ 대 XML 스키마 맵핑:

DFHLS2SC 및 DFHLS2WS 유틸리티 프로그램은 C 및 C++ 데이터 유형과 XML 스키마 정의 간의 맵핑을 지원합니다.

다음 규칙에 따라 C 및 C++ 이름이 XML 이름으로 변환됩니다.

1. XML 요소 이름에서 유효하지 않은 문자는 'X'로 바꿉니다.

예를 들어, monthly-total은 monthlyXtotal이 됩니다.

2. 중복된 이름은 하나 이상의 숫자를 추가하여 고유한 이름이 됩니다.

예를 들어, year의 두 인스턴스는 year와 year1이 됩니다.

DFHLS2SC 및 DFHLS2WS는 다음 표에 따라 C 및 C++ 데이터 유형을 스키마 요소로 맵핑합니다. 표에 없는 C 및 C++ 유형은 DFHLS2SC 또는 DFHLS2WS에서 지원되지 않습니다. `_Packed` 지정자는 구조에 대해 지원됩니다. 다음과 같은 제한사항이 적용됩니다.

- 헤더 파일은 최상위 레벨 struct 인스턴스를 포함해야 합니다.
- 자체를 멤버로 포함하는 구조 유형은 선언할 수 없습니다.
- 다음 C 및 C++ 데이터 유형은 지원되지 않습니다.

decimal
long double
wchar_t(C++만 해당)

- 다음 문자가 헤더 파일에 있는 경우에는 무시됩니다.

저장영역 클래스 지정자:

auto
register
static
extern
mutable

규정자

const
volatile
_Export(C++만 해당)

함수 지정자

inline(C++만 해당)
virtual(C++만 해당)

초기값

- 헤더 파일에 다음 항목이 포함되지 않아야 합니다.

공용체

클래스 선언

열거 데이터 유형

포인터 유형 변수

템플릿 선언

사전 정의된 매크로. 즉, 해당 이름이 두 개의 밑줄 문자(_)로 시작하고 끝나는 매크로

행 연속 시퀀스(줄 바꾸기 문자 바로 앞에 오는 \ 기호)

프로토타입 함수 선언자

프리프로세서 지시문

비트 필드

`_cdecl`(또는 `_cdecl`) 키워드(C++만 해당)

- 애플리케이션 프로그래머는 `int`가 4바이트로 맵핑되도록 32비트 컴파일러를 사용해야 합니다.
- 다음의 C++ 예약 키워드는 지원되지 않습니다.

`explicit`

`using`

`namespace`

`typename`

`typeid`

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 `NULL`로 설정된 경우, 문자 배열은 `xsd:string`으로 맵핑되고 널(null) 종료 문자 열로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 `BINARY`로 설정된 경우, 문자 배열은 `xsd:base64Binary`로 맵핑되고 2진 데이터로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 `COLLAPSE`로 설정된 경우, 문자열에 대해 `<xsd:whiteSpace value="collapse"/>`가 생성됩니다.

C 및 C++ 데이터 유형	스키마 <code>simpleType</code>
<code>char[z]</code>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType> </pre>

C 및 C++ 데이터 유형	스키마 simpleType
char[8] DATETIME=PACKED15인 경우 맵핑 레벨 3.0 이상에서 지원됨	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre> 시간소인 형식은 CICS ABSTIME입니다.
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++만 해당)	<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>
float 맵핑 레벨 1.2 이상에서 지원됨	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
double 맵핑 레벨 1.2 이상에서 지원됨	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

XML 스키마 대 C 및 C++ 맵핑:

DFHSC2LS 및 DFHWS2LS 유틸리티 프로그램은 각 웹 서비스 설명에 포함된 XML 스키마 정의와 C 및 C++ 데이터 유형 간의 맵핑을 지원합니다.

CICS 지원은 다음 규칙을 사용하여 스키마 요소 이름에서 C 및 C++ 변수의 고유하고 올바른 이름을 생성합니다.

1. A-Z, a-z, 0-9 또는 _ 이외의 문자는 'X'로 대체됩니다.

예를 들어, `monthly-total`은 `monthlyXtotal`이 됩니다.

2. 첫 번째 문자가 영문자가 아니면 선행 'X'로 대체됩니다.

예를 들어, `_monthlysummary`는 `Xmonthlysummary`가 됩니다.

3. 스키마 요소 이름이 50자보다 길면 해당 길이로 잘립니다.
4. 동일한 범위의 중복된 이름은 하나 이상의 숫자를 추가하여 고유하게 됩니다.

예를 들어, `year`의 두 인스턴스는 `year`와 `year1`이 됩니다.

5. 문자열 `_cont` 또는 `_num`에는 다섯 개 문자가 예약되어 스키마가 변수의 카디널리티가 가변적인 것으로 지정하는 경우(즉, `minOccurs` 및 `maxOccurs`가 `xsd:element`에서 지정되는 경우) 사용됩니다.

자세한 정보는 132 페이지의 『요소의 가변 배열』을 참조하십시오.

6. 속성의 경우 이전 규칙이 요소 이름에 적용됩니다. 접두부 `attr_`이 요소 이름에 추가되고 그 다음에 `_value` 또는 `_exist`가 옵니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다.

`nillable` 속성은 특수 규칙을 갖습니다. 접두부 `attr_`이 추가되지만 요소 이름 처음에 `nil-`도 추가됩니다. 요소 이름 다음에는 `-value`가 나타납니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다.

결과 이름의 총 길이는 57자 이하입니다.

FHSC2LS 및 DFHWS2LS는 다음 표에 따라 스키마 유형을 C 및 C++ 데이터 유형에 맵핑합니다. 다음 규칙도 적용됩니다.

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 NULL로 설정되면 가변 길이 문자 데이터가 널 종료 문자열에 맵핑되고 널 종료자에 추가 문자가 할당됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 YES로 설정되면 가변 길이 문자 데이터가 관련 두 요소(길이 피드와 데이터 필드)에 맵핑됩니다.

스키마 simpleType	C 및 C++ 데이터 유형
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 2.0 이하: 지원되지 않음 맵핑 레벨 2.1 이상: 지원됨
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: char[255]
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. - string - normalizedString - token - Name - NMTOKEN - language - NCName - ID - IDREF - ENTITY - hexBinary	모든 맵핑 레벨: char[z]

스키마 simpleType	C 및 C++ 데이터 유형
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. - duration - date - decimal - time - gDay - gMonth - gYear - gMonthDay - gYearMonth	모든 맵핑 레벨: char[32]
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.2 이하: char[32] 맵핑 레벨 2.0 이상: char[40] 맵핑 레벨 3.0 이상: char[8] 시간소인 형식은 CICS ABSTIME입니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: signed char
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: char
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: short
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: unsigned short
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: char[33]
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: int

스키마 simpleType	C 및 C++ 데이터 유형
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: unsigned int
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: long long
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: unsigned long long
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: bool(C++만 해당) short(C만 해당)
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: char[255]
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: char[255]
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType> <xsd:simpleType> <xsd:restriction base="xsd:base64binary"> </xsd:restriction> </xsd:simpleType></pre> 길이가 정의되지 않는 경우	맵핑 레벨 1.1 이하: char[y] 여기서 $y = 4 \times (\text{ceil}(z/3))$. $\text{ceil}(x)$은 x보다 크거나 같은 가장 작은 정수입니다. 맵핑 레벨 1.2 이상: char[z] 여기서 길이는 고정적입니다. char[16] 길이가 정의되지 않은 경우 2진 데이터를 저장하는 컨테이너의 이름입니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: char[32] 맵핑 레벨 1.2 이상: float(*)

스키마 simpleType	C 및 C++ 데이터 유형
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.0 이하: char[32] 맵핑 레벨 1.2 이상: double(*)

PL/I 대 XML 스키마 맵핑:

DFHLS2SC 및 DFHLS2WS 유틸리티 프로그램은 PL/I 데이터 구조와 XML 스키마 정의 간의 맵핑을 지원합니다. Enterprise PL/I 컴파일러와 이전 PL/I 컴파일러가 다르므로 두 개의 언어 옵션인 PLI-ENTERPRISE 및 PLI-OTHER가 지원됩니다.

다음 규칙에 따라 PL/I 이름이 XML 이름으로 변환됩니다.

1. XML 요소 이름에서 유효하지 않은 문자는 'x'로 바꿉니다.

예를 들어 monthly\$total은 monthlyxtotal이 됩니다.

2. 중복된 이름은 하나 이상의 숫자를 추가하여 고유한 이름이 됩니다.

예를 들어, year의 두 인스턴스는 year와 year1이 됩니다.

DFHLS2SC 및 DFHLS2WS는 다음 표에 따라 PL/I 데이터 유형을 스키마 요소로 맵핑합니다. 표에 없는 PL/I 유형은 DFHLS2SC 또는 DFHLS2WS에서 지원되지 않습니다. 다음 제한사항도 적용됩니다.

- COMPLEX 속성이 있는 데이터 항목은 지원되지 않습니다.
- FLOAT 속성이 있는 데이터 항목은 1.2 이상의 맵핑 레벨에서 지원됩니다. 엔터프라이즈 PL/I FLOAT IEEE는 지원되지 않습니다.
- VARYING 및 VARYINGZ 순수 DBCS 문자열은 1.2 이상의 맵핑 레벨에서 지원됩니다.
- DECIMAL(p,q)로 지정된 데이터 항목은 $p \geq q$ 인 경우에만 지원됩니다.
- BINARY(p,q)로 지정된 데이터 항목은 $q = 0$ 인 경우에만 지원됩니다.
- 데이터 항목에 PRECISION 속성이 지정된 경우 무시됩니다.
- PICTURE 문자열은 지원되지 않습니다.
- ORDINAL 데이터 항목은 FIXED BINARY(7) 데이터 유형으로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 NULL로 설정된 경우, 문자 배열은 xsd:string으로 맵핑되고 널(null) 종료 문자열로 처리됩니다. 엔터프라이즈 PL/I의 경우에는 이 맵핑이 적용되지 않습니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 BINARY로 설정된 경우, 문자 배열은 xsd:base64Binary로 맵핑되고 2진 데이터로 처리됩니다.

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 **COLLAPSE**로 설정된 경우, 문자열에 대해 `<xsd:whiteSpace value="collapse"/>`가 생성됩니다.

DFHLS2SC 및 DFHLS2WS는 PL/I의 채우기 알고리즘을 완전히 구현하지 않으므로, 데이터 구조에서 명시적으로 채우기 바이트를 선언해야 합니다. DFHLS2SC 및 DFHLS2WS는 채우기 바이트가 누락되었음을 발견하는 경우 메시지를 발행합니다. 각 최상위 레벨 구조는 double-word 경계에서 시작하며, 구조의 각 바이트가 해당 경계로 맵핑되어야 합니다. 다음 코드 단편을 살펴보겠습니다.

```
3 FIELD1 FIXED BINARY(7),
3 FIELD2 FIXED BINARY(31),
3 FIELD3 FIXED BINARY(63);
```

이 예제의 경우:

- FIELD1은 1바이트 길이이고, 모든 경계로 할당 가능합니다.
- FIELD2는 4바이트 길이이고, 전자(full word) 경계로 할당 가능합니다.
- FIELD3은 8바이트 길이이고, 더블 워드(double word) 경계로 할당 가능합니다.

엔터프라이즈 PL/I 컴파일러는 다음과 같은 순서로 필드를 할당합니다.

1. FIELD3이 가장 강력한 경계 요구사항을 가지므로 먼저 할당됩니다.
2. FIELD2는 FIELD3 바로 앞에 전자(fullword) 경계로 할당됩니다.
3. FIELD1은 FIELD3 바로 앞에 바이트 경계로 할당됩니다.

마지막으로, 전체 구조를 전자(fullword) 경계로 할당하기 위해 컴파일러가 FIELD1 바로 앞에 채우기 3바이트를 삽입합니다.

DFHLS2WS는 상응하는 채우기 바이트를 삽입하지 않으므로, DFHLS2WS에서 구조를 처리하기 전에 명시적으로 선언되어야 합니다. 예를 들어, 다음과 같습니다.

```
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);
```

또는 모든 필드를 unaligned로 선언하도록 구조를 변경하고, 해당 구조를 사용하는 애플리케이션을 다시 컴파일할 수 있습니다. PL/I 구조 메모리 할당 요구사항에 대한 자세한 정보는 *Enterprise PL/I Language Reference*를 참조하십시오.

PL/I 데이터 설명	스키마
FIXED BINARY (<i>n</i>) 여기서 $n \leq 7$	<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"/> </xsd:simpleType></pre>
FIXED BINARY (<i>n</i>) 여기서 $8 \leq n \leq 15$	<pre><xsd:simpleType> <xsd:restriction base="xsd:short"/> </xsd:simpleType></pre>

PL/I 데이터 설명	스키마
FIXED BINARY (n) 여기서 $16 \leq n \leq 31$	<pre><xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType></pre>
FIXED BINARY (n) 여기서 $32 \leq n \leq 63$ 제한사항: 엔터프라이즈 PL/I만 해당	<pre><xsd:simpleType> <xsd:restriction base="xsd:long"/> </xsd:simpleType></pre>
UNSIGNED FIXED BINARY(n) 여기서 $n \leq 8$ 제한사항: 엔터프라이즈 PL/I만 해당	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"/> </xsd:simpleType></pre>
UNSIGNED FIXED BINARY(n) 여기서 $9 \leq n \leq 16$ 제한사항: 엔터프라이즈 PL/I만 해당	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"/> </xsd:simpleType></pre>
UNSIGNED FIXED BINARY(n) 여기서 $17 \leq n \leq 32$ 제한사항: 엔터프라이즈 PL/I만 해당	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType></pre>
UNSIGNED FIXED BINARY(n) 여기서 $33 \leq n \leq 64$ 제한사항: 엔터프라이즈 PL/I만 해당	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"/> </xsd:simpleType></pre>
FIXED DECIMAL(n, m)	<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="n"/> <xsd:fractionDigits value="m"/> </xsd:restriction> </xsd:simpleType></pre>
FIXED DECIMAL(15) DATETIME=PACKED15인 경우 맵핑 레벨 3.0 이상에서 지원됨	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre> 시간소인 형식은 CICS ABSTIME입니다.
BIT(n) 여기서 n 은 8의 배수입니다. 기타 값은 지원되지 않습니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="m"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 $m = n/8$
CHARACTER(n) VARYING 및 VARYINGZ도 맵핑 레벨 1.2 이상에서 지원됩니다. 제한사항: VARYINGZ는 엔터프라이즈 PL/I에서만 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre>

PL/I 데이터 설명	스키마
GRAPHIC(<i>n</i>) VARYING 및 VARYINGZ도 맵핑 레벨 1.2 이상에서 지원됩니다. 제한사항: VARYINGZ는 엔터프라이즈 PL/I에서만 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="m"/> </xsd:restriction> </xsd:simpleType></pre> 1.0 및 1.1의 맵핑 레벨이며, 여기서 $m = 2*n$ <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> 1.2 이상의 맵핑 레벨
WIDECHAR(<i>n</i>) 제한사항: 엔터프라이즈 PL/I만 해당	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="m"/> </xsd:restriction> </xsd:simpleType></pre> 1.0 및 1.1의 맵핑 레벨이며, 여기서 $m = 2*n$ <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="n"/> </xsd:restriction> </xsd:simpleType></pre> 1.2 이상의 맵핑 레벨
ORDINAL 제한사항: 엔터프라이즈 PL/I만 해당	<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"/> </xsd:simpleType></pre>
BINARY FLOAT(<i>n</i>) 여기서 $n \leq 21$ 맵핑 레벨 1.2 이상에서 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
BINARY FLOAT(<i>n</i>) 여기서 $21 < n \leq 53$ 53보다 큰 값은 지원되지 않습니다. 맵핑 레벨 1.2 이상에서 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>
DECIMAL FLOAT(<i>n</i>) 여기서 $n \leq 6$ 맵핑 레벨 1.2 이상에서 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
DECIMAL FLOAT(<i>n</i>) 여기서 $6 < n \leq 16$ 16보다 큰 값은 지원되지 않습니다. 맵핑 레벨 1.2 이상에서 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

XML 스키마 대 PL/I 맵핑:

DFHSC2LS 및 DFHWS2LS 유틸리티 프로그램은 XML 스키마 정의와 PL/I 데이터 구조 간 맵핑을 지원합니다. 엔터프라이즈 PL/I 컴파일러와 이전 PL/I 컴파일러는 다르므로 두 가지 언어 옵션(PLI-ENTERPRISE 및 PLI-OTHER)이 지원됩니다.

CICS 지원은 다음 규칙을 사용하여 스키마 요소 이름에서 PL/I 변수의 고유하고 올바른 이름을 생성합니다.

1. A-Z, a-z, 0-9, @, # 또는 \$ 이외의 문자는 'X'로 대체됩니다.

예를 들어, `monthly-total`은 `monthlyXtotal`이 됩니다.

2. 스키마가 변수의 카디널리티가 가변적이고(즉, `minOccurs` 및 `maxOccurs` 속성이 `xsd:element`에서 다른 값으로 지정됨) 스키마 요소 이름이 24자보다 긴 것으로 지정하면 해당 길이에서 잘립니다.

스키마가 변수의 카디널리티가 고정적이고 스키마 요소 이름이 29자보다 긴 것으로 지정하면 해당 길이에서 잘립니다.

3. 동일한 범위의 중복된 이름은 이름의 두 번째 및 후속 인스턴스에 하나 이상의 숫자를 추가하여 고유한 이름이 됩니다.

예를 들어, `year`의 세 가지 인스턴스는 `year`, `year1`, `year2`가 됩니다.

4. 문자열 `_cont` 또는 `_num`에는 다섯 개 문자가 예약되어 스키마가 변수의 카디널리티가 가변적인 것으로 지정하는 경우(즉, `minOccurs` 및 `maxOccurs` 속성이 다른 값으로 지정되는 경우) 사용됩니다.

자세한 정보는 132 페이지의 『요소의 가변 배열』을 참조하십시오.

5. 속성의 경우 이전 규칙이 요소 이름에 적용됩니다. 접두부 `attr-`이 요소 이름에 추가되고 그 다음에 `-value` 또는 `-exist`가 나타납니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다. 자세한 정보는 137 페이지의 『XML 속성에 대한 지원』을 참조하십시오.

`nillable` 속성은 특수 규칙을 갖습니다. 접두부 `attr-`이 추가되지만 요소 이름 처음에 `nil-`도 추가됩니다. 요소 이름 다음에는 `-value`가 나타납니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다.

결과 이름의 총 길이는 31자 이하입니다.

DFHSC2LS 및 DFHWS2LS는 다음 표에 따라 스키마 유형을 PL/I 데이터 유형에 맵핑합니다. 다음 사항에 주의하십시오.

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 NULL로 설정되면 가변 길이 문자 데이터가 널 종료 문자열에 맵핑되고 널 종료자에 추가 문자가 할당됩니다.

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 지정되지 않는 경우에는 기본적으로 가변 길이 문자 데이터가 **VARYINGZ** 데이터 유형(엔터프라이즈 PL/I의 경우) 및 **VARYING** 데이터 유형(기타 PL/I의 경우)에 맵핑됩니다.
- 가변 길이 2진 데이터는 **VARYING** 데이터 유형(32 768바이트 미만인 경우) 및 컨테이너(32 768바이트보다 큰 경우)에 맵핑됩니다.

스키마	PL/I 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 2.0 이하: 지원되지 않음 맵핑 레벨 2.1 이상: 지원됨
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpleType"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이상:CHAR(255)
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> <xsd:maxLength value="z"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY	모든 맵핑 레벨:CHARACTER(z)

스키마	PL/I 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. duration date time gDay gMonth gYear gMonthDay gYearMonth	모든 맵핑 레벨:CHAR(32)
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.2 이하: CHAR(32) 맵핑 레벨 2.0 이상: CHAR(40) 맵핑 레벨 3.0 이상: FIXED DECIMAL(15) 시간소인 형식은 CICS ABSTIME입니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="y"/> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: BIT(<i>z</i>) 여기서 $z = 8 \times y$, $z < 4095$바이트 CHAR(<i>z</i>) 여기서 $z = 8 \times y$, $z > 4095$바이트 맵핑 레벨 1.2 이상: CHAR(<i>y</i>)
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I SIGNED FIXED BINARY (7) 기타 PL/I FIXED BINARY (7)
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I UNSIGNED FIXED BINARY (8) 기타 PL/I FIXED BINARY (8)

스키마	PL/I 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I SIGNED FIXED BINARY (15) 기타 PL/I FIXED BINARY (15)
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I UNSIGNED FIXED BINARY (16) 기타 PL/I FIXED BINARY (16)
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I FIXED DECIMAL(31,0) 기타 PL/I FIXED DECIMAL(15,0)
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I SIGNED FIXED BINARY (31) 기타 PL/I FIXED BINARY (31)
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: 엔터프라이즈 PL/I UNSIGNED FIXED BINARY(32) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I CHAR(y) 여기서 y는 16MB 미만의 고정 길이입니다. 모든 맵핑 레벨: 기타 PL/I BIT(64)

스키마	PL/I 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: 엔터프라이즈 PL/I SIGNED FIXED BINARY(63) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I CHAR(y) 여기서 y는 16MB 미만의 고정 길이입니다. 모든 맵핑 레벨: 기타 PL/I BIT(64)
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: 엔터프라이즈 PL/I UNSIGNED FIXED BINARY(64) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I CHAR(y) 여기서 y는 16MB 미만의 고정 길이입니다. 모든 맵핑 레벨: 기타 PL/I BIT(64)
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: 엔터프라이즈 PL/I SIGNED FIXED BINARY (7) 기타 PL/I FIXED BINARY (7) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I BIT(7) BIT(1) 기타 PL/I BIT(7) BIT(1) 여기서 BIT(7)은 맞추기 용도로 제공되며 BIT(1)은 부울 맵핑 값을 갖습니다.

스키마	PL/I 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="n"/> <xsd:fractionDigits value="m"/> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: FIXED DECIMAL(n, m)
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	모든 맵핑 레벨: CHAR(255)
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	모든 맵핑 레벨: CHAR(255)
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value="y"/> </xsd:restriction> </xsd:simpleType> <xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> </xsd:restriction> </xsd:simpleType></pre> 길이가 정의되지 않는 경우	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1: CHAR(z) 여기서 $z = 4 \times (\text{ceil}(y/3))$. $\text{ceil}(x)$은 x보다 크거나 같은 가장 작은 정수입니다. 맵핑 레벨 1.2 이상: CHAR(y) 여기서 길이는 고정적입니다. CHAR(16) 길이가 정의되지 않는 경우. 이 필드는 2진 데이터를 저장하는 컨테이너의 16바이트 이름을 갖습니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.0 및 1.1: CHAR(32) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I DECIMAL FLOAT(6) HEXADEC 기타 PL/I DECIMAL FLOAT(6)
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.0 및 1.1: CHAR(32) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I DECIMAL FLOAT(16) HEXADEC 기타 PL/I DECIMAL FLOAT(16)

요소의 가변 배열:

XML은 요소 수가 가변적인 배열을 포함할 수 있습니다. 일반적으로 요소 수가 일정하지 않은 WSDL 문서와 XML 스키마는 단일 고급 언어 데이터 구조에 효율적으로 맵핑되지 않습니다. CICS는 컨테이너 기반 맵핑 또는 인라인 맵핑을 사용하여 XML에서 수가 일정하지 않은 요소를 처리합니다.

요소 수가 가변적인 배열은 XML 스키마에서 요소 선언에 `minOccurs` 및 `maxOccurs` 속성을 사용하여 표시됩니다.

- `minOccurs` 속성은 요소가 발생할 수 있는 최소 횟수를 지정합니다. 0 이상의 양의 정수 값을 지정할 수 있습니다.
- `maxOccurs` 속성은 요소가 발생할 수 있는 최대 횟수를 지정합니다. `minOccurs` 속성 값보다 크거나 같은 양의 정수 값을 지정할 수 있습니다. `unbounded` 값을 가질 수도 있으며 이는 요소가 발생할 수 있는 횟수에 상한이 적용되지 않음을 나타냅니다.
- 두 속성의 기본값은 모두 1입니다.

이 예제는 선택적인 8바이트 문자열을 보여줍니다. 즉, 애플리케이션 XML 또는 SOAP 메시지에서 나타나지 않거나 한 번 나타날 수 있습니다.

```
<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>
```

다음 예제는 최소 한 번 나타나야 하는 8바이트 문자열을 보여줍니다.

```
<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>
```

일반적으로 요소 수가 일정하지 않은 WSDL 문서는 단일 고급 언어 데이터 구조에 효율적으로 맵핑되지 않습니다. 따라서 이러한 사례를 처리하기 위해 CICS는 일련의 컨테이너에서 애플리케이션 프로그램에 전달되는 일련의 연결된 데이터 구조를 사용합니다. 이러한 구조는 애플리케이션의 입력 및 출력으로 사용됩니다.

- CICS가 XML을 애플리케이션 데이터로 변환하는 경우, 애플리케이션 데이터로 해당 구조를 채우고 애플리케이션이 해당 구조를 읽습니다.

- CICS가 애플리케이션 데이터를 XML로 변환하는 경우, 애플리케이션으로 채운 구조에서 애플리케이션 데이터를 읽습니다.

이러한 데이터 구조의 형식은 일련의 예로 가장 정확하게 설명할 수 있습니다. XML은 SOAP 메시지 또는 애플리케이션에서 가져올 수 있습니다. 이 예제는 단순 8바이트 필드의 배열을 사용합니다. 그러나 모델은 복잡한 데이터 유형 배열과 다른 배열을 포함하는 데이터 유형 배열을 지원합니다.

고정 요소 수

첫 번째 예제에서는 정확히 세 번 나타나는 요소를 보여줍니다.

```
<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>
```

이 예제에서는 요소 발생 횟수가 미리 알려져 있으므로 단순 COBOL 선언에서 고정 길이 배열(다른 언어의 경우 이에 상응하는 배열)로 표시될 수 있습니다.

```
05 component PIC X(8) OCCURS 3 TIMES
```

맵핑 레벨 2 이하의 가변 요소 수

이 예제는 한 번에서 다섯 번까지 발생할 수 있는 필수 요소를 나타냅니다.

```
<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>
```

주 데이터 구조에는 두 필드의 선언이 포함됩니다. CICS가 XML을 2진 데이터로 변환하는 경우 첫 번째 필드 component-num에는 요소가 XML에서 나타나는 횟수가 포함되고 두 번째 필드 component-cont에는 컨테이너 이름이 포함됩니다.

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

두 번째 데이터 구조에는 요소 선언 자체가 포함됩니다.

```
01 DFHWS-component
  02 component PIC X(8)
```

component-num(1 - 5 범위의 값을 가짐)의 값을 검사하여 요소 발생 횟수를 확인해야 합니다. 요소 콘텐츠는 component-cont 컨테이너에 있습니다. 이 컨테이너는 각 요소가 DFHWS-component 데이터 구조에 의해 매핑되는 요소 배열을 포함합니다.

minOccurs="0" 및 maxOccurs="1"인 경우에는 요소가 선택사항입니다. 애플리케이션 프로그램에서 데이터 구조를 처리하려면 component-num 값을 검사해야 합니다.

- 이 값이 0이면 메시지에 컴포넌트 요소가 없고 component-cont의 콘텐츠가 정의되지 않습니다.
- 이 값이 1이면 컴포넌트 요소가 component-cont 컨테이너에 있습니다.

이 컨테이너의 콘텐츠는 DFHWS-component 데이터 구조에 따라 매핑됩니다.

참고: SOAP 메시지가 단일 반복 요소로 구성되는 경우에는 DFHWS2LS가 두 개 언어 구조를 생성합니다. 주 언어 구조에는 배열의 요소 수와 요소 배열을 포함하는 컨테이너의 이름이 포함됩니다. 두 번째 언어 구조는 반복 요소의 단일 인스턴스를 매핑합니다.

매핑 레벨 2.1 이상의 가변 요소 수

매핑 레벨 2.1 이상의 경우 CICS 지원에서 **INLINE-MAXOCCURS-LIMIT** 매개변수를 사용할 수 있습니다. **INLINE-MAXOCCURS-LIMIT** 매개변수는 가변적인 수의 요소가 처리되는 방식을 지정합니다. 가변적인 수의 요소에 대한 매핑 옵션은 컨테이너 기반 매핑(133 페이지의 『매핑 레벨 2 이하의 가변 요소 수』의 설명 참조) 또는 인라인 매핑입니다. 이 매개변수의 value는 0 - 32767 범위의 양의 정수일 수 있습니다.

- **INLINE-MAXOCCURS-LIMIT**의 기본값은 1이며 이 경우 선택적 요소가 인라인으로 매핑됩니다.
- **INLINE-MAXOCCURS-LIMIT** 매개변수의 값이 0이면 인라인 매핑을 방지합니다.
- maxOccurs가 **INLINE-MAXOCCURS-LIMIT** 값 이하인 경우 인라인 매핑이 사용됩니다.
- maxOccurs가 **INLINE-MAXOCCURS-LIMIT** 값보다 크면 컨테이너 기반 매핑이 사용됩니다.

가변적인 수의 요소를 인라인으로 매핑하면 위 고정 발생 예제의 경우와 같이 배열과 카운터가 생성됩니다. component-num 필드는 존재하는 요소 인스턴스 수를 나타내며 해당 인스턴스는 배열로 가리킵니다. 예를 들어, 133 페이지의 『매핑 레벨 2 이하의 가변 요소 수』처럼 **INLINE-MAXOCCURS-LIMIT**이 5 이하인 경우에는 다음과 같은 데이터 구조가 생성됩니다.

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

첫 번째 필드, component-num은 이전 섹션에 나와 있는 컨테이너 기반 매핑 예제의 출력과 같습니다. 두 번째 필드에는 생성될 수 있는 최대 요소 수를 포함할 수 있는 길이가 5인 배열이 포함됩니다.

인라인 매핑은 요소 발생 횟수와 데이터가 배치되는 컨테이너의 이름을 저장하는 컨테이너 기반 매핑과 다릅니다. 이 매핑은 현재 컨테이너에 모든 데이터를 저장하기 때문

입니다. 데이터를 현재 컨테이너에 저장하면 일반적으로 성능이 향상되므로 인라인 맵핑을 선호하게 됩니다.

중첩된 가변 배열

복잡한 WSDL 문서와 XML 스키마에는 가변 반복 요소가 포함될 수 있으며 이 요소에도 가변 반복 요소가 포함될 수 있습니다. 이러한 경우, 설명되는 구조는 예제에서 설명하는 두 가지 레벨 이상으로 확장됩니다.

이 예제는 필수 요소 <component1>에 중첩되는 선택적 요소 <component2>를 보여줍니다. 이 경우 필수 요소는 한 번에서 다섯 번까지 발생할 수 있습니다.

```
<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>
```

최상위 레벨 데이터 구조는 이전 예제와 정확하게 일치합니다.

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

그러나 두 번째 데이터 구조에는 다음 요소가 포함됩니다.

```
01 DFHWS-component1
  02 component2-num PIC S9(9) COMP-5
  02 component2-cont PIC X(16)
```

세 번째 구조에는 다음 요소가 포함됩니다.

```
01 DFHWS-component2
  02 component2 PIC X(8)
```

가장 바깥쪽 요소 <component1>의 발생 횟수는 component1-num에 나타납니다.

component1-cont에서 이름이 지정된 컨테이너는 두 번째 데이터 구조 DFHWS-component1의 인스턴스 수를 갖는 배열을 포함합니다.

component2-cont의 각 인스턴스는 다른 컨테이너의 이름을 지정하며 각각 세 번째 레벨 구조 DFHWS-component2가 맵핑하는 데이터 구조를 포함합니다.

이 구조를 나타내려면 예제와 일치하는 XML 단편을 고려하십시오.

```
<component1><component2>string1</component2></component1>
<component1><component2>string2</component2></component1>
<component1></component1>
```

<component1> occurs three times. 처음 두 인스턴스 각각 <component2> 인스턴스를 포함합니다. 세 번째 인스턴스는 해당되지 않습니다.

최상위 레벨 데이터 구조의 경우 component1-num은 값 3을 갖습니다.

component1-cont에서 이름이 지정된 컨테이너는 다음 세 가지 DFHWS-component1 인스턴스를 갖습니다.

1. 첫 번째의 경우 component2-num의 값은 1이고 component2-cont에서 이름이 지정된 컨테이너는 *string1*을 포함합니다.
2. 두 번째의 경우 component2-num의 값은 1이고 component2-cont에서 이름이 지정된 컨테이너는 *string2*를 포함합니다.
3. 세 번째의 경우 component2-num의 값은 0이고 component2-cont의 콘텐츠가 정의되지 않습니다.

이 인스턴스에서 전체 데이터 구조는 모두 네 개의 컨테이너로 나타냅니다.

- 컨테이너 DFHWS-DATA의 루트 데이터 구조
- component1-cont에서 이름이 지정된 컨테이너
- component2-cont의 처음 두 인스턴스에서 이름이 지정된 두 개 컨테이너

선택적 구조와 **xsd:choice**

DFHWS2LS 및 DFHSC2LS는 맵핑 레벨이 2.1 이상이 경우 <xsd:sequence>, <xsd:choice>, <xsd:all> 요소에서 maxOccurs 및 minOccurs 사용을 지원합니다. 여기서 minOccurs 및 maxOccurs 속성은 minOccurs="0" 및 maxOccurs="1"로 설정됩니다.

지원은 이 요소를 각 하위 요소가 선택사항인 것처럼 처리하는 맵핑을 생성합니다. 이러한 요소로 애플리케이션을 구현하는 경우 애플리케이션에 의해 올바르지 않은 옵션 조합이 생성되지 않도록 해야 합니다. 각각의 요소는 생성된 언어 구조에서 고유한 count 필드를 가지며 해당 필드는 모두 "0"으로 설정되거나 모두 "1"로 설정되어야 합니다. <xsd:choice> 요소를 제외하고는 다른 값 조합은 모두 올바르지 않습니다.

<xsd:choice> 요소는 요소의 옵션 중 하나만 사용할 수 있음을 나타냅니다. 이 요소는 모든 맵핑 레벨에서 지원됩니다. 지원은 <xsd:choice>에서 각 옵션을 minOccurs="0" 및 maxOccurs="1"인 <xsd:sequence> 요소에 있는 것처럼 처리합니다. 애플리케이션에 의해 올바르지 않은 조합의 옵션이 생성되지 않도록 하기 위해 <xsd:choice> 요소를 사용하여 애플리케이션을 구현하는 경우 유의해야 합니다. 각 요소는 생성된 언어 구조에서 고유한 count 필드를 가집니다. 그 중 필드는 반드시 '1'로 설정되어야 하며 다른 필드는 모두 '0'으로 설정되어야 합니다. 다른 모든 값 조합은 올바르지 않은 조

합입니다. 단, <xsd:choice> 요소 자체가 선택사항인 경우는 예외이며 이 경우에는 모든 필드가 '0'으로 설정되어야 합니다.

XML 속성에 대한 지원:

XML 스키마는 XML에서 필요하거나 허용되는 속성을 지정할 수 있습니다. CICS 지원 유틸리티 DFHWS2LS 및 DFHSC2LS는 기본적으로 XML 속성을 무시합니다. XML 스키마에서 정의되는 XML 속성을 처리하려면 **MAPPING-LEVEL** 매개변수의 값이 1.1 이상이어야 합니다.

선택적 속성

속성은 선택사항 또는 필수사항이며 SOAP 메시지의 요소 또는 애플리케이션의 경우 XML과 연관될 수 있습니다. 스키마에서 정의되는 모든 선택적 속성의 경우 다음과 같이 해당 언어 구조에서 두 개 필드가 생성됩니다.

1. 존재 플래그. 이 필드는 부울 데이터 유형으로 처리되며 일반적으로 길이는 1바이트입니다.
2. 값. 이 필드는 같은 유형의 XML 요소와 동일한 방식으로 맵핑됩니다. 예를 들어, NMTOKEN 유형의 속성은 NMTOKEN 유형의 XML 요소와 동일한 방식으로 맵핑됩니다.

속성 존재 및 값 필드는 생성된 언어 구조에서 연관된 요소에 대한 필드 앞에 나타납니다. 인스턴스 문서에 나타나는 예상치 못한 속성은 무시됩니다.

예를 들어, 다음 스키마 속성 정의를 고려하십시오.

```
<xsd:attribute name="age" type="xsd:short" use="optional" />
```

이 선택적 속성은 다음 COBOL 구조에 맵핑됩니다.

```
05 attr-age-exist PIC X DISPLAY  
05 attr-age-value PIC S9999 COMP-5 SYNC
```

선택적 속성의 런타임 처리

다음 런타임 처리는 선택적 속성에 대해 수행됩니다.

- 속성이 있으면 존재 플래그가 설정되고 값이 맵핑됩니다.
- 속성이 없으면 존재 플래그가 설정되지 않습니다.
- 속성이 기본값을 갖고 존재하는 경우에는 값이 맵핑됩니다.
- 속성이 기본값을 갖지만 존재하지 않는 경우에는 기본값이 맵핑됩니다.

기본값을 갖는 선택적 속성은 필수 속성으로 처리됩니다.

CICS가 데이터를 XML로 변환하는 경우 다음 런타임 처리가 수행됩니다.

- 존재 플래그가 설정된 경우 속성이 변환되고 XML에 포함됩니다.

- 존재 플래그가 설정되지 않은 경우에는 속성이 XML에 포함되지 않습니다.

필수 속성 및 런타임 처리

필요한 모든 속성마다 해당 언어 구조에서 값 필드만 생성됩니다.

속성이 XML에 존재하는 경우에는 값이 매핑됩니다. 속성이 존재하지 않으면 다음과 같이 처리됩니다.

- 애플리케이션이 웹 서비스 제공자인 경우에는 CICS가 클라이언트 SOAP 메시지에서 오류를 나타내는 SOAP 결함 메시지를 생성합니다.
- 애플리케이션이 웹 서비스 요청자인 경우에는 CICS가 메시지를 발행하고 RESP2 코드 13과 함께 변환 오류 응답을 애플리케이션에 리턴합니다.
- 애플리케이션이 **TRANSFORM XMLTODATA** 명령을 사용하는 경우에는 CICS가 메시지를 발행하고 RESP2 코드 3과 함께 올바른지 않은 요청 응답을 애플리케이션에 리턴합니다.

CICS가 COMMAREA 또는 컨테이너의 콘텐츠를 기반으로 하는 SOAP 메시지를 생성하는 경우에는 속성이 변환되어 메시지에 포함됩니다. 애플리케이션이 **TRANSFORM DATATOXML** 명령을 사용하는 경우에는 CICS가 또한 속성을 변환하고 XML에 포함시킵니다.

nillable 속성

nillable 속성은 XML 스키마에서 `xsd:element`에 나타날 수 있는 특수 속성입니다. 이 속성은 `xsi:nil` 속성이 XML의 요소에 올바른 것으로 지정합니다. 요소에 `xsi:nil` 속성이 지정되는 경우 요소는 있지만 값을 갖지 않음을 나타내므로 연관된 콘텐츠가 없습니다.

XML 스키마가 nillable 속성을 true로 정의한 경우에는 부울 값을 갖는 필수 속성으로 매핑됩니다.

CICS가 SOAP 메시지를 수신하거나 `xsi:nil` 속성을 갖는 애플리케이션에 대해 XML을 변환해야 하는 경우 속성 값은 true 또는 false입니다. 값이 true인 경우에는 애플리케이션이 요소 또는 `xsi:nil` 속성 범위에 중첩된 요소의 값을 무시해야 합니다.

CICS가 `xsi:nil` 속성의 값이 true인 COMMAREA 또는 컨테이너의 콘텐츠를 기반으로 SOAP 메시지 또는 XML을 생성하는 경우에는 다음과 같이 처리됩니다.

- `xsi:nil` 속성은 XML 또는 SOAP 메시지에 생성됩니다.
- 연관된 요소의 값은 무시됩니다.
- 요소 내에 중첩된 요소는 무시됩니다.

SOAP 메시지 예

WSDL 문서의 일부일 수 있는 다음 XML 스키마 예를 고려하십시오.

```
<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>
```

다음은 이 스키마를 준수하는 부분 SOAP 메시지의 예입니다.

```
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <num xsi:nil="true"/>
  <num>15</num>
  <num xsi:nil="true"/>
</root>
```

COBOL에서는 이 SOAP 메시지가 다음 요소에 맵핑됩니다.

```
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> 및 xsd:anyType에 대한 지원:

DFHWS2LS 및 DFHSC2LS는 XML 스키마에서 <xsd:any> 및 xsd:anyType 사용을 지원합니다. <xsd:any> XML 스키마 요소를 사용하여 정의되지 않은 콘텐츠가 있는 XML 문서의 섹션을 설명할 수 있습니다. xsd:anyType은 단순하고 복잡한 모든 데이터 유형이 파생되는 기본 데이터 유형입니다. 데이터 콘텐츠에는 제한사항 또는 제한조건이 없습니다.

CICS 지원과 함께 <xsd:any> 및 xsd:anyType을 사용하려면 먼저 다음 매개변수를 설정하십시오.

- **MAPPING-LEVEL** 매개변수를 2.1 이상으로 설정하십시오.
- 웹 서비스 제공자 애플리케이션의 경우 **PGMINT** 매개변수를 **CHANNEL**로 설정하십시오.

<xsd:any> 예제

이 예제에서는 <xsd:any> 요소를 사용하여 "Customer" 태그에서 "Surname" 태그 다음에 있는 구조화되지 않은 일부 선택적 XML 콘텐츠를 설명합니다.

```
<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>

```

이 XML 스키마를 준수하는 SOAP 메시지 예는 다음과 같습니다.

```

<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>

```

이 SOAP 메시지가 CICS로 전송되면 CICS가 다음 XML 데이터로 Customer-xml-cont 컨테이너를 채웁니다.

```

<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>

```

CICS는 또한 범위에 포함되는 다음 XML 네임스페이스 선언으로 Customer-xmlns-cont 컨테이너를 채웁니다. 이러한 선언은 공백으로 구분됩니다.

```
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns="http://www.example.org/anyExample"
```

xsd:anyType 예제

xsd:anyType은 단순 및 복잡 데이터 유형이 모두 파생되는 기본 데이터 유형입니다. 데이터 콘텐츠를 제한하지 않습니다. 데이터 유형을 지정하지 않는 경우 기본값은 xsd:anyType입니다. 예를 들어, 다음 두 XML 단편은 동일합니다.

```

<xsd:element name="Name" type="xsd:anyType"/>
<xsd:element name="Name"/>

```

생성된 언어 구조

<xsd:any> 또는 xsd:anyType에 대해 생성된 언어 구조는 COBOL에서 다음 양식을 나타내며 다른 언어의 경우에는 그에 상응하는 양식을 나타냅니다.

elementName-xml-cont PIC X(16)

원시 XML을 포함하는 컨테이너의 이름. CICS가 수신 SOAP 메시지를 처리하는 경우, <xsd:any> 또는 xsd:anyType이 정의하는 SOAP 메시지의 서브세트를 이 컨테이너에 저장합니다. 애플리케이션은 기본적으로 XML 데이터만 처리할 수 있습니다. 애플리케이션은 XML을 생성하고 이 컨테이너를 채워야 하며 컨테이너 이름을 제공해야 합니다.

이 컨테이너는 텍스트 모드에서 채워야 합니다. CICS가 이 컨테이너를 채우는 경우, 사용하도록 정의된 웹 서비스와 동일한 EBCDIC 변형을 사용합니다. 대상 EBCDIC 코드 페이지에 없는 문자는 애플리케이션이 컨테이너를 UTF-8로 읽는 경우라도 대체 문자로 바꿉니다.

elementName-xmlns-cont PIC X(16)

범위에 있는 네임스페이스 접두부 선언을 포함하는 컨테이너의 이름. 이 컨테이너의 콘텐츠는 SOAP 엔벨로프 태그의 서브세트만이 아니라 범위 내에 있고 관련이 있는 모든 네임스페이스 선언을 포함한다는 점을 제외하고는 DFHWS-XMLNS 컨테이너의 콘텐츠와 유사합니다.

이 컨테이너는 텍스트 모드에서 채워야 합니다. CICS가 이 컨테이너를 채우는 경우, 사용하도록 정의된 웹 서비스와 동일한 EBCDIC 변형을 사용합니다. 대상 EBCDIC 코드 페이지에 없는 문자는 애플리케이션이 컨테이너를 UTF-8로 읽는 경우라도 대체 문자로 바꿉니다.

이 컨테이너는 CICS로 보낸 SOAP 메시지를 처리하는 경우에만 사용됩니다. 애플리케이션이 출력 SOAP 메시지가 생성될 때 컨테이너에 네임스페이스 선언을 제공하려고 시도하는 경우, 컨테이너와 해당 콘텐츠를 CICS가 무시합니다. CICS에서는 애플리케이션이 제공하는 XML이 네임스페이스 선언에 있어 전적으로 자체 포함되어야 합니다.

<xsd:any> 요소를 포함하는 XML 요소의 이름은 <xsd:any> 요소에 대해 생성되는 변수 이름에 포함됩니다. <xsd:any> 예제에서, <xsd:any> 요소는 <xsd:element name="Customer"> 요소 내부에 중첩되고 <xsd:any> 요소에 대해 생성되는 변수 이름은 Customer-xml-cont PIC X(16) 및 Customer-xmlns-cont PIC X(16)입니다.

xsd:anyType 유형의 경우에는 직접 XML 요소 이름이 사용됩니다. 위 xsd:anyType 예제에서 변수 이름은 Name-xml-cont PIC X(16) 및 Name-xmlns-cont PIC X(16)입니다.

<xsd:choice>에 대한 지원:

<xsd:choice> 요소는 요소의 옵션 중 하나만 사용할 수 있음을 나타냅니다. CICS 지원은 다양한 맵핑 레벨에서 <xsd:choice> 요소에 대한 다양한 수준의 지원을 제공합니다.

맵핑 레벨 2.2 이상의 <xsd:choice>에 대한 지원

맵핑 레벨이 2.2 이상인 경우 DFHWS2LS 및 DFHSC2LS는 <xsd:choice> 요소에 대한 향상된 지원을 제공합니다. 지원은 <xsd:choice> 요소와 연관된 값을 저장하는 새 컨테이너를 생성합니다. 지원은 새 컨테이너의 이름과 추가 필드를 포함하는 언어 구조를 생성합니다.

fieldname-enum

<xsd:choice> 요소가 사용하는 옵션을 표시하는 구분 필드

fieldname-cont

사용될 옵션을 저장하는 컨테이너의 이름. 옵션 값을 맵핑하기 위한 추가 언어 구조가 생성됩니다.

다음 XML 스키마 단편에는 <xsd:choice> 요소가 포함됩니다.

```
<xsd:element name="choiceExample">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="option1" type="xsd:string" />
      <xsd:element name="option2" type="xsd:int" />
      <xsd:element name="option3" type="xsd:short" maxOccurs="2" minOccurs="2" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

이 XML 스키마 단편이 맵핑 레벨 2.2 이상에서 처리되는 경우 지원은 다음 COBOL 언어 구조를 생성합니다.

```
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.
```

맵핑 레벨 2.2 이상의 <xsd:choice>에 대한 제한사항

DFHSC2LS 및 DFHWS2LS는 중첩된 <xsd:choice> 요소를 지원하지 않습니다. 예를 들어, 다음 XML은 지원되지 않습니다.

```
<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 및 DFHWS2LS는 반복 <xsd:choice> 요소를 지원하지 않습니다. 예를 들어, 다음 XML은 지원되지 않습니다.

```

<xsd:choice maxOccurs="2">
    <xsd:element name ="name1" type="string"/>
</xsd:choice>

```

DFHSC2LS 및 DFHWS2LS는 <xsd:choice> 요소에서 최대 255개 옵션을 지원합니다.

맵핑 레벨 2.1 이하의 <xsd:choice>에 대한 지원

맵핑 레벨 2.1 이하의 경우 DFHWS2LS는 <xsd:choice> 요소에 대한 제한된 지원을 제공합니다. DFHWS2LS는 <xsd:choice> 요소의 각 옵션을 최대한 한 번 발생할 수 있는 <xsd:sequence> 요소인 것처럼 처리합니다.

<xsd:choice> 요소의 옵션 중 하나만 사용할 수 있으므로 <xsd:choice> 요소를 사용하여 애플리케이션을 구현하는 경우 주의해야 합니다. 올바른 옵션 조합만 생성합니다. 각 요소는 생성된 언어 구조에서 고유한 count 필드를 가집니다. 그 중 필드는 반드시 1로 설정되어야 하며 다른 필드는 모두 0으로 설정되어야 합니다. 다른 모든 값 조합은 잘못된 조합입니다. 단, <xsd:choice>가 선택사항인 경우는 예외이며 이 경우에는 모든 필드가 0으로 설정되어야 합니다.

관련 참조:

139 페이지의 『<xsd:any> 및 xsd:anyType에 대한 지원』

DFHWS2LS 및 DFHSC2LS는 XML 스키마에서 <xsd:any> 및 xsd:anyType 사용을 지원합니다. <xsd:any> XML 스키마 요소를 사용하여 정의되지 않은 콘텐츠가 있는 XML 문서의 섹션을 설명할 수 있습니다. xsd:anyType은 단순하고 복잡한 모든 데이터 유형이 파생되는 기본 데이터 유형입니다. 데이터 콘텐츠에는 제한사항 또는 제한조건이 없습니다.

145 페이지의 『추상 요소 및 추상 데이터 유형에 대한 지원』

CICS 지원은 맵핑 레벨 2.2 이상에서 추상 요소 및 추상 데이터 유형에 대한 지원을 제공합니다. CICS 지원은 추상 요소와 추상 데이터 유형을 유사한 방식으로 대체 그룹에 맵핑합니다.

144 페이지의 『대체 그룹에 대한 지원』

대체 그룹을 사용하여 상호 교환할 수 있는 XML 요소의 그룹을 정의할 수 있습니다. CICS 지원은 맵핑 레벨 2.2 이상에서 대체 그룹에 대한 지원을 제공합니다.

대체 그룹에 대한 지원:

대체 그룹을 사용하여 상호 교환할 수 있는 XML 요소의 그룹을 정의할 수 있습니다. CICS 지원은 맵핑 레벨 2.2 이상에서 대체 그룹에 대한 지원을 제공합니다.

맵핑 레벨이 2.2 이상인 경우, DFHSC2LS 및 DFHWS2LS는 <xsd:choice> 요소에 사용한 맵핑과 유사한 맵핑을 사용하여 대체 그룹을 지원합니다. 지원은 언어 구조에서 열거 필드와 새 컨테이너 이름을 생성합니다.

다음 XML 스키마 단편에는 각각 replacementOption1 또는 replacementOption2: 로 바꿀 수 있는 두 subGroupParent 요소의 배열이 포함됩니다.

```
<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" />
```

이 XML 단편을 지원과 함께 처리하면 다음 COBOL 언어 구조가 생성됩니다.

```
03 subGroupExample.
  06 subGroupParent OCCURS2.
    09 subGroupExample-enum PIC X DISPLAY.
      88 empty VALUE X '00'.
      88 replacementOption1 VALUE X '01'.
      88 replacementOption2 VALUE X '02'.
      88 subGroupParent VALUE X '03'.
    09 subGroupExample-cont PIC X (16).

01 Example-replacementOption1.
  03 replacementOption1 PIC S9(9) COMP-5 SYNC.

01 Example-replacementOption2.
  03 replacementOption2 PIC S9999 COMP-5 SYNC.

01 Example-subGroupParent.
  03 subGroupParent-length PIC S9999 COMP-5 SYNC.
  03 subGroupParent PIC X(255).
```

대체 그룹에 대한 자세한 정보는 *W3C XML 스키마 파트 1: 구조 Second Edition 스펙*(http://www.w3.org/TR/xmlschema-1/#Elements_Equivalence_Class)을 참조하십시오.

관련 참조:

139 페이지의 『<xsd:any> 및 xsd:anyType에 대한 지원』

DFHWS2LS 및 DFHSC2LS는 XML 스키마에서 <xsd:any> 및 xsd:anyType 사용을 지원합니다. <xsd:any> XML 스키마 요소를 사용하여 정의되지 않은 콘텐츠가 있는 XML 문서의 섹션을 설명할 수 있습니다. xsd:anyType은 단순하고 복잡한 모든 데이터 유형이 파생되는 기본 데이터 유형입니다. 데이터 콘텐츠에는 제한사항 또는 제한조건이 없습니다.

141 페이지의 『<xsd:choice>에 대한 지원』

<xsd:choice> 요소는 요소의 옵션 중 하나만 사용할 수 있음을 나타냅니다. CICS 지원은 다양한 맵핑 레벨에서 <xsd:choice> 요소에 대한 다양한 수준의 지원을 제공합니다.

『추상 요소 및 추상 데이터 유형에 대한 지원』

CICS 지원은 맵핑 레벨 2.2 이상에서 추상 요소 및 추상 데이터 유형에 대한 지원을 제공합니다. CICS 지원은 추상 요소와 추상 데이터 유형을 유사한 방식으로 대체 그룹에 맵핑합니다.

추상 요소 및 추상 데이터 유형에 대한 지원:

CICS 지원은 맵핑 레벨 2.2 이상에서 추상 요소 및 추상 데이터 유형에 대한 지원을 제공합니다. CICS 지원은 추상 요소와 추상 데이터 유형을 유사한 방식으로 대체 그룹에 맵핑합니다.

맵핑 레벨 2.2 이상의 추상 요소에 대한 지원

맵핑 레벨 2.2 이상의 경우 DFHSC2LS 및 DFHWS2LS는 추상 요소가 그룹의 올바른 멤버가 아니라는 점을 제외하고 대체 그룹과 거의 동일한 방식으로 추상 요소를 처리합니다. 대체 요소가 없는 경우에는 추상 요소가 <xsd:any> 요소로 처리되며 맵핑 레벨 2.1의 <xsd:any> 요소와 동일한 맵핑을 사용합니다.

다음 XML 스키마 단편은 추상 요소 대신 사용할 수 있는 두 가지 옵션을 지정합니다. 추상 요소 자체는 올바른 옵션이 아닙니다.

```
<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" />
```

이 XML 단편을 지원과 함께 처리하면 다음 COBOL 언어 구조가 생성됩니다.

```
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.

```

추상 요소에 대한 자세한 정보는 *W3C XML 스키마 파트 0: 입문서 Second Edition 스펙*(<http://www.w3.org/TR/xmlschema-0/#SubsGroups>)을 참조하십시오.

맵핑 레벨 2.2 이상의 추상 데이터 유형에 대한 지원

맵핑 레벨 2.2 이상의 경우 DFHSC2LS 및 DFHWS2LS는 추상 데이터 유형을 대체 그룹으로 처리합니다. 지원은 언어 구조에서 열거 필드와 새 컨테이너 이름을 생성합니다.

다음 XML 스키마 단편은 추상 유형 대신 사용할 수 있는 두 가지 대안을 지정합니다.

```

<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>

```

이 XML 단편을 지원과 함께 처리하면 다음 COBOL 언어 구조가 생성됩니다.

```

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).

```

언어 구조는 개별 카피북에 생성됩니다. option1에 대해 생성된 언어 구조는 하나의 카피북에 생성됩니다.

03 option1 PIC X(5).

option2에 대한 언어 구조는 다른 카피북에 생성됩니다.

03 option2 PIC X(10).

추상 데이터 유형에 대한 자세한 정보는 *W3C XML Schema Part 0: 입문서 Second Edition 스펙*(<http://www.w3.org/TR/xmlschema-0/#SubsGroups>)을 참조하십시오.

관련 참조:

139 페이지의 『<xsd:any> 및 xsd:anyType에 대한 지원』

DFHWS2LS 및 DFHSC2LS는 XML 스키마에서 <xsd:any> 및 xsd:anyType 사용을 지원합니다. <xsd:any> XML 스키마 요소를 사용하여 정의되지 않은 콘텐츠가 있는 XML 문서의 섹션을 설명할 수 있습니다. xsd:anyType은 단순하고 복잡한 모든 데이터 유형이 파생되는 기본 데이터 유형입니다. 데이터 콘텐츠에는 제한사항 또는 제한조건이 없습니다.

141 페이지의 『<xsd:choice>에 대한 지원』

<xsd:choice> 요소는 요소의 옵션 중 하나만 사용할 수 있음을 나타냅니다. CICS 지원은 다양한 맵핑 레벨에서 <xsd:choice> 요소에 대한 다양한 수준의 지원을 제공합니다.

144 페이지의 『대체 그룹에 대한 지원』

대체 그룹을 사용하여 상호 교환할 수 있는 XML 요소의 그룹을 정의할 수 있습니다. CICS 지원은 맵핑 레벨 2.2 이상에서 대체 그룹에 대한 지원을 제공합니다.

COBOL의 가변 반복 콘텐츠 핸들링 방법:

COBOL의 경우 포인터 산술식을 사용하여 데이터의 각 인스턴스를 처리함으로써 가변 반복 콘텐츠를 처리할 수 없습니다. 다른 프로그래밍 언어에는 이 제한사항이 없습니다. 이 예는 웹 서비스 애플리케이션에 대해 COBOL로 가변 반복 콘텐츠를 처리하는 방법을 보여줍니다.

이 기술은 **TRANSFORM** API 명령을 사용하여 XML을 애플리케이션 데이터로 변환하는 경우에도 적용됩니다. 다음 예제 WSDL 문서는 가변 횟수만큼 반복되는 8자 문자열로 구성되는 애플리케이션 데이터를 포함하는 웹 서비스를 나타냅니다.

```
<?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>

```

이 WSDL 문서를 DFHWS2LS를 통해 처리하면 다음 COBOL 언어 구조가 생성됩니다.

```
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).
```

8자 component 필드는 별도 구조 DFHWS-component에서 정의됩니다. 기본 데이터 구조는 applicationData이며 두 필드, component-num 및 component-cont를 포함합니다. component-num 필드는 존재하는 component 데이터 인스턴스 수를 나타내고 component-cont 필드는 component 필드의 연결 목록을 포함하는 컨테이너의 이름을 나타냅니다.

다음 COBOL 코드는 가변 반복 데이터 목록을 처리하는 한 가지 방법을 보여줍니다. 이 코드는 연계 섹션 배열을 사용하여 데이터의 후속 인스턴스를 처리합니다. 각 인스턴스는 DISPLAY 문을 사용하여 표시됩니다.

```

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.

```

위 코드는 가변 반복 콘텐츠 핸들링을 위한 일반적인 솔루션을 제공합니다. BIG-ARRAY 배열은 각 컴포넌트의 처음으로 차례로 이동하며 데이터 시작 시 고정된 상태를 유지하지 않습니다. 컴포넌트 데이터 구조는 다음 컴포넌트의 첫 번째 바이트를 가리키도록 이동합니다. COMPONENT-PTR은 필요한 경우 컴포넌트 데이터의 시작 위치를 복구하는 데 사용될 수 있습니다.

다음은 WSDL 문서를 준수하는 SOAP 메시지 예입니다.

```

<?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>

```

다음은 SOAP 메시지를 처리할 때 COBOL 프로그램에서 생성되는 출력입니다.

```

CPIH 20080115103151 component value is: VALUE1
CPIH 20080115103151 component value is: VALUE2
CPIH 20080115103151 component value is: VALUE3

```

웹 서비스 지원을 사용하여 웹 서비스 제공자 작성

고급 언어 데이터 구조에서 또는 WSDL 1.1 또는 WSDL 2.0을 준수하는 웹 서비스 설명에서 서비스 제공자 애플리케이션을 작성할 수 있습니다. CICS 웹 서비스 지원은 서비스 제공자 설정에서 CICS 애플리케이션을 배치하는 데 도움을 줍니다.

이 태스크 정보

지원을 사용하여 CICS 애플리케이션을 서비스 제공자로 배치할 때 다음 두 가지 옵션이 있습니다.

- 웹 서비스 설명을 시작하고 지원을 사용하여 언어 데이터 구조를 생성합니다.

기존 웹 서비스 설명에 부합되는 서비스 제공자를 구현할 때에는 이 옵션을 사용하지 않습니다.

- 언어 데이터 구조를 시작하고 지원을 사용하여 웹 서비스 설명을 생성합니다.

기존 프로그램을 웹 서비스로 노출시키고 웹 서비스 설명 및 SOAP 메시지에 프로그램 인터페이스의 측면을 노출시키려는 경우에 이 옵션을 사용하지 않습니다.

URI를 사용하여 서비스 제공자와 연관된 웹 서비스 설명을 노출시킬 수 있습니다. 이 URI는 접미부 `?wsdl`이 추가된 `WEBSERVICE`와 연관된 URI와 같은 경로를 가집니다. 이 경우 비즈니스 내 또는 외부의 요청자가 서비스 제공자와 연관된 WSDL 파일을 검색할 수 있게 됩니다.

웹 서비스 설명에서 서비스 제공자 애플리케이션 작성:

CICS 웹 서비스 지원을 사용하여 WSDL 1.1 또는 WSDL 2.0을 준수하는 웹 서비스 설명에서 서비스 제공자 애플리케이션을 작성할 수 있습니다.

시작하기 전에

서비스 제공자 애플리케이션을 작성하려면 먼저 다음 조건이 만족되어야 합니다.

- 웹 서비스 설명이 z/OS의 UNIX 파일에 있어야 하고 CICS 리전에서 적절한 제공자 모드 파이프라인을 작성해야 합니다.
- OMVS에 DFHLS2WS를 실행하는 사용자 ID를 정의해야 합니다.
- 사용자 ID는 z/OS UNIX 및 PDS 라이브러리에 대한 읽기 권한 및 **LOGFILE**, **WSBIND** 및 **WSDL** 매개변수에서 지정되는 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- Java를 실행할 수 있도록 사용자 ID에 충분한 저장영역을 할당해야 합니다. 지원되는 모든 Java 버전을 사용할 수 있습니다. 기본적으로 DFHWS2LS는 **JAVADIR** 매개변수에 지정된 Java 버전을 사용합니다.

이 태스크 정보

웹 서비스 지원을 사용하여 서비스 제공자 애플리케이션의 WSDL에서 언어 구조를 작성할 수 있습니다. IBM WSRR(WebSphere Service Registry and Repository) 서버에 저장되는 WSDL 문서를 사용할 수도 있습니다.

프로시저

1. DFHWS2LS 일괄처리 프로그램을 사용하여 웹 서비스 바인딩 파일 및 한 개 이상의 언어 데이터 구조를 생성하십시오. DFHWS2LS에는 사용자 애플리케이션이 필요로 하는 바인딩 파일 및 언어 구조를 작성할 수 있도록 사용자에게 유연성을 제공하는 대형 선택적 매개변수 세트가 포함되어 있습니다. 웹 서비스에 대해 기존 애플리케이션을 사용으로 설정할 때 이러한 옵션을 고려하십시오.
 - a. 데이터를 서비스 제공자 애플리케이션 프로그램에 전달하기 위해 **CICS**가 사용하는 메커니즘은?
 - 채널을 사용하여 컨테이너로 데이터를 전달하거나 **COMMAREA**를 사용할 수 있습니다. 채널 및 컨테이너가 권장됩니다. 이를 **PGMINT** 매개변수로 지정하십시오.
 - b. 생성할 언어는?
 - DFHWS2LS는 **COBOL**, **C/C++** 또는 **PL/I** 언어 데이터 구조를 생성할 수 있습니다. **LANG** 매개변수를 사용하여 언어를 지정하십시오.
 - c. 사용할 맵핑 레벨은?
 - 맵핑 레벨이 높을수록 런타임 시 문자 및 2진 데이터의 핸들링에 사용 가능한 제어 및 지원이 많아집니다. 일부 선택적 매개변수는 높은 맵핑 레벨에서만 사용 가능합니다. 사용 가능한 가장 높은 레벨의 맵핑을 사용하도록 권장됩니다. **MAPPING-LEVEL** 매개변수로 맵핑 레벨을 지정하십시오.
 - d. 웹 서비스 요청자가 사용하게 할 **URI**는?
 - **URI** 매개변수를 사용하여 상대 **URI**를 지정하십시오(예: **URI=/my/test/webservice**). 이 값은 **URIMAP** 자원을 작성할 때 **CICS**에서 사용됩니다.
 - e. 웹 서비스 요청 및 응답을 실행할 트랜잭션 및 사용자 **ID**는?
 - 서비스 요청자에 대한 응답을 구성하기 위해 별명 트랜잭션을 사용하여 애플리케이션을 실행할 수 있습니다. 별명 트랜잭션은 사용자 **ID** 아래 첨부됩니다.
 - 이를 **TRANSACTION** 및 **USERID** 매개변수로 지정하십시오. 이러한 값은 **URIMAP** 자원을 작성할 때 사용됩니다. 특정 트랜잭션을 사용하지 않으려는 경우에는 이러한 매개변수를 사용하지 마십시오.
 - f. **WSDL** 문서가 저장되는 위치는?
 - 로컬 파일 시스템 대신 **WSRR** 서버에서 **WSDL** 문서를 검색하려면 DFHWS2LS에서 특정 매개변수를 지정해야 합니다.
 - 최소한 **WSRR** 서버의 위치로 **WSRR-SERVER** 매개변수를 지정하고 **WSRR**에서 검색하려는 **WSDL** 문서의 이름으로 **WSRR-NAME** 매개변수를 지정해야 합니다.
 - **WSRR**을 사용하는 경우 지정할 기타 매개변수에 대한 정보는 79 페이지의 『DFHWS2LS: WSDL을 고급 언어로 변환』의 내용을 참조하십시오.

g. **WSRR** 서버에서 **WSDL** 문서를 검색하려는 경우 보안 연결을 사용하여 검색 하시겠습니까?

- 적절한 매개변수를 설정하여 **SSL(Secure Socket Layer)** 암호화를 사용하면 안전하게 **WSRR**과 상호 운용할 수 있습니다. 예를 들어, 244 페이지의 『웹 서비스 지원 및 **WSRR**에서 **SSL**을 사용하는 방법의 예제』의 내용을 참조하십시오.

- **DFHWS2LS**를 제출할 때 **CICS**는 웹 서비스 바인딩 파일을 생성하고 이 파일을 **WSBIND** 매개변수로 지정한 위치에 배치합니다. 언어 구조는 **PDSLIB** 매개변수로 지정한 파티션된 데이터 세트에 배치됩니다.

2. 생성된 웹 서비스 바인딩 파일을 웹 서비스 애플리케이션에 사용할 제공자 모드 **PIPELINE** 자원의 **pickup** 디렉토리에 복사하십시오. 바인딩 파일은 2진 모드로 복사해야 합니다.

3. 옵션: 웹 서비스 설명 또는 한 개 이상의 웹 서비스 설명을 포함하는 아카이브 파일을 웹 서비스 바인딩 파일과 같은 디렉토리에 복사하십시오. 아카이브 파일은 **.zip** 파일이어야 하며 파일 이름은 **WSDL** 파일 이름과 일치해야 합니다. 이 사본을 사용하여 **WSDL**을 검색할 수 있습니다.

4. 생성된 언어 구조와 인터페이스할 서비스 제공자 애플리케이션 프로그램을 작성하고 필수 비즈니스 로직을 구현하십시오.

5. **WEBSERVICE** 자원과 두 개의 **URIMAP** 자원을 작성하십시오.

- **WEBSERVICE** 자원은 **CICS**의 웹 서비스 바인딩 파일을 요약하고 런타임 시 사용됩니다.
- 첫 번째 **URIMAP** 자원은 **CICS**에 **WEBSERVICE** 자원과 특정 **URI**를 연관시키기 위한 정보를 제공합니다.
- 두 번째 **URIMAP** 자원은 **CICS**에 **WSDL** 아카이브 파일 또는 **WSDL** 문서와 특정 **URI**를 연관시키기 위한 정보를 제공합니다.
 - 이 **URI**는 접미부 **?wsdl**이 추가된 **WEBSERVICE**와 연관된 **URI**와 같은 경로를 가집니다.
 - 이 **URIMAP** 자원은 외부 요청자가 **URI**를 사용하여 **WSDL** 아카이브 파일 또는 **WSDL** 문서를 검색할 수 있도록 작성됩니다.
 - 웹 서비스 설명 또는 한 개 이상의 웹 서비스 설명을 포함하는 아카이브 파일이 웹 서비스 바인딩 파일과 같은 디렉토리에 복사된 경우에만 이 **URIMAP** 자원이 작성됩니다.
 - 픽업 디렉토리에 **WSDL** 아카이브 파일과 **WSDL** 문서가 포함된 경우 **URI**는 아카이브 파일에 있는 **WSDL**만 리턴합니다.
 - 이 기능은 파이프라인 스캔 조작을 사용하여 설치된 웹 서비스에서만 사용 가능합니다.

다음과 같은 방식으로 자원을 작성할 수 있습니다.

- a. **PIPELINE SCAN** 명령을 사용하여 WEBSERVICE 자원 및 URIMAP 자원을 동적으로 작성하십시오.
- b. 직접 자원을 정의합니다. CICS Explorer®를 사용하여 CICS 번들에서 WEBSERVICE 자원을 정의하는 경우 웹 서비스 바인딩 파일 및 WSDL 문서 또는 WSDL 아카이브 파일을 가져와서 번들에 포함시킬 수 있습니다. 그러면 웹 서비스 지원을 위한 URIMAP 정의를 생성하고 이러한 정의를 번들로 패키징할 수 있습니다. CICS Explorer를 사용하여 CICS 번들에서 자원을 작성하고 편집하는 작업에 대한 자세한 도움말은 CICS Explorer 제품 문서의 번들에 대한 작업의 내용을 참조하십시오.

결과

DFHWS2LS 제출에 문제점이 있거나 자원이 올바르게 설치되지 않는 경우 703 페이지의 『배치 오류 진단』의 내용을 참조하십시오.

데이터 구조에서 서비스 제공자 애플리케이션 작성:

CICS 웹 서비스 지원을 사용하여 고급 언어 데이터 구조에서 서비스 제공자 애플리케이션을 작성할 수 있습니다.

시작하기 전에

서비스 제공자 애플리케이션을 작성하기 전에 다음 전제조건이 완료되었는지 확인하십시오.

- 고급 언어 데이터 구조가 다음 기준을 충족시켜야 합니다.
 - 데이터 구조는 소스 프로그램(예: COBOL 카피북)과 별도로 정의되어야 합니다.
 - PL/I 또는 COBOL 애플리케이션 프로그램이 입력 및 출력에 서로 다른 데이터 구조를 사용하는 경우 데이터 구조는 파티션된 데이터 세트에 있는 두 개의 다른 멤버에서 정의되어야 합니다. 입력 및 출력에 같은 구조를 사용하는 경우 이 구조는 단일 멤버에서 정의되어야 합니다.

C 및 C++에서 데이터 구조는 파티션된 데이터 세트에서 같은 멤버에 있을 수 있습니다.

- 처리하는 데이터 구조는 랩퍼 프로그램을 사용하는지 여부에 따라 다릅니다.
 - 랩퍼 프로그램을 사용하는 경우 카피북은 랩퍼 프로그램에 대한 인터페이스입니다.
 - 랩퍼 프로그램을 사용하지 않는 경우 카피북은 비즈니스 로직에 대한 인터페이스입니다.
- 언어 구조는 파티션된 데이터 세트에서 사용 가능해야 하며 CICS 리전에서 적절한 PIPELINE 자원을 작성해야 합니다.
 - OMVS에 DFHLS2WS를 실행하는 사용자 ID를 정의해야 합니다.

- 사용자 ID는 z/OS UNIX 및 PDS 라이브러리에 대한 읽기 권한 및 **LOGFILE**, **WSBIND** 및 **WSDL** 매개변수에서 지정되는 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다. 지원되는 모든 Java 버전을 사용할 수 있습니다. 기본적으로 DFHLS2WS는 **JAVADIR** 매개변수에 지정된 Java 버전을 사용합니다.

프로시저

데이터 구조에서 서비스 제공자 애플리케이션을 작성하려면 다음 단계를 따르십시오.

1. 서비스 제공자 애플리케이션 인터페이스에서 채널 및 다수의 컨테이너를 사용하는 경우 XML로 인터페이스를 설명하는 채널 설명 문서를 작성하십시오. 채널 설명 문서는 z/OS UNIX의 적절한 디렉토리에 넣어야 합니다. CICS는 이 문서를 사용하여 채널의 여러 컨테이너에서 SOAP 메시지를 구성하고 구성 해제합니다. 또는 한 채널에서 한 개의 컨테이너를 사용하고 채널 설명 문서를 작성하지 않을 수 있습니다.
 - 채널 설명 문서를 작성하는 방법에 대한 자세한 정보는 158 페이지의 『채널 설명 문서 작성』의 내용을 참조하십시오.
2. DFHLS2WS 일괄처리 프로그램을 사용하여 언어 구조에서 웹 서비스 설명 및 웹 서비스 바인딩 파일을 생성하십시오. DFHLS2WS에는 사용자 애플리케이션이 필요로 하는 바인딩 파일 및 언어 구조를 작성할 수 있도록 사용자에게 유연성을 제공하는 대형 선택적 매개변수 세트가 포함되어 있습니다. 웹 서비스가 기존 애플리케이션을 사용으로 설정할 때 다음 옵션을 고려하십시오.
 - a. 데이터를 서비스 제공자 애플리케이션 프로그램에 전달하기 위해 CICS가 사용하는 메커니즘은?
 - 채널을 사용하여 컨테이너로 데이터를 전달하거나 COMMAREA를 사용할 수 있습니다. **PGMINT** 매개변수를 사용하여 메커니즘을 지정하십시오. 애플리케이션 인터페이스가 채널 및 다수의 컨테이너를 사용하는 경우 **REQUEST-CHANNEL** 매개변수 및 선택적으로 **RESPONSE-CHANNEL**을 지정하십시오. 이러한 매개변수는 맵핑 레벨이 3.0 이상인 경우에만 사용할 수 있습니다.
 - b. 생성할 웹 서비스 설명(WSDL 문서)의 레벨은?
 - CICS는 WSDL 1.1 또는 WSDL 2.0 문서를 준수하는 설명을 생성합니다. 서비스 제공자 애플리케이션이 두 레벨의 WSDL을 모두 준수하는 요청을 지원하도록 하려는 경우 **WSDL_1.1** 및 **WSDL_2.0** 매개변수에 값을 지정하십시오. 둘 이상의 WSDL 매개변수를 사용할 때 파일 이름이 서로 다른지 확인하십시오. 이 스펙은 두 개의 웹 서비스 설명과 한 개의 바인딩 파일을 생성합니다.
 - c. 사용할 SOAP 프로토콜의 버전은?

- **SOAPVER** 매개변수로 버전을 지정할 수 있습니다. ALL 값 사용이 권장됩니다. 이 값은 SOAP 1.2 메시지 핸들러를 사용하여 구성되는 파이프라인에 웹 서비스를 설치해야 하는 단점도 있지만 웹 서비스 설명에 대한 바인딩으로 SOAP 1.1 또는 SOAP 1.2를 사용할 수 있는 유연성을 제공합니다. 이 매개변수는 **MINIMUM-RUNTIME-LEVEL**이 2.0 이상인 경우에만 사용할 수 있습니다.
- d. 사용할 맵핑 레벨은?
- 맵핑 레벨이 높을수록 런타임 시 문자 및 2진 데이터의 핸들링에 사용 가능한 제어 및 지원이 많아집니다. 일부 선택적 매개변수는 높은 맵핑 레벨에서만 사용 가능합니다. **MAPPING-LEVEL** 매개변수에 사용 가능한 가장 높은 레벨의 맵핑을 지정하도록 권장됩니다.
- e. 웹 서비스 요청자가 사용하게 할 URI는?
- **URI** 매개변수를 사용하여 절대 URI를 지정하십시오(예: **URI**=http://www.example.org:80/my/test/webservice). 이 주소의 상대 파트인 /my/test/webservice는 URIMAP 자원을 작성할 때 사용됩니다. 전체 URI는 웹 서비스 설명에서 <soap:address> 요소로 사용됩니다. 이 사용법은 HTTP 및 WebSphere MQ URI 둘 다에 대해 true입니다.
- f. WSDL 문서를 WSRR(IBM WebSphere Service Registry and Repository)에 공개하시겠습니까?
- WSDL 문서를 WSRR에 공개하려는 경우 DFHLS2WS에서 **WSRR-SERVER** 매개변수를 지정해야 합니다. WSRR을 사용할 때 지정할 수 있는 매개변수에 대한 자세한 정보는 63 페이지의 『DFHLS2WS: 고급 언어 대 WSDL 변환』의 내용을 참조하십시오.
- g. WSDL 문서를 WSRR 서버에 공개하려는 경우 보안 연결을 사용하여 공개하시겠습니까?
- 적절한 매개변수를 설정하여 SSL(Secure Socket Layer) 암호화를 사용하면 안전하게 WSRR과 상호 운용할 수 있습니다. 예를 들어, 244 페이지의 『웹 서비스 지원 및 WSRR에서 SSL을 사용하는 방법의 예제』의 내용을 참조하십시오.
 - DFHLS2WS를 제출할 때 CICS는 웹 서비스 바인딩 파일을 생성하고 이 파일을 **WSBIND** 매개변수로 지정한 위치에 배치합니다. 생성된 웹 서비스 설명은 **WSDL**, **WSDL_1.1** 또는 **WSDL_2.0** 매개변수로 지정한 위치에 배치됩니다.
 - DFHLS2WS에서 WSRR 매개변수를 사용한 경우 WSDL 문서는 사용자가 지정한 WSRR 서버에 공개됩니다.
3. 생성된 웹 서비스 설명을 검토하고 필요한 사용자 정의를 수행하십시오. 자세한 정보는 159 페이지의 『생성된 웹 서비스 설명 문서 사용자 정의』의 내용을 참조하십시오.

4. 웹 서비스 바인딩 파일을 웹 서비스 애플리케이션에 사용할 제공자 모드 파이프라인의 픽업 디렉토리에 복사하십시오. 웹 서비스 바인딩 파일은 2진 모드로 복사해야 합니다.
5. 옵션: 웹 서비스 설명 또는 한 개 이상의 웹 서비스 설명을 포함하는 아카이브 파일을 웹 서비스 바인딩 파일과 같은 디렉토리에 복사하십시오. 아카이브 파일은 .zip 파일이어야 하며 파일 이름은 WSDL 파일 이름과 일치해야 합니다. 이 사본을 사용하여 WSDL을 검색할 수 있습니다.
6. **PIPELINE SCAN** 명령을 사용하여 WEBSERVICE 자원 및 URIMAP 자원을 동시에 작성하십시오. WEBSERVICE 자원은 CICS의 웹 서비스 바인딩 파일을 요약하고 런타임 시 사용됩니다.
 - 첫 번째 URIMAP 자원은 CICS에 WEBSERVICE 자원과 특정 URI를 연관시키기 위한 정보를 제공합니다.
 - 두 번째 URIMAP 자원은 CICS에 WSDL 아카이브 파일 또는 WSDL 문서와 특정 URI를 연관시키기 위한 정보를 제공합니다.
 - 이 URI는 접미부 ?wsdl이 추가된 WEBSERVICE와 연관된 URI와 같은 경로를 가집니다.
 - 이 URIMAP 자원은 외부 요청자가 URI를 사용하여 WSDL 아카이브 파일 또는 WSDL 문서를 검색할 수 있도록 작성됩니다.
 - 웹 서비스 설명 또는 한 개 이상의 웹 서비스 설명을 포함하는 아카이브 파일이 웹 서비스 바인딩 파일과 같은 디렉토리에 복사된 경우에만 이 URIMAP 자원이 작성됩니다.
 - 픽업 디렉토리에 WSDL 아카이브 파일과 WSDL 문서가 포함된 경우 URI는 아카이브 파일에 있는 WSDL만 리턴합니다.
 - 이 기능은 파이프라인 스캔 조작을 사용하여 설치된 웹 서비스에서만 사용 가능합니다.

또는 직접 자원을 정의할 수 있지만 권장되지 않습니다.

결과

CICS 자원을 성공적으로 작성한 경우에 서비스 제공자 애플리케이션의 작성이 완료됩니다.

DFHLS2WS 제출에 문제점이 있거나 자원이 올바르게 설치되지 않는 경우 703 페이지의 『배치 오류 진단』의 내용을 참조하십시오.

다음에 수행할 작업

서비스에 액세스할 웹 서비스 요청자를 개발해야 하는 모든 사용자가 사용할 수 있도록 웹 서비스 설명을 공개하십시오.

채널 설명 문서 작성:

서비스 제공자 애플리케이션이 여러 컨테이너와의 채널 인터페이스를 사용하는 경우 채널 설명 문서를 작성합니다.

이 태스크 정보

XML 편집기를 사용하여 채널 설명 문서를 작성하십시오. 채널 설명을 위한 스키마는 channel.xsd라고 하고 /usr/lpp/cicsts/cicsts52/schemas/channel 디렉토리(여기서, /usr/lpp/cicsts/cicsts52는 CICS 파일의 기본 설치 디렉토리임)에 있습니다.

프로시저

1. <채널> 요소 및 CICS 채널 네임스페이스를 사용하여 XML 문서를 작성하십시오.

```
<channel name="myChannel" xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```

2. 애플리케이션 프로그램 인터페이스가 채널에서 사용하는 모든 컨테이너에 <컨테이너> 요소를 추가하십시오. 이름, 유형 및 속성을 사용하여 각 컨테이너를 설명해야 합니다. 다음 예제는 서로 다른 속성 값을 가지는 여섯 개의 컨테이너를 보여줍니다.

```
<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>
```

이 구조 요소는 파티션된 데이터 세트 멤버에 있는 언어 구조로 콘텐츠가 정의됨을 표시합니다.

3. z/OS UNIX에서 XML 문서를 저장하십시오.

채널 스키마

채널 설명 문서는 다음 스키마를 준수해야 합니다.

```
<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"> ❶
    <complexType>
      <sequence>
        <element name="container" maxOccurs="unbounded" "unbounded" minOccurs="0"> ❷
          <complexType>
            <sequence>
              <element name="structure" minOccurs="0"> ❸
                <complexType>
                  <attribute name="location" type="string" use="required"/>

```

```

        <attribute name="structure" type="string" use="optional"/>
    </complexType>
</element>
</sequence>
<attribute name="name" type="tns:name16Type" use="required"/>
<attribute name="type" type="tns:typeType" use="required"/>
<attribute name="use" type="tns:useType" use="required"/>
</complexType>
</element>
</sequence>
<attribute name="name" type="tns:name16Type" use="optional" />
</complexType>
</element>
<simpleType name="name16Type">
    <restriction base="string">
        <maxLength value="16"/>
    </restriction>
</simpleType>
<simpleType name="typeType">
    <restriction base="string">
        <enumeration value="char"/>
        <enumeration value="bit"/>
    </restriction>
</simpleType>
<simpleType name="useType">
    <restriction base="string">
        <enumeration value="required"/>
        <enumeration value="optional"/>
    </restriction>
</simpleType>
</schema>

```

1. 이 요소는 CICS 채널을 나타냅니다.
2. 이 요소는 채널 내 CICS 컨테이너를 나타냅니다.
3. 구조는 '비트' 모드 컨테이너와 함께만 사용할 수 있습니다. '위치' 속성은 컨테이너의 콘텐츠를 맵핑하는 파일의 위치를 표시합니다. '구조' 속성은 C 및 C++에서 사용되어 구조의 이름을 표시할 수 있습니다.

다음에 수행할 작업

DFHLS2WS를 실행하여 웹 서비스 제공자 애플리케이션에 대한 맵핑 및 WSDL 문서를 작성하십시오. DFHLS2WS는 채널 설명 문서에서 컨테이너가 지정되는 순서대로 채널의 맵핑을 WSDL 문서에 넣습니다.

생성된 웹 서비스 설명 문서 사용자 정의:

DFHLS2WS에 의해 생성되는 웹 서비스 설명(WSDL) 문서에는 공개 전 변경하기에 적합할 수 있는 일부 자동 생성 콘텐츠가 있습니다. WSDL 문서를 사용자 정의하면 웹 서비스 바인딩 파일이 재생성될 수 있으며 일부 경우에는 랩퍼 프로그램이 작성될 수 있습니다.

이 태스크 정보

다음 단계에 따라 생성된 웹 서비스 설명 문서를 사용자 정의하십시오.

프로시저

1. HTTPS에 대한 지원을 광고하거나 WebSphere MQ를 사용하여 통신하려는 경우 DFHLS2WS에서 **URI** 매개변수를 사용하여 절대 URI를 설정하십시오. **URI** 매개변수를 사용하지 않은 경우 WSDL 문서의 끝에서 <wsdl:service> 및 <wsdl:binding> 요소를 변경해야 합니다. 생성된 WSDL에는 이러한 변경사항 작성을 지원하는 주석이 포함됩니다. 이러한 요소를 변경하더라도 웹 서비스 바인딩 파일을 재생성하지 않아도 됩니다.
2. 웹 서비스의 네트워크 위치를 제공하려면 DFHLS2JS에서 **URI** 매개변수를 사용하여 절대 URI를 설정하십시오. **URI** 매개변수를 사용하지 않은 경우 wsdl:service 요소의 soap:address에 세부사항을 추가하십시오.
 - a. HTTP 기반 프로토콜을 사용하는 경우 my-server를 CICS 리전의 TCP/IP 호스트 이름으로 대체하고 my-port를 TCPIP SERVICE 자원의 포트 번호로 대체하십시오.
 - b. WebSphere MQ를 전송 프로토콜로 사용하는 경우 myQueue를 해당 큐의 이름으로 대체하십시오.

웹 서비스 바인딩 파일에 대한 변경 없이도 이러한 변경을 수행할 수 있습니다.

WSBind 파일을 재생성하지 않고 포트 이름 및 네임스페이스를 변경하는 경우 런타임 레벨 2.1 이상에서 모니터링 정보가 틀릴 수 있습니다.

3. WSDL 문서의 자동으로 생성된 이름이 사용자의 목적에 적합한지 여부를 고려해 보십시오. 이러한 값의 이름을 바꿀 수 있습니다.
 - WSDL 문서의 targetNamespace
 - WSDL 문서 내 XML 스키마의 targetNamespace
 - <wsdl:portType> 이름
 - <wsdl:operation> 이름
 - <wsdl:binding> 이름
 - <wsdl:service> 이름
 - WSDL 문서 내 XML 스키마의 필드 이름.

이러한 값은 클라이언트 프로그램을 코드화할 프로그래밍 방식 인터페이스 일부를 구성합니다. 생성된 이름의 의미가 분명하지 않으면 장기간에 걸친 애플리케이션 코드의 유지보수가 좀 더 어려울 수 있습니다. DFHLS2WS **REQUEST-NAMESPACE** 및 **RESPONSE-NAMESPACE** 매개변수를 사용하여 XML 스키마의 targetNamespace 및 WSDL 문서의 targetNamespace를 변경하기 위한 **WSDL-NAMESPACE** 매개변수를 변경하십시오.

이러한 값을 변경하는 경우 DFHWS2LS를 사용하여 웹 서비스 바인딩 파일을 재생성해야 합니다. 생성되는 언어 구조는 기존 언어 구조와 같지 않지만 기존 애플

리케이션과는 호환 가능하기 때문에 애플리케이션 변경이 필요하지 않습니다. 그러나 새 언어 구조를 무시하고 원래 구조로 새 웹 서비스 바인딩 파일을 사용할 수 있습니다.

4. XML 스키마에서 노출된 COMMAREA 필드가 적합한지 고려하십시오. 웹 서비스 클라이언트 개발자에게 유용하지 않은 필드 제거를 고려할 수 있습니다.
 - 출력 값에만 사용되는 필드는 입력 데이터 구조를 맵핑하는 스키마에서 제거할 수 있습니다.
 - 필터 필드입니다.
 - 자동으로 생성되는 어노테이션입니다.

이러한 변경사항을 작성하는 경우 DFHWS2LS를 사용하여 웹 서비스 바인딩 파일을 재생성해야 합니다. 생성되는 새 언어 구조가 원래 언어 구조와 호환 가능하지 않기 때문에 랩퍼 프로그램을 작성하여 새 표시에서 이전 표시로 데이터를 맵핑해야 합니다. 이 랩퍼 프로그램은 대상 애플리케이션 프로그램에 대해 EXEC CICS LINK 명령을 수행한 다음 리턴된 데이터를 맵핑해야 합니다.

이 사용자 정의 레벨에서는 가장 많은 노력이 필요하지만 웹 서비스 클라이언트 개발자에게 가장 유용한 프로그래밍 방식 인터페이스가 생성됩니다.

5. DFHWS2LS를 통해 생성된 WSDL 문서를 넣어 새 언어 구조를 작성하려는 경우 WSDL 문서에 어노테이션을 보관할지 여부를 결정하십시오. 어노테이션은 DFHWS2LS가 언어 구조를 생성할 때 일반 맵핑 규칙을 대체합니다. 맵핑 규칙을 대체할 때 생성된 언어 구조가 DFHLS2WS에서 사용된 버전과 호환 가능한지 확인하십시오. 기본 맵핑 규칙을 사용하여 언어 구조를 생성하려면 어노테이션을 제거하십시오.

결과

사용자 정의한 WSDL 문서를 WSRR(IBM WebSphere Service Registry and Repository) 서버에 공개하려면 WSRR 인터페이스를 사용하여 수동으로 공개해야 합니다. 다음 위치에서 WSRR에 대한 자세한 정보를 찾을 수 있습니다. WebSphere Service Registry and Repository.

예

WSDL 문서의 예제는 생성된 WSDL 문서의 예제를 참조하십시오.

SOAP 결함 전송:

서비스 제공자에서 CICS API를 사용하여 웹 서비스 요청자에게 SOAP 결함을 보낼 수 있습니다. 이 결함은 파이프라인에서 서비스 제공자 애플리케이션 또는 헤더 처리 프로그램이 발행할 수 있습니다.

시작하기 전에

API를 사용하려면 서비스 제공자 애플리케이션이 채널 및 컨테이너를 사용해야 합니다. 애플리케이션이 COMMAREA를 사용하는 경우 SOAP 결함 메시지를 작성하는 데 채널 및 컨테이너를 사용하는 랩퍼 프로그램을 작성하십시오. 헤더 처리 프로그램의 API는 CICS 제공 SOAP 메시지 핸들러가 직접 호출하는 경우에만 사용할 수 있습니다.

이 태스크 정보

애플리케이션 로직이 요청을 충족시킬 수 없는 경우(예: 요청 메시지에 기본적인 문제점이 있는 경우) 웹 서비스 요청자에게 SOAP 결함을 발행하려고 할 수 있습니다. CICS에서는 SOAP 결함 발행을 오류로 고려하지 않기 때문에 오류 처리가 아닌 정상적인 메시지 응답 파이프라인 처리가 발생할 수 있음에 유의하십시오. 트랜잭션을 롤백하지 않으려면 애플리케이션 프로그램을 사용해야 합니다.

프로시저

1. 프로그램에서 **EXEC CICS SOAPFAULT CREATE** 명령을 사용하여 SOAP 결함을 전송하십시오(SOAPFAULT CREATE 참조).
2. 명령에 **CLIENT** 또는 **SERVER** 옵션을 추가하십시오. 이 옵션은 문제점이 발생한 위치가 클라이언트 측인지 또는 서버 측인지 표시합니다.
 - **CLIENT**는 문제점이 수신한 요청 메시지에 있음을 표시합니다.
 - **SERVER**는 요청 메시지가 CICS에서 처리될 때 문제점이 발생함을 표시합니다. 이 문제점은 애플리케이션 프로그램에 있을 수 있습니다. 예를 들어, 요청을 충족시킬 수 없거나 파이프라인 처리 중 발생하는 기본적인 문제점이 될 수 있습니다.
3. 서비스 제공자가 결함을 발행한 이유의 요약을 제공하는 **FAULTSTRING** 옵션 및 **FAULTSTRLEN** 옵션에 지정된 그 길이를 추가하십시오. 이 옵션의 콘텐츠는 XML로 되어 있습니다. 애플리케이션이 제공하는 모든 데이터는 XML 문서에 직접 포함시키기에 적합한 형식이어야 합니다. 애플리케이션은 일부 문자를 XML 엔티티로 지정해야 할 수도 있습니다. 예를 들어, < 문자가 XML 태그 시작부가 아닌 다른 위치에 사용되면 애플리케이션은 이 문자를 <로 변경해야 합니다. 다음 예제는 잘못된 **FAULTSTRING** 옵션을 보여줍니다.

```
dc1 msg_faultString char(*) constant('Error: Value A < Value B');
```

이 **FAULTSTRING** 옵션을 지정하는 올바른 방식은 다음과 같습니다.

```
dc1 msg_faultString char(*) constant('Error: Value A &lt; Value B');
```

팁: XML 엔티티 사용을 피하기 위해 XML CDATA 구성으로 데이터를 랩퍼할 수 있습니다. XML 프로세서는 이 구성의 문자 데이터를 구문 분석하지 않습니다. 이 메소드를 사용하여 다음 **FAULTSTRING** 옵션을 지정할 수 있습니다.

```
dc1 msg_faultString char(*) constant('<![CDATA[Error: Value A < Value B]]>');
```

4. 서비스 제공자가 결함을 발행한 이유의 세부사항을 제공하는 **DETAIL** 옵션 및 **DETAILLENGTH** 옵션에 지정된 그 길이를 코드화하십시오. 이 옵션의 콘텐츠는 XML로 되어 있습니다. **FAULTSTRING** 옵션에 대한 지침과 동일한 지침이 **DETAIL** 옵션에 적용됩니다.
5. 옵션: 헤더 처리 프로그램에서 API를 호출하는 경우 파이프라인 구성 파일에서 프로그램을 정의하십시오. 헤더 처리 프로그램은 `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` 또는 `<cics_soap_1.2_handler_java>` 요소에서 정의됩니다.

결과

프로그램이 이 명령을 발행하는 경우 CICS는 적절한 SOAP 레벨에서 SOAP 결함 응답 메시지를 작성합니다. 서비스 제공자 애플리케이션이 이 명령을 발행하는 경우 SOAP 응답을 작성하고 DFHRESPONSE 컨테이너에 넣지 않아도 됩니다. 파이프라인은 메시지 핸들러를 통해 SOAP 결함을 처리하고 웹 서비스 제공자에게 응답을 전송합니다.

예제

SOAPFAULT CREATE 명령에는 웹 서비스 요청자에게 적절하게 응답할 수 있도록 유연성을 제공하는 여러 옵션이 있습니다. 다음은 SOAP 1.1 및 SOAP 1.2 둘 다에 사용할 수 있는 SOAP 결함을 작성하는 완료된 명령의 단순 예제입니다.

```
EXEC CICS SOAPFAULT CREATE CLIENT
      DETAIL(msg_detail)
      DETAILLENGTH(length(msg_detail))
      FAULTSTRING(msg_faultString)
      FAULTSTRLEN(length(msg_faultString));
```

다음 값을 사용하여 `msg_detail` 및 `msg_faultString`을 코드화할 수 있습니다.

```
dcl 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>');
dcl msg_faultString char(*) constant('Something went wrong');
```

웹 서비스 지원을 사용하여 웹 서비스 요청자 작성

WSDL 1.1 또는 WSDL 2.0을 준수하는 웹 서비스 설명에서 서비스 요청자 애플리케이션을 작성할 수 있습니다. CICS 웹 서비스 지원은 서비스 요청자 설정에서 CICS 애플리케이션을 배치하는 데 도움을 줍니다.

시작하기 전에

웹 서비스 설명이 z/OS UNIX의 파일에 있거나 IBM WSRR(WebSphere Services Registry and Repository) 서버에 공개되어야 하고 요청자 모드 파이프라인이 CICS 리전에 설치되어야 합니다.

Java를 실행할 수 있도록 사용자 ID에 충분한 저장영역을 할당해야 합니다. 지원되는 모든 Java 버전을 사용할 수 있습니다. 기본적으로 DFHWS2LS는 **JAVADIR** 매개변수에 지정된 Java 버전을 사용합니다.

이 태스크 정보

CICS 웹 서비스 지원을 사용하여 CICS 애플리케이션을 서비스 요청자로 배치할 때 웹 서비스 설명을 시작하고 이 설명으로부터 언어 데이터 구조를 생성해야 합니다.

프로시저

1. DFHWS2LS 일괄처리 프로그램을 사용하여 웹 서비스 바인딩 파일 및 한 개 이상의 언어 구조를 생성하십시오. 웹 서비스 설명에서 서비스 요청자 애플리케이션을 작성할 때 다음 옵션을 고려하십시오.
 - 사용할 맵핑 레벨은? 맵핑 레벨이 높을수록 런타임 시 문자 및 2진 데이터의 핸들링에 사용 가능한 제어 및 지원이 많아집니다. 일부 선택적 매개변수는 높은 맵핑 레벨에서만 사용 가능합니다. 사용 가능한 가장 높은 레벨의 맵핑을 사용하도록 권장됩니다.
 - 런타임 시 데이터를 변환할 때 사용할 코드 페이지는? CICS 리전의 코드 페이지와는 다른 특정 코드 페이지를 애플리케이션에 사용하려는 경우에 **CCSID** 매개변수를 사용하십시오.
 - 웹 서비스 설명에서 선언된 조작의 서브세트를 지원하시겠습니까? 매우 큰 웹 서비스 설명을 가지고 있고 서비스 요청자 애플리케이션이 특정 수의 조작을 지원하게 하려는 경우 **OPERATION** 매개변수를 사용하여 원하는 조작을 나열하십시오. 각 조작은 공백으로 구분되어야 하고 대소문자를 구분합니다.
 - WSDL 문서가 저장되는 위치는? DFHWS2LS에 대한 입력으로 사용할 WSDL 문서가 **WSRR** 서버에 저장되는 경우 특정 매개변수가 지정된 DFHWS2LS를 실행하여 해당 문서를 검색할 수 있습니다. **WSRR-SERVER** 매개변수를 사용하여 **WSRR** 서버의 위치를 지정하고 **WSRR-NAME** 매개변수를 사용하여 검색할 WSDL 문서의 이름을 지정하십시오. **WSRR**과 상호작용하기 위해 사용할 수 있는 DFHWS2LS의 기타 매개변수에 대한 정보는 79 페이지의 『DFHWS2LS: WSDL을 고급 언어로 변환』의 내용을 참조하십시오.
 - **WSRR** 서버에서 WSDL 문서를 검색하려는 경우 보안 연결을 사용하여 검색하시겠습니까? 웹 서비스 지원에서 SSL(Secure Socket Layer) 암호화를 사용하여 안전하게 **WSRR**과 상호 운용할 수 있습니다. 예를 들어, 244 페이지의 『웹 서비스 지원 및 **WSRR**에서 SSL을 사용하는 방법의 예제』의 내용을 참조하십시오.

DFHWS2LS를 사용할 때 **PROGRAM**, **URI**, **TRANSACTION** 및 **USERID**와 같은 매개변수를 지정하지 마십시오. 이러한 매개변수는 서비스 제공자 애플리케이션에만 적

용되며 이러한 매개변수를 지정하면 제공자 모드 웹 서비스 바인딩 파일이 생성되는 원인이 됩니다. 웹 서비스 바인딩 파일 외에도 프로그램은 언어 데이터 구조를 생성합니다.

2. 로그 파일을 확인하여 DFHWS2LS이 바인딩 파일 및 언어 구조를 생성하는 동안 문제점이 발생했는지 여부를 확인하십시오. CICS는 웹 서비스 설명의 일부 요소 또는 옵션을 지원하지 않을 수 있습니다. 경고 또는 오류 메시지가 발생한 경우 제공되는 조언을 읽고 적절한 조치를 취하십시오. 일괄처리 프로그램을 재실행해야 할 수도 있습니다.
 3. 웹 서비스 바인딩 파일을 웹 서비스 애플리케이션에 사용할 요청자 모드 파이프라인의 픽업 디렉토리에 복사하십시오.
 4. PIPELINE 자원이 서비스 요청자 애플리케이션에 대해 구성되는지 확인하십시오. **MODE** 매개변수의 값은 파이프라인이 요청자 웹 서비스 애플리케이션을 지원하는지 또는 제공자 웹 서비스 애플리케이션을 지원하는지 표시합니다.
 5. 웹 서비스의 파이프라인에서 정확한 SOAP 프로토콜이 지원되는지 확인하십시오. **SOAPLEVEL** 매개변수는 지원되는 버전을 표시합니다. 서비스 요청자 모드에서 웹 서비스의 바인딩은 파이프라인에서 지원되는 SOAP의 버전과 일치해야 합니다. SOAP 1.2를 지원하는 서비스 요청자 파이프라인에 SOAP 1.1 바인딩을 사용하여 웹 서비스를 설치할 수 없습니다.
 6. 파이프라인에 대해 구성된 제한시간이 서비스 요청자 애플리케이션에 대해 적절한지 확인하십시오. 제한시간은 PIPELINE 자원에서 RESPWAIT 속성의 값으로 표시됩니다. 파이프라인에서 제한시간이 지정되지 않은 경우에는 전송 기본값이 사용됩니다.
 - HTTP에 대한 기본 제한시간은 10초입니다.
 - WebSphere MQ에 대한 기본 제한시간은 60초입니다.
- CICS 리전의 각 트랜잭션에는 디스패처 제한시간이 있습니다. 이 값이 두 프로토콜 중 하나의 기본값보다 적으면 제한시간 초과가 디스패처와 함께 발생합니다.
7. 옵션: 런타임 시 웹 서비스에 대한 유효성 검증을 켤 수 있도록 웹 서비스 바인딩 파일과 같은 픽업 디렉토리에 웹 서비스 설명을 복사하십시오.
 8. WEBSERVICE 자원을 작성하십시오. 이 자원은 CICS의 웹 서비스 바인딩 파일을 포함하며 런타임 시 사용됩니다.

다음과 같은 방식으로 이를 수행할 수 있습니다.

- a. **PIPELINE SCAN** 명령을 사용하여 WEBSERVICE 자원을 동적으로 작성하십시오.
- b. 직접 자원을 정의합니다. CICS Explorer를 사용하여 CICS 번들에서 WEBSERVICE 자원을 정의하는 경우 웹 서비스 바인딩 파일 및 WSDL 문서 또는 WSDL 아카이브 파일을 가져와서 번들에 포함시킬 수 있습니다. 그러면 웹 서비스 지원을 위한 URIMAP 정의를 생성하고 이러한 정의를 번들로 패키

정할 수 있습니다. CICS Explorer를 사용하여 CICS 번들에서 자원을 작성하고 편집하는 작업에 대한 자세한 도움말은 CICS Explorer 제품 문서의 번들에 대한 작업의 내용을 참조하십시오.

9. 통신 로직을 대체할 수 있는 랩퍼 프로그램을 작성하십시오. 1단계에서 생성된 언어 데이터 구조를 사용하여 랩퍼 프로그램을 작성하십시오. 랩퍼 프로그램에서 **EXEC CICS INVOKE SERVICE** 명령을 사용하여 웹 서비스와 통신하십시오. 이 명령에는 다음 옵션이 포함됩니다.

- 웹 서비스의 URI
- 웹 서비스가 호출되는 조작

웹 서비스를 호출할 때 웹 서비스의 URI에 대한 정보를 포함하는 URIMAP 자원을 지정할 수 있습니다. URIMAP 자원을 사용하는 대신 INVOKE SERVICE 명령에서 직접 이 정보를 지정할 수 있습니다. 그러나 URIMAP 자원의 사용은 서비스 제공자의 URI가 변경되는 경우 사용자 애플리케이션을 다시 컴파일하지 않아도 됨을 의미합니다. 후속 요청을 위해 애플리케이션이 재사용하거나 같은 서비스를 호출하는 다른 애플리케이션이 재사용할 수 있도록 URIMAP 자원을 통해 CICS가 클라이언트 연결 사용 후 이 연결을 열린 상태로 유지하는 연결 풀 구현을 선택할 수도 있습니다. PIPELINE SCAN 명령은 서비스 요청자의 URIMAP 자원을 작성하지 않으므로 애플리케이션 개발에서 HTTP 클라이언트로 CICS에 대한 URIMAP 자원 작성의 지시사항에 따라 URIMAP 자원을 직접 정의해야 합니다.

결과

CICS 자원을 성공적으로 작성한 경우에 서비스 요청자 애플리케이션의 작성이 완료됩니다.

PIPELINE 자원의 구성 확인

다음 인터페이스로 PIPELINE의 구성을 확인할 수 있습니다.

CICS Explorer



CICS Explorer 관리 보기

파이프라인 보기를 사용하십시오.

CICSplex® SM

PIPELINE 정의 보기

CEMT



INQUIRE PIPELINE 명령

CICS SPI

INQUIRE PIPELINE 명령

도구를 사용하여 웹 서비스 작성

웹 서비스 지원 JCL을 사용하는 대신 z Systems용 Rational Developer을 사용하거나 자체 Java 프로그램을 작성하여 CICS에서 필수 파일을 작성할 수 있습니다.

프로시저

1. 두 가지 선택사항이 있습니다.

- z Systems용 Rational Developer 도구를 사용하여 웹 서비스 바인딩 파일 및 웹 서비스 설명 또는 언어 구조를 작성합니다. 이 도구에 대한 자세한 정보는 Rational Developer for System z의 내용을 참조하십시오.
- 제공된 API를 사용하여 웹 서비스 지원을 호출하기 위한 자체 Java 프로그램을 작성합니다. 이 API는 웹 서비스 지원: 클래스 참조 Javadoc에 설명되어 있습니다. 여기에는 클래스를 설명하는 주석이 포함되고 웹 서비스 지원을 호출하는 방식에 대한 예제를 제공하기 위해 샘플 코드가 제공됩니다. Javadoc에는 필수인 JAR 파일 및 z/OS UNIX에서의 그 위치에 대한 전체 목록도 포함됩니다.

z/OS, Windows 또는 Linux 플랫폼에서 Java 프로그램을 실행할 수 있습니다. Windows 또는 Linux에서 프로그램을 실행하는 경우 FTP 또는 동등한 프로세스를 사용하여 생성된 웹 서비스 바인딩 파일을 2진 모드로 적절한 픽업 디렉토리에 전송하십시오.

2. 옵션: 언어 구조에서 웹 서비스 설명을 생성하는 경우 파일을 검토하고 필요한 사용자 정의를 수행하십시오. 자세한 정보는 159 페이지의 『생성된 웹 서비스 설명 문서 사용자 정의』의 내용을 참조하십시오.
3. 생성된 웹 서비스 바인딩 파일을 적절한 파이프라인 픽업 디렉토리에 배치하십시오.
4. 옵션: 예상대로 작동하는지 확인하기 위해 웹 서비스의 유효성 검증을 수행할 수 있도록 웹 서비스 설명을 파이프라인의 픽업 디렉토리에 복사하십시오.
5. 웹 서비스 설명을 시작한 경우 생성된 언어 구조와 인터페이스할 서비스 제공자 또는 서비스 요청자 애플리케이션 프로그램을 작성하십시오.
6. 필수 CICS 자원이 자동으로 작성되게 하려면 **PIPELINE SCAN** 명령을 실행하십시오.

자체 XML 인식 웹 서비스 애플리케이션 작성

CICS 제공 데이터 맵핑을 사용하지 않기로 선택한 경우 대신 두 가지 방식으로 자체 XML 인식 데이터 애플리케이션을 작성할 수 있습니다. DFHWS2LS에서 **XML-ONLY** 매개변수를 사용하거나 어떠한 도구도 사용하지 않고 자체 애플리케이션을 작성할 수

있습니다. **XML-ONLY** 매개변수를 사용하는 것은 처리할 XML 데이터를 애플리케이션에 전달하도록 CICS 파이프라인 프로세스를 구성하는 가장 간단한 방식입니다.

이 태스크 정보

자체 XML 인식 애플리케이션을 작성하는 작업에는 XML 문서를 구문 분석하고 생성하기 위한 코드 작성이 포함됩니다. 자체 XML 인식 애플리케이션을 작성하는 한 가지 방식은 COBOL에서 XML PARSE 및 XML GENERATE 명령문을 사용하는 것입니다. 자체 XML 인식 애플리케이션을 작성하는 다른 방식은 다른 IBM 도구를 사용하는 것입니다. 예를 들어, 애플리케이션에서 호출할 수 있는 COBOL XML 변환기 프로그램을 생성하기 위한 도구로 z Systems용 Rational Developer을 사용할 수 있습니다.

XML 인식 서비스 제공자 애플리케이션 작성:

XML 인식 서비스 제공자 애플리케이션은 이 애플리케이션에 전달되는 컨테이너에서 작동하고 XML과 프로그램 언어 간 데이터 변환을 처리해야 합니다.

이 태스크 정보

다음 단계는 CICS 도구의 사용에 대한 의사결정을 포함하여 XML 인식 애플리케이션의 작성 과정을 안내합니다.

프로시저

1. DFHWS2LS를 사용하여 XML 인식 애플리케이션의 웹 서비스 바인딩 파일을 생성할지 여부를 결정하십시오. 웹 서비스 바인딩 파일 생성의 장점은 유효성 검증과 같은 CICS 서비스를 사용하여 웹 서비스 및 글로벌 사용자 종료를 사용하는 CICS 모니터링을 테스트할 수 있다는 것입니다.
 - 웹 서비스 바인딩 파일을 생성하려면 **XML-ONLY** 매개변수를 지정하고 2.1 이상의 **MINIMUM-RUNTIME-LEVEL**을 지정하여 DFHWS2LS를 실행하십시오. 웹 서비스 바인딩 파일을 사용하면 애플리케이션 프로그램이 DFHWS-BODY 컨테이너의 콘텐츠에 대해 직접 작업할 수 있습니다. 모든 면에서 생성된 바인딩 파일은 **XML-ONLY** 매개변수 없이 생성된 파일과 같은 배치 특성 및 같은 런타임 동작(예: SOAP 메시지 핸들링 중 XML의 구문 분석)을 공유합니다. 이러한 구문 분석을 방지하려면 파이프라인 구성 파일에서 SOAP 메시지 핸들러를 지정하지 말아야 합니다.
 - 웹 서비스 바인딩 파일을 사용하지 않으려면 웹 서비스 요청이 XML 인식 애플리케이션에 도달하도록 서비스 제공자 파이프라인을 구성하십시오. 파이프라인 구성 파일에서 터미널 핸들러를 구성하여 XML 인식 애플리케이션 프로그램을 사용하거나 파이프라인에서 수신되는 URI에 따라 애플리케이션을 동적으로 전환하는 메시지 핸들러를 작성할 수 있습니다.
2. 다음 컨테이너에 보관되는 웹 서비스 요청을 처리할 애플리케이션을 작성하십시오.

DFHWS-BODY

파이프라인에 CICS 제공 SOAP 메시지 핸들러가 포함된 경우 인바운드 SOAP 요청에 대한 SOAP 본문의 콘텐츠입니다.

DFHREQUEST

파이프라인에서 수신한 SOAP 요청의 엔벨로프를 포함하는 전체 요청입니다.

DFHWS-XMLNS

네임스페이스 접두부를 요청의 XML 콘텐츠에 대한 네임스페이스로 맵핑하는 이름 값 쌍의 목록입니다.

DFHWS-SOAPACTION

컨테이너 DFHWS-BODY에서 SOAP 메시지와 연관된 SOAPAction 헤더입니다.

컨테이너에 대한 작업을 위해 API 명령을 코드화할 때 CHANNEL 옵션을 지정하지 마십시오. 모든 컨테이너가 현재 채널(프로그램에 전달된 채널)과 연관되기 때문입니다. 채널의 이름을 알아야 하는 경우, **EXEC CICS ASSIGN CHANNEL** 명령을 사용하십시오.

3. 옵션: 애플리케이션은 메시지 핸들러가 처리의 일부로 작성하는 다른 컨테이너 외에도 파이프라인에서 메시지 핸들러에 사용 가능한 추가 컨테이너를 사용할 수도 있습니다. 컨테이너의 전체 목록은 345 페이지의 『파이프라인에서 사용되는 컨테이너』의 내용을 참조하십시오.
4. 애플리케이션이 요청 처리를 완료한 후에는 다음 컨테이너를 사용하여 웹 서비스 응답을 구성하십시오.

DFHRESPONSE

파이프라인에 전달될 전체 응답 메시지입니다. 메시지에 SOAP을 사용하지 않거나 프로그램에서 엔벨로프를 포함하는 전체 SOAP 메시지를 빌드하려는 경우에 CICS 제공 SOAP 메시지 핸들러를 사용하는 대신 이 컨테이너를 사용하십시오.

컨테이너 DFHWS-BODY에 SOAP 본문을 제공하면 DFHRESPONSE는 무시됩니다.

DFHWS-BODY

아웃바운드 SOAP 응답에서 SOAP 본문의 콘텐츠입니다. 파이프라인의 터미널 핸들러가 CICS 제공 SOAP 메시지 핸들러인 경우 이 컨테이너를 제공하십시오. 메시지 핸들러는 본문을 포함한 전체 SOAP 메시지를 구성합니다.

요청 및 응답이 동일하더라도 프로그램은 이 컨테이너를 작성해야 합니다. 이 컨테이너를 작성하지 않으면 CICS가 내부 서버 오류를 발행합니다.

다른 컨테이너를 사용하여 아웃바운드 응답을 처리하기 위해 파이프라인이 필요로 하는 정보를 전달할 수도 있습니다.

웹 서비스가 응답을 리턴하지 않는 경우 컨테이너 DFHNORESPONSE를 리턴하여 응답이 없음을 표시해야 합니다. 메시지 핸들러는 컨테이너가 있는지 여부만 확인하기 때문에 컨테이너의 콘텐츠는 중요하지 않습니다.

5. URIMAP 자원을 작성하십시오. **XML-ONLY** 매개변수를 사용하고 있고 DFHWS2LS의 **URI** 매개변수에 값을 지정한 경우 PIPELINE SCAN 프로세스 중에 URIMAP이 자동으로 작성됩니다.

XML 인식 서비스 요청자 애플리케이션 작성:

XML 인식 웹 서비스 요청자 애플리케이션은 XML과 프로그래밍 언어 사이의 데이터 변환을 처리하고 파이프라인에서 제어 컨테이너를 채웁니다.

시작하기 전에

DFHWS2LS에서 **XML-ONLY** 매개변수를 사용하거나 어떠한 도구도 사용하지 않고 자체 XML 인식 서비스 요청자 애플리케이션을 작성할 수 있습니다. 자체 XML 인식 서비스 요청자 애플리케이션을 작성하는 가장 단순한 방법은 DFHWS2LS에서 **XML-ONLY** 매개변수를 사용하여 작성하는 것입니다. **XML-ONLY** 매개변수는 런타임 레벨 2.1 이상에서 사용 가능합니다.

이 태스크 정보

XML-ONLY 매개변수를 사용하면 CICS에 애플리케이션이 DFHWS-BODY 컨테이너의 콘텐츠를 사용하여 직접 작업하도록 지시하는 WSBind 파일이 생성됩니다. 생성된 WSBind 파일은 요청자 모드 PIPELINE에 설치되어 요청자 모드 WEBSERVICE 자원을 작성해야 합니다. 애플리케이션은 웹 서비스 요청의 본문에 대해 XML을 생성하고 생성된 XML을 DFHWS-BODY 컨테이너에 저장해야 합니다. 그런 다음 **EXEC CICS INVOKE SERVICE** 명령을 호출해야 합니다. 웹 서비스 제공자에게 아웃바운드 메시지가 전송됩니다. 호출 완료 후 응답 메시지의 본문은 DFHWS-BODY 컨테이너에도 있습니다.

SOAP 메시지 핸들링 중 응답 메시지의 XML이 구문 분석됩니다. 이러한 구문 분석을 방지하려면 파이프라인 구성 파일에서 SOAP 메시지 핸들러를 지정하지 말아야 합니다.

XML 인식 요청자 애플리케이션은 원격 제공자 모드 애플리케이션으로부터 SOAP 결합 메시지를 다시 수신할 수 있습니다. 이 경우, 요청자 애플리케이션은 SOAP 결합 해석 및 이 결합을 일반적인 응답 메시지와 구별하는 작업을 담당합니다. **INVOKE SERVICE** 명령이 **XML-ONLY WEBSERVICE**와 함께 사용되는 경우 CICS는 일반적으로 SOAP 결합을 수신했음을 나타내는 데 사용되는 응답 코드를 설정하지 않습니다.

XML-ONLY 옵션을 사용하지 않고 자체 XML 인식 서비스 요청자 애플리케이션을 작성하는 경우 다음 단계를 완료하십시오.

프로시저

1. 채널을 작성하고 이를 컨테이너로 채우십시오. 제어 컨테이너는 모두 **CHAR** 모드로 채워져야 합니다. 각 컨테이너에 다음 정보를 제공하십시오.

DFHWS-PIPELINE

아웃바운드 요청에 사용되는 **PIPELINE** 자원의 이름입니다.

DFHWS-URI

대상 웹 서비스의 **URI**입니다.

DFHWS-BODY

아웃바운드 **SOAP** 요청에서 **SOAP** 본문의 콘텐츠입니다. 파이프라인에 **CICS** 제공 **SOAP** 메시지 핸들러가 포함되는 경우 이 컨테이너를 제공하십시오. 메시지 핸들러는 본문을 포함한 전체 **SOAP** 메시지를 구성합니다.

DFHREQUEST

파이프라인에 전달될 전체 요청 메시지입니다. 메시지에 **SOAP**을 사용하지 않거나 프로그램에서 엔벨로프를 포함하는 전체 **SOAP** 메시지를 빌드하려는 경우에 이 컨테이너를 사용하십시오. 파이프라인은 아웃바운드 메시지로 중복 **SOAP** 헤더가 전송되는 것을 피하기 위해 **CICS** 제공 **SOAP** 메시지 핸들러를 포함하지 않아야 합니다.

컨테이너 **DFHWS-BODY**에 **SOAP** 본문이 제공되는 경우 **DFHREQUEST**는 비어 있어야 합니다. **DFHWS-BODY** 및 **DFHREQUEST** 둘 다에 콘텐츠를 제공하는 경우 **CICS**는 **DFHREQUEST**를 사용합니다.

DFHWS-XMLNS

네임스페이스 접두부를 요청의 XML 콘텐츠에 대한 네임스페이스로 맵핑하는 이름 값 쌍의 목록입니다.

DFHWS-SOAPACTION

컨테이너 **DFHWS-BODY**에서 지정된 **SOAP** 메시지에 추가할 **SOAPAction** 헤더입니다.

팁: 채널에 컨테이너 **DFHWS-NOABEND**를 추가하는 경우 **DFHPIRT**가 호출되면 **DFHPIRT** 내에서 이상 종료가 발생되지 않습니다. 이러한 작업은 **C/C++** 프로그램을 실행 중인 경우에 유용합니다. **DFHERROR** 컨테이너를 통해 오류를 처리할 수 있기 때문입니다.

2. 프로그램 **DFHPIRT**에 대한 링크입니다. 다음 명령을 사용하십시오.

```
EXEC CICS LINK PROGRAM(DFHPIRT) CHANNEL(userchannel)
```

여기서, *userchannel*은 컨테이너를 보관하는 채널입니다. 아웃바운드 메시지는 파이프라인에서 메시지 핸들러 및 헤더 처리 프로그램에 의해 처리되고 웹 서비스 제공자에게 전송됩니다.

3. 같은 채널에서 웹 서비스 응답을 포함하는 컨테이너를 검색하십시오. 웹 서비스 제공자의 응답은 성공적인 응답이거나 SOAP 결함일 수 있습니다. 웹 서비스 요청자 애플리케이션은 서비스 제공자의 두 응답 유형 모두 처리할 수 있어야 합니다. 전체 응답은 다음 컨테이너에 포함됩니다.

DFHRESPONSE

웹 서비스 제공자에서 수신한 SOAP 응답의 엔벨로프를 포함하는 전체 응답입니다.

DFHWS-BODY

파이프라인에 CICS 제공 SOAP 메시지 핸들러가 포함될 때 SOAP 본문
의 콘텐츠입니다.

DFHERROR

파이프라인의 오류 정보입니다.

참고: 일부 오류 케이스에서 DFHWS-BODY가 갱신되지 않을 수 있습니다. SOAP 결함에 대해 DFHRESPONSE를 확인해야 합니다.

웹 서비스에 Java 사용

Java를 사용하여 웹 서비스 애플리케이션을 작성할 수 있습니다. 이러한 애플리케이션을 작성하는 데에는 다른 프로그래밍 언어가 사용된 기법과 비교할 때 다른 기법이 사용됩니다.

대부분의 비Java 프로그래밍 언어의 경우 웹 서비스 지원을 사용하여 애플리케이션을 사용으로 설정합니다. 웹 서비스 지원 사용은 CICS가 데이터를 웹 서비스에서 애플리케이션에 적합한 양식으로 변환하고 이를 컨테이너 또는 COMMAREA에 배치한다는 것을 의미합니다. Java 애플리케이션과 함께 웹 서비스 지원을 사용할 수 있지만 다음 태스크는 Java 애플리케이션용 Java 웹 서비스 작성에 보다 적합한 메소드를 제공합니다.

Java 제공자 모드 웹 서비스를 Axis2 JVM 서버에 배치:

CICS에서 Axis2 애플리케이션을 제공자 모드 웹 서비스로 배치할 수 있습니다. 이러한 애플리케이션은 일반적으로 JAX-WS를 사용하여 생성되며 Java가 사용 설정된 파이프라인에서 호스팅할 수 있습니다.

다음 이유 중 하나로 이 메소드를 사용하여 Java 애플리케이션을 배치하려고 할 수 있습니다.

- Axis2 핸들러 인터페이스를 사용하는 기존 투자가 있습니다.
- CICS 파이프라인 구성을 사용하려고 합니다.

참고: Axis2 스타일 애플리케이션은 WEBSERVICE 자원을 사용하지 않습니다. 이러한 애플리케이션은 Axis2 프로그래밍 모델을 사용하여 CICS와 상호작용하기 때문에 일부 CICS 웹 서비스 지원을 사용할 수 없습니다. 다음 서비스는 Axis2 스타일 애플리케이션에서 완전히 지원되지는 않습니다.

- SOAPFAULT CREATE
- WSACONTEXT GET
- 360 페이지의 『DFHWS-OPERATION 컨테이너』
- 359 페이지의 『DFHWS-MEP 컨테이너』
- 366 페이지의 『DFHWS-USERID 컨테이너』
- 361 페이지의 『DFHWS-TRANID 컨테이너』
- 웹 서비스 보안

시작하기 전에

Axis2에 배치하기에 적합한 Java 애플리케이션(예: JAX-WS를 사용하는 POJO 애플리케이션)이 있어야 합니다. 이 태스크에서 다음 POJO 애플리케이션이 예제로 사용됩니다.

```
/**
 * Simple example
 */
@javax.jws.WebService(targetNamespace = "com.ibm.cics.example", name = "pojoExample")
public class TestAxis2
{
    public String getMessage(String input)
    {
        return "CICS got this: '" + input + "'";
    }
}
```

이 애플리케이션은 WSDL을 생성하는 데 사용되는 XML 네임스페이스 및 웹 서비스와 연관시킬 이름을 지정합니다.

이 애플리케이션의 Java 코드를 컴파일해야 하고 JAX-WS 생성기를 실행하여 애플리케이션을 TestAxis2.jar이라고 하는 jar 파일로 패키징합니다. 이러한 작업은 다음 코드를 발행하여 수행할 수 있습니다.

```
javac TestAxis2.java
wsngen -cp . TestAxis2 -wsdl
jar -cvf TestAxis2.jar *
```

JAX-WS 생성기는 WSDL 문서 및 Axis2에서 사용되는 바인딩도 작성합니다.

이 태스크 정보

Axis2 웹 서비스를 배치하려면 웹 서비스를 위한 파이프라인 인프라를 작성해야 합니다. 파이프라인을 작성한 경우 웹 서비스를 작성할 수 있습니다. 필요한 수의 웹 서비

스에 대해 작성된 파이프라인을 재사용할 수 있습니다. 다음 단계는 파이프라인 및 웹 서비스를 작성하는 방법에 대해 설명합니다.

참고: WEBSERVICE 자원은 이 태스크의 일부로 작성되거나 설치되지 않습니다.

프로시저

1. 파이프라인 인프라를 작성하십시오.
 - a. Java 파이프라인에 대한 웹 서비스 인프라를 작성하십시오. 자세한 정보는 257 페이지의 『SOAP 서비스 제공자를 위한 CICS 인프라 작성』의 내용을 참조하십시오.
 - b. Axis2 저장소를 작성하십시오. 이를 수행하려면 `$CICS_HOME/lib/pipeline/repository`에 있는 제공된 저장소의 사본을 작성하십시오.
 - c. 파이프라인 구성 파일에 `<repository>` 요소를 추가하십시오. 이 요소는 사용자가 작성한 Axis2 저장소의 이름을 지정해야 합니다.
 - d. PIPELINE 자원을 작성하고 사용으로 설정하십시오.
2. 파이프라인과 연관된 각 웹 서비스에 대해 다음 단계를 반복하여 웹 서비스를 작성하십시오.
 - a. Axis2 애플리케이션을 Axis2 저장소에 배치하십시오. 예를 들어, 예제에서 작성된 jar 파일은 저장소 디렉토리에 있는 `servicejars`라고 하는 디렉토리에 배치되어야 합니다. 이 디렉토리가 없는 경우에는 작성해야 합니다.
 - b. 웹 서비스를 위한 URIMAP 자원을 정의하고 설치하십시오. URIMAP 자원은 웹 서비스와 연관된 URI 및 PIPELINE 자원을 지정해야 합니다. URI는 URI에 대한 Axis2 이름 지정 규칙을 따라야 합니다. 기본 Axis2 이름 지정 규칙은 `/name_of_service서비스.name_of_portPort/suffix`입니다. 여기서, `name_of_service`는 WSDL에서 웹 서비스의 이름이고 `name_of_port`는 WSDL에서 포트의 이름이며 `suffix`는 사용자가 정의할 수 있는 선택적 접미부입니다. 이전 예제에서 다음 URIMAP 자원을 사용할 수 있습니다.

Urimap	: EXAMPLE
Group	: EXAMPLE
Status	: Enabled
USAge	: Pipeline
SCHEME	: HTTP
Port	: No
HOST	: *
Path	: /TestAxis2Service.pojoExamplePort/example/TestAxis2
Transaction	: CPIH
Pipeline	: EXAMPLE

이 예제에서는 사용되는 PIPELINE 자원을 EXAMPLE이라고 하는 것으로 가정합니다.

다음에 수행할 작업

웹 서비스가 올바르게 실행되는지 테스트하십시오.

XML을 생성하고 구문 분석하는 Java 웹 서비스 작성:

XML을 스스로 구문 분석하고 생성하는 Java 애플리케이션을 작성할 수 있습니다. 이러한 애플리케이션은 다른 프로그래밍 언어로 작성된 XML 인식 애플리케이션과 일치하지만 XML 처리에 표준 Java 기술을 사용한다는 이점이 있습니다.

프로시저

1. XML-ONLY WEBSERVICE 자원을 작성하십시오. 자세한 정보는 170 페이지의 『XML 인식 서비스 요청자 애플리케이션 작성』 또는 168 페이지의 『XML 인식 서비스 제공자 애플리케이션 작성』의 내용을 참조하십시오.
2. SOAP 메시지의 본문에 대해 XML을 구문 분석하고 생성할 수 있는 Java 웹 서비스를 작성하십시오. 다음과 같은 기능이 있는 Java 웹 서비스를 작성하는 데 도움이 되는 다양한 도구(예: Java 6 JAXB(Java Architecture for XML Binding) 라이브러리)를 사용할 수 있습니다.
3. 옵션: 제공자 파이프라인을 사용하고 있고 SOAP 결함 메시지가 요청자에게 리턴 되도록 하는 기능을 추가하려는 경우 JCICS SoapFault 클래스를 사용하여 **EXEC CICS SOAPFAULT CREATE** 명령을 발행하십시오.
4. 옵션: 요청자 파이프라인을 사용하고 있는 경우 JCICS 서비스 클래스를 사용하여 **EXEC CICS INVOKE SERVICE** 명령을 발행하십시오.

COBOL 인터페이스가 있는 Java 웹 서비스 작성:

다른 프로그래밍 언어에서 사용된 것과 같은 기법을 사용하여 CICS와 상호작용하는 Java 애플리케이션을 작성할 수 있습니다. 이러한 애플리케이션을 작성하려면 Java 코드를 작성하거나 생성하여 구조화된 COMMERA- 또는 container- 스타일 데이터를 작성해야 합니다.

프로시저

1. DFHWS2LS를 사용하여 웹 서비스를 위한 COBOL 언어 구조를 작성하십시오.
2. COBOL 언어 구조를 생성하고 구문 분석하는 Java 웹 서비스를 작성하십시오. Java 프로그램이 기존 CICS 애플리케이션 데이터에 액세스하기 위해 사용하는 도구에 대한 자세한 정보 및 COBOL 언어 구조를 생성하고 구문 분석할 수 있는 Java 웹 서비스 작성 방법 예제에 대한 링크는 애플리케이션 개발에서 Java의 구조 데이터와 상호 작용의 내용을 참조하십시오.
3. 옵션: 제공자 파이프라인을 사용하고 있고 SOAP 결함 메시지가 요청자에게 리턴 되도록 하는 기능을 추가하려는 경우 JCICS SoapFault 클래스를 사용하여 **EXEC CICS SOAPFAULT CREATE** 명령을 발행하십시오.

4. 옵션: 요청자 파이프라인을 사용하고 있는 경우 JCICS 서비스 클래스를 사용하여 CICS 서비스 API와 인터페이스하고 **EXEC CICS INVOKE SERVICE** 명령을 발행하십시오.

요청자 모드 JAX-WS 웹 서비스 배치:

CICS에서 JAX-WS 애플리케이션을 요청자 모드 웹 서비스로 배치할 수 있습니다. 그러나 이러한 애플리케이션은 **EXEC CICS INVOKE** 명령을 사용하지 않는 대신 JAX-WS를 사용하여 원격 웹 서비스와 상호작용합니다.

시작하기 전에

OSGi를 지원하도록 JVM 서버가 구성되어 있어야 합니다. 자세한 정보는 구성에서 JVM 서버 설정의 내용을 참조하십시오.

이 태스크 정보

JAX-WS 애플리케이션을 요청자 모드 웹 서비스로 배치하는 것의 장점은 zEnterprise Application Assist Processor(zAAP)를 사용하는 플랫폼에 의존하지 않는 웹 서비스 요청자 애플리케이션을 작성할 수 있다는 것입니다. zAAP를 사용하면 트랜잭션의 비용을 줄일 수 있습니다. 자세한 정보는 IBM Redbooks 서적: zAAP(zSeries Application Assist Processor) 구현을 참조하십시오.

프로시저

1. Java에서 웹 서비스 요청자 애플리케이션을 작성하고 적절한 API(예: JAX-WS(Java API for XML web services))를 사용하여 원격 웹 서비스를 호출하십시오.
2. 옵션: 원격 웹 서비스를 시작하는 데 JAX-WS를 사용하는 경우 SOAP 메시지 생성, 네트워크 통신 처리 및 SOAP 응답 처리에도 JAX-WS를 사용해야 합니다.
3. JVM 서버에서 Java 애플리케이션을 배치하고 설치하십시오.

다음에 수행할 작업

웹 서비스가 올바르게 시작되는지 테스트하십시오.

Liberty JVM 서버에서 Java 제공자 모드 웹 서비스 배치:

Liberty JVM 서버에서 웹 애플리케이션을 제공자 모드 웹 서비스로 배치할 수 있습니다. 이러한 애플리케이션은 Java 표준 JAX-WS 및 JAXB를 사용하여 작성됩니다.

이 태스크 정보

CICS TS V5.3에는 JAX-WS(Java API for XML Web Service) 및 JAXB(Java Architecture for XML Binding)에 대한 기능을 제공하는 최신 WebSphere Application Server Liberty Profile(WLP) V8.5.5가 포함됩니다. 이러한 기술을 함께 사용하면 Java에서 CICS 애플리케이션의 일부로 SOAP 웹 서비스를 작성할 수 있습니다. 다음 기사

에서는 Eclipse를 설정하고 샘플 웹 서비스 프로젝트를 테스트하며 CICS에 샘플을 배치하고 JCICS를 사용하도록 샘플을 수정하며 웹 서비스 탐색기에서 이 샘플을 테스트하는 방법을 보여줍니다. 세부사항은 DeveloperWorks, CICS TS V5.2 오픈 베타 Liberty 프로파일의 JAX-WS 및 JAXB 지원의 내용을 참조하십시오.

다음 이유 중 하나로 이 메소드를 사용하여 Java 애플리케이션을 배치하려고 할 수 있습니다.

- Java에서 웹 서비스를 작성하려고 합니다.
- CICS 웹 서비스 지원을 사용하여 처리하기 어려울 수 있는 복잡한 WSDL 문서가 있습니다.
- 웹 서비스 애플리케이션의 핸들링을 zEnterprise Application Assist Processor(zAAP)에 분담하려고 합니다.

참고: Liberty JVM 서버에 배치된 웹 애플리케이션은 WEBSERVICE 또는 TCPIPSERVICE 자원을 사용하지 않습니다. 이러한 애플리케이션은 Liberty HTTP 리스너를 사용하여 웹 요청과 상호작용하기 때문에 CICS 웹 서비스 지원의 기능을 사용할 수 없습니다.

SOAP 메시지 유효성 검증

CICS 웹 서비스를 사용할 때 SOAP 메시지가 웹 서비스 설명에 포함되어 있는 스키마를 준수하는지 확인하기 위해 SOAP 메시지를 유효성 검증하도록 지정할 수 있습니다. 제공자 및 요청자 모드 애플리케이션을 둘 다 유효성 검증할 수 있습니다.

시작하기 전에

웹 서비스 배치를 개발하고 테스트하는 동안 전체 유효성 검증은 서비스 요청자와 서비스 제공자 사이의 메시지 교환에 있어서 문제점을 발견할 수 있도록 지원합니다. 그러나 SOAP 메시지의 전체 유효성 검증에는 상당한 오버헤드가 따르고 완전히 테스트되는 프로덕션 애플리케이션에서 메시지를 유효성 검증하는 것은 바람직하지 않습니다.

CICS는 Java 프로그램을 사용하여 SOAP 메시지를 유효성 검증합니다. 따라서 SOAP 메시지를 유효성 검증하려면 Java 지원이 CICS 리전에서 사용으로 설정되어 있어야 합니다.

이 태스크 정보

SOAP 메시지는 애플리케이션 데이터 구조로 변환되기 전 및 SOAP 메시지가 애플리케이션 데이터 구조에서 생성될 때 유효성 검증됩니다. SOAP 메시지는 WSDL에서 XML 스키마를 사용하여 유효성 검증되고 CICS의 변환 요구사항에 대해 다시 유효성 검증됩니다. WEBSERVICE 자원의 **WSDLFILE** 속성에서 지정된 WSDL 파일을 사용하거나 WEBSERVICE 자원의 **ARCHIVEFILE** 속성에서 지정된 .zip 파일에 포함되어

있는 WSDL 파일을 사용할 수 있습니다. 두 속성 모두 지정된 경우 **ARCHIVEFILE** 속성에서 지정된 아카이브 파일의 WSDL 파일이 사용됩니다.

유효성 검증이 꺼진 경우 CICS는 Java 프로그램을 사용하지 않습니다. CICS는 SOAP 메시지에 잘 구성된 XML이 포함되어 있는지 확인하고 변환해야 하는 정도로만 SOAP 메시지를 유효성 검증합니다. 따라서 SOAP 메시지는 성공적으로 유효성 검증되지만 런타임 환경에서 실패할 수 있고 그 반대의 경우도 발생할 수 있습니다.

프로시저

1. 옵션: CICS 리전에서 JVM 서버를 설정하십시오. OSGi 프레임워크 또는 Axis2에서 SOAP 유효성 검증을 실행할 수 있지만 Liberty 프로파일에서는 실행할 수 없습니다. CICS는 OSGi 프레임워크를 사용하는 JVM 서버를 빠르게 설정할 수 있도록 샘플을 제공합니다. Java 파이프라인을 사용하는 경우 추가 JVM 서버를 정의하지 않아도 됩니다.

- 유효성 검증에 사용할 JVM 서버가 이미 정의되어 있는 경우 JVMSERVER 자원의 이름을 참조하도록 그룹 DFHPIVAL에서 DFHPIVAL 프로그램 정의를 수정하십시오. DFHPIVAL 정의는 잠그지 않으므로 편집할 수 있습니다. 기본적으로 이 정의는 DFHJVMs를 참조합니다.

또는

- CICS 제공 샘플을 사용하려는 경우 샘플 JVM 서버 DFHJVMs를 그룹 DFH\$OSGI에 설치해야 합니다. 자세한 정보는 구성에서 JVM 서버 설정의 내용을 참조하십시오.

2. WEBSERVICE 자원과 연관된 웹 서비스 설명이 있는지 확인하십시오. 이 연관은 WSDL 파일 또는 한 개 이상의 WSDL 파일을 포함하는 .zip 파일이 파이프라인 스캔 중 파이프라인의 픽업 디렉토리에 있는 경우 자동으로 작성되는 WEBSERVICE 자원에 대해 작성됩니다.

RDO를 사용하여 작성되는 WEBSERVICE 정의의 경우 웹 서비스 설명은 WSDLFILE 속성에 지정됩니다.

3. WEBSERVICE 자원의 **VALIDATION=YES** 속성을 지정하여 웹 서비스 유효성 검증을 켜십시오. 자원을 정의할 때 유효성 검증이 필요한지 여부를 지정할 수 있고 자원이 설치된 후에는 이 설정을 변경할 수 있습니다.

결과

SOAP 메시지가 유효한지 여부를 알아보려면 시스템 로그를 확인하십시오. 메시지 DFHPI1002는 SOAP 메시지가 성공적으로 유효성 검증되었음을 나타내고 메시지 DFHPI1001은 유효성 검증에 실패했음을 나타냅니다.

다음에 수행할 작업

유효성 검증이 더 이상 필요하지 않으면 끄십시오.

웹 서비스의 유효성 검증 상태 변경

다음 인터페이스를 사용하여 웹 서비스의 유효성 검증 상태를 변경할 수 있습니다.



CICS Explorer



CICS Explorer 관리 보기

웹 서비스 보기의 유효성 검증 상태 속성 사용

CICSplex SM

WEBSERVICE 정의 보기

CEMT



SET WEBSERVICE 명령

CICS SPI



SET WEBSERVICE 명령

JSON 웹 서비스 작성

기존 CICS 애플리케이션을 JSON 웹 서비스로 노출하여 SOAP 웹 서비스 제공자 역할을 할 새 CICS 애플리케이션을 작성할 수 있습니다.

시작하기 전에

JSON 웹 서비스 작성을 시작하기 전에 JSON 웹 서비스를 지원하도록 CICS 시스템을 구성해야 합니다. 자세한 정보는 261 페이지의 『JSON 서비스 제공자를 위한 CICS 인프라 작성』의 내용을 참조하십시오.

이 태스크 정보

CICS JSON 지원은 사용자가 새 JSON 웹 서비스 제공자 애플리케이션에 필요한 아티팩트를 작성하거나 기존 애플리케이션을 JSON 웹 서비스 제공자로 사용할 수 있도록 돕는 제공 유틸리티입니다.

CICS JSON 지원은 고급 언어 구조에서 JSON 스키마를 작성하거나 기존 JSON 스키마에서 고급 언어 구조를 작성할 수 있으며 COBOL, C/C++ 및 PL/I를 지원합니다. 또한 JSON 메시지를 컨테이너 및 COMMAREA로 또는 그 반대로 자동 런타임 변환

을 가능하게 하기 위해 사용되는 정보를 생성합니다. 이 정보는 파이프라인 처리 중 CICS JSON 웹 서비스 지원에서 사용됩니다.

다음 프로시저에서 설명하는 대로 JSON 웹 서비스를 작성하고 작성한 웹 서비스가 올바르게 작동하는지 유효성 검증하십시오.

프로시저

1. JSON 웹 서비스를 작성하십시오. JSON 지원을 사용하여 JSON 스키마 또는 언어 구조를 작성하고 작성한 항목을 CICS에 배치합니다. **PIPELINE SCAN** 명령을 사용하여 필수 CICS 지원을 자동 작성합니다.
2. JSON 웹 서비스를 시작하여 웹 서비스가 의도한 대로 작동하는지 테스트하십시오.

다음에 수행할 작업

이러한 단계는 다음 주제에서 보다 자세히 설명됩니다.

CICS JSON 지원

CICS JSON 지원은 JSON 스키마와 언어 구조 사이에 맵핑을 작성하는 일괄처리 유틸리티 세트입니다. 이 맵핑은 JSON과 애플리케이션 데이터 사이의 변환을 수행하기 위해 런타임 시 CICS에 의해 사용됩니다. 지원은 최소 프로그래밍 노력으로 서비스 제공자 및 서비스 요청자에서 사용하기 위한 CICS 애플리케이션의 빠른 배치를 지원합니다.

CICS에 JSON 지원을 사용할 때, 인바운드 메시지를 구문 분석하고 아웃바운드 메시지를 구성하기 위한 사용자 자체 코드를 작성할 필요가 없습니다. CICS는 JSON 메시지와 애플리케이션 프로그램의 데이터 구조 사이의 데이터를 맵핑합니다.

지원은 고급 언어 구조에서 JSON 스키마를 작성하거나 기존 JSON 스키마에서 고급 언어 구조를 작성할 수 있고, COBOL, C/C++ 및 PL/I를 지원합니다. 이는 컨테이너 및 COMMAREA에 대한 JSON 메시지의 자동화 런타임 변환을 가능하게 하거나 또는 그 반대에 사용되는 정보를 생성합니다.

CICS JSON 지원은 두 개의 유틸리티 프로그램을 포함합니다.

DFHLS2JS

언어 구조에서 웹 서비스 바인딩 파일을 생성합니다. 이 유틸리티는 JSON 스키마도 생성합니다.

DFHJS2LS

JSON 스키마에서 웹 서비스 바인딩 파일을 생성합니다. 이 유틸리티는 애플리케이션 프로그램에서 사용할 수 있는 언어 구조도 생성합니다.

양쪽 모두의 프로그램을 실행하기 위한 JCL 프로시저는 *hlq.XDFHINST* 라이브러리에 있습니다.

JSON 지원 유틸리티 프로그램 및 데이터 맵핑에 대한 자세한 정보는 다음 주제를 참조하십시오.

DFHLS2JS: 요청-응답 서비스를 위한 고급 언어 대 JSON 스키마 변환:

DFHLS2JS 프로시저는 고급 언어 데이터 구조에서 JSON 스키마 파일을 생성합니다. DFHLS2JS는 CICS 애플리케이션 프로그램을 서비스 제공자로 표시할 때 사용할 수 있습니다.

DFHLS2JS와 기호 매개변수, 입력 매개변수와 설명, 작업 예제에 대한 작업 제어 명령문은 이 프로시저를 사용하는 데 도움을 줍니다.

DFHLS2JS JCL 프로시저는 데이터 세트 *HLQ.XDFHINST*에 설치됩니다. 여기서 *HLQ*는 CICS가 설치된 상위 레벨 규정자입니다.

DFHLS2JS 또는 DFHJS2LS 프로시저의 관련 사용 모드는 다음과 같이 요구사항에 따라 다릅니다.

- DFHLS2JS: 링크 가능 인터페이스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 링크 가능 인터페이스를 위한 JSON 스키마 대 고급 언어 변환
- DFHLS2JS: 요청-응답 서비스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 요청-응답 서비스를 위한 JSON 스키마 대 고급 언어 변환
- DFHJS2LS: RESTful 서비스를 위한 JSON 스키마 대 고급 언어 변환

DFHLS2JS의 작업 제어 명령문

JOB 작업을 시작합니다.

EXEC

프로시저 이름(DFHLS2JS)을 지정합니다.

INPUT.SYSUT1 DD

입력을 지정합니다. 입력 매개변수는 일반적으로 입력 스트림에 지정됩니다. 그러나 데이터 세트 또는 파티션된 데이터 세트의 멤버에 정의될 수 있습니다.

기호 매개변수

DFHLS2JS에서는 다음 기호 매개변수가 정의됩니다.

JAVADIR=path

DFHLS2JS에서 사용되는 Java 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 */usr/lpp/*에 추가되어 */usr/lpp/path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

PATHPREF=prefix

다른 매개변수 또는 ''(빈 문자열)(접두부를 사용하지 않는 경우)에서 사용되는 z/OS UNIX 디렉토리 경로를 확장하는 접두부를 지정합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **PATHPREF** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

SERVICE=value

이 매개변수는 IBM 지원 센터에서 지시하는 경우에만 사용합니다.

TMPDIR=tmpdir

DFHLS2WS가 z/OS UNIX에서 임시 작업공간으로 사용하는 디렉토리의 위치를 지정합니다. 작업을 실행하는 데 사용되는 사용자 ID는 이 디렉토리에 대한 읽기 및 쓰기 권한이 있어야 합니다.

기본값은 /tmp입니다.

TMPFILE=tmpprefix

DFHLS2JS가 임시 작업공간 파일의 이름을 구성하기 위해 사용하는 접두부를 지정합니다.

기본값은 LS2JS입니다.

USSDIR=path

UNIX 시스템 서비스 파일 시스템에서 CICS TS 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/cicsts/에 추가되어 /usr/lpp/cicsts/path의 전체 경로 이름을 생성합니다. 기본값을 사용하는 경우 '.'(마침표)로 지정되어야 합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **USSDIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

임시 작업공간

DFHLS2JS는 런타임에 다음 세 개 임시 파일을 작성합니다.

tmpdir/tmpprefix.in

tmpdir/tmpprefix.out

tmpdir/tmpprefix.err

여기서,

*tmpdir*은 **TMPDIR** 매개변수에 지정된 값입니다.

*tmpprefix*는 **TMPFILE** 매개변수에 지정된 값입니다.

TMPDIR 및 **TMPFILE**이 지정되지 않은 경우 파일의 기본 이름은 다음과 같습니다.

/tmp/LS2JS.in

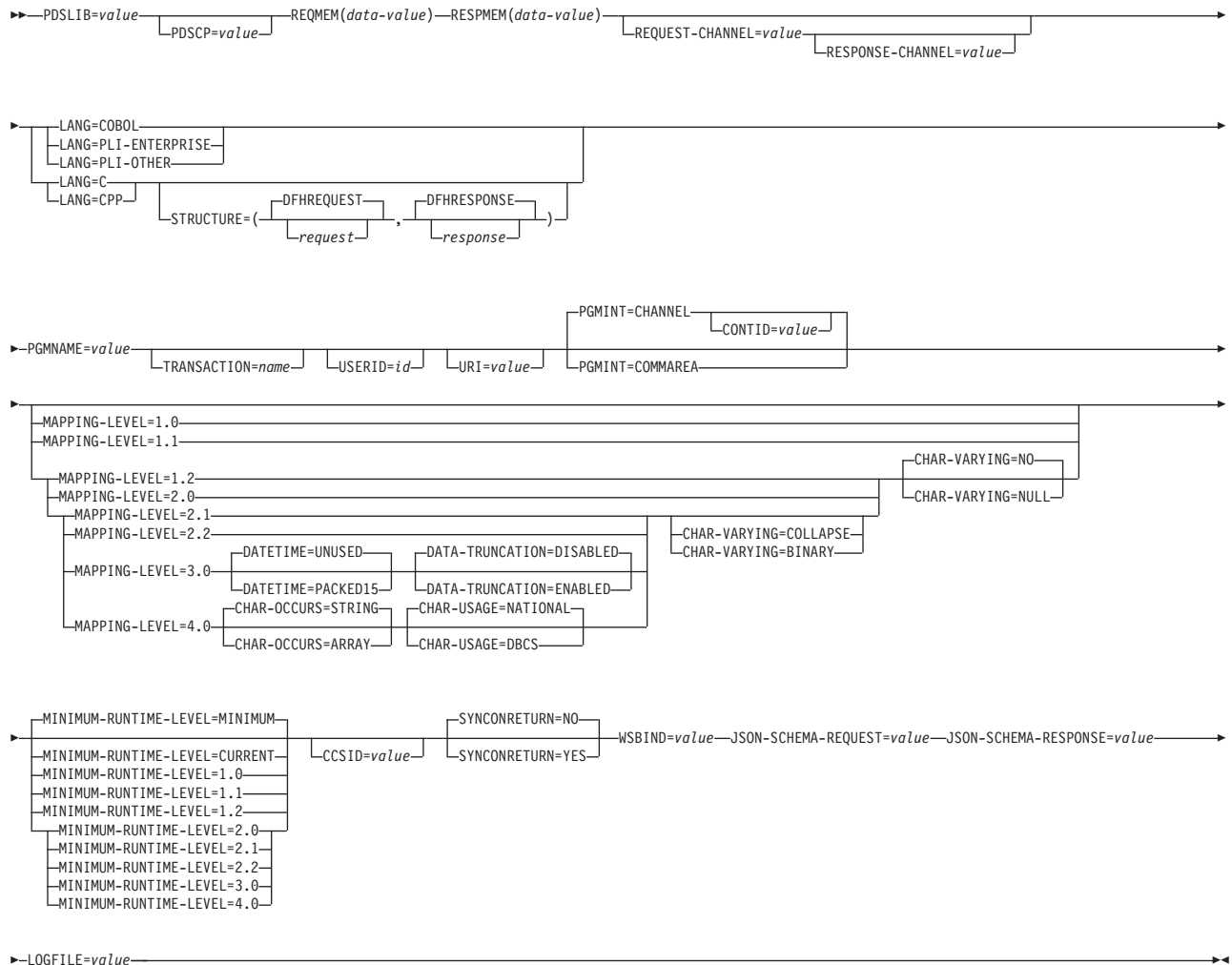
/tmp/LS2JS.out

/tmp/LS2JS.err

중요사항: DFHLS2JS는 z/OS UNIX 파일 또는 데이터 세트 멤버에 대한 액세스를 차단하지 않습니다. 따라서 두 개 이상의 DFHLS2JS 인스턴스가 동시에 실행되고 동일한 임시 작업공간 파일을 사용하는 경우, 다른 작업이 작업공간 파일을 사용하는 동안 특정 작업이 해당 파일을 겹쳐쓸 수 있어 예상치 못한 실패를 경험할 수 있습니다.

따라서 이러한 상황을 방지하는 이름 지정 규칙과 작동 프로시저를 고안해야 합니다. 예를 들어, 시스템 기호 매개변수 **SYSUID**를 사용하여 개별 사용자에게 고유한 작업공간 파일 이름을 생성할 수 있습니다. 이러한 임시 파일은 작업이 종료되기 전에 삭제됩니다.

DFHLS2JS의 입력 매개변수



매개변수 사용

- 입력 매개변수는 순서에 관계 없이 지정할 수 있습니다.
- 각 매개변수는 행을 바꿔 시작해야 합니다.
- 매개변수는 사용 시 해당 연속 문자와 함께 열 72를 넘어가서는 안됩니다.
열 73 - 80은 공백이어야 합니다.
- 매개변수가 너무 길어 한 행에 맞지 않는 경우에는 행 끝에서 별표(*) 문자를 사용하여 매개변수가 다음 행에서 연속됨을 표시해야 합니다. 별표 앞의 모든 내용은 공백을 포함하여 매개변수의 일부로 간주됩니다. 예를 들어, 다음과 같습니다.

```
WSBIND=wsbinddir*  
/app1
```

구문은 다음과 같습니다.

```
WSBIND=wsbinddir/app1
```

- 행에서 첫 번째 문자 위치의 # 문자는 주석 문자입니다. 이 행은 무시됩니다.
- 행에서 마지막 문자 위치의 쉼표는 선택적 섹션 구분 기호이며 무시됩니다.

매개변수 설명

CCSID=*value*

런타임에 애플리케이션 데이터 구조에서 문자 데이터를 인코딩하는 데 사용되는 CCSID를 지정합니다. 이 매개변수의 값은 **LOCALCCSID** 시스템 초기화 매개변수의 값을 대체합니다. *value*는 Java 및 z/OS Unicode Services User's Guide and Reference에서 지원하는 EBCDIC CCSID여야 합니다. 이 매개변수를 지정하지 않으면 시스템 초기화 매개변수에 지정된 CCSID를 사용하여 애플리케이션 데이터 구조가 인코딩됩니다.

CHAR-VARYING={NO|NULL|COLLAPSE|BINARY}

맵핑 레벨이 1.2 이상일 때 언어 구조의 문자 필드가 맵핑되는 방식을 지정합니다. COBOL의 문자 필드가 X 유형의 Picture 절인 경우(예: PIC(X) 10) C/C++의 문자 필드는 문자 배열입니다. 다음 옵션을 선택할 수 있습니다.

NO 문자 필드가 JSON 문자열에 맵핑되고 고정 길이 필드로 처리됩니다. 데이터의 최대 길이는 필드 길이와 같습니다. 맵핑 레벨이 2.0 이하인 COBOL 및 PL/I의 경우 **CHAR-VARYING** 매개변수의 기본값은 NO입니다.

이 값은 엔터프라이즈 및 기타 PL/I 언어 구조에는 적용되지 않습니다.

NULL

문자 필드가 JSON 문자열에 맵핑되고 널 종료 문자열로 처리됩니다. CICS는 JSON 메시지에서 변환될 때 종료 널 문자를 추가합니다. 문자열의 최대 길이는 언어 구조에 표시된 길이보다 한 문자 적은 길이로 계산됩니다. C/C++의 경우 **CHAR-VARYING** 매개변수의 기본값은 NULL입니다.

이 값은 엔터프라이즈 및 기타 PL/I 언어 구조에는 적용되지 않습니다.

COLLAPSE

문자 필드가 JSON 문자열에 맵핑됩니다. 필드에서 후미 공백은 JSON 메시지에 포함되지 않습니다. 인바운드 JSON 메시지는 선두, 후미, 임베디드 공백을 모두 제거하기 위해 구문 분석됩니다. 맵핑 레벨이 2.1 이상인 COBOL 및 PL/I의 경우 **CHAR-VARYING** 매개변수의 기본값은 COLLAPSE 입니다.

BINARY

문자 필드가 Base-64 인코딩 데이터를 포함하는 JSON 문자열에 맵핑되고 고정 길이 필드로 처리됩니다. **CHAR-VARYING** 매개변수의 BINARY 값은 맵핑 레벨이 2.1 이상인 경우에만 사용할 수 있습니다.

CHAR-OCCURS={STRING|ARRAY}

맵핑 레벨이 4.0 이상일 때 언어 구조의 문자 배열이 맵핑되는 방식을 지정합니다 (예: PIC X OCCURS 20). 이 매개변수는 COBOL 언어만 사용할 수 있습니다.

ARRAY

문자 배열이 JSON 배열에 맵핑됩니다. 이는 모든 문자가 개별 JSON 요소에 맵핑됨을 의미합니다. 맵핑 레벨이 3.0 이하인 경우의 동작이기도 합니다.

STRING

문자 배열이 JSON 문자열에 맵핑됩니다. 이는 전체 COBOL 배열이 단일 JSON 요소에 맵핑됨을 의미합니다.

CHAR-USAGE={NATIONAL|DBCS}

COBOL에서 국가적 데이터 유형인 PIC N은 UTF-16 또는 DBCS 데이터에 사용될 수 있습니다. 이 설정은 NSYMBOL 컴파일러 옵션에 의해 제어됩니다. 데이터가 적절하게 처리되는지 확인하려면 보조의 **CHAR-USAGE** 매개변수를 NSYMBOL 컴파일러 옵션과 동일한 값으로 설정해야 합니다. 이는 주로 UTF-16을 사용할 때 CHAR-USAGE=NATIONAL로 설정됩니다.

DBCS PIC (n) 필드의 데이터는 UTF-16 인코딩 데이터로 처리됩니다.

NATIONAL

PIC (n) 필드의 데이터는 DBCS 인코딩 데이터로 처리됩니다.

CONTID=value

서비스 제공자에서, JSON 메시지를 나타내는 데 사용되는 상위 레벨 데이터 구조를 보유하는 컨테이너의 이름을 지정합니다.

CICS가 대상 애플리케이션 프로그램으로 전달하는 컨테이너의 길이는 요청 컨테이너의 길이와 응답 컨테이너의 길이 중 긴 값입니다.

DATA-TRUNCATION={DISABLED|ENABLED}

가변 길이 데이터가 고정 길이 필드 구조에서 용인되는지 여부를 지정합니다.

DISABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 거부하고 오류 메시지를 발행합니다.

ENABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 용인하고 누락된 데이터를 널값으로 처리합니다.

DATETIME={UNUSED|PACKED15}

고급 언어 구조의 잠재적 ABTIME 필드가 시간소인으로 매핑되는지 여부를 지정합니다.

PACKED15

길이가 15(8바이트)인 팩형 10진수 필드가 CICS ABTIME 필드로 처리되고 시간소인으로 매핑됩니다.

UNUSED

길이가 15(8바이트)인 팩형 10진수 필드가 시간소인으로 처리되지 않습니다.

이 매개변수는 맵핑 레벨 3.0에서 설정할 수 있습니다.

JSON-SCHEMA-REQUEST=*value*

필수 매개변수입니다.

값은 요청 JSON 스키마가 저장되는 UNIX System Services 위치를 나타냅니다.

JSON-SCHEMA-RESPONSE=*value*

필수 매개변수입니다.

값은 응답 JSON 스키마가 저장되는 UNIX System Services 위치를 나타냅니다.

LANG=COBOL

고급 언어 구조의 프로그래밍 언어가 COBOL인 것으로 지정합니다.

LANG=PLI-ENTERPRISE

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I인 것으로 지정합니다.

LANG=PLI-OTHER

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I 이외의 PL/I 레벨인 것으로 지정합니다.

LANG=C

고급 언어 구조의 프로그래밍 언어가 C인 것으로 지정합니다.

LANG=CPP

고급 언어 구조의 프로그래밍 언어가 C++인 것으로 지정합니다.

LOGFILE=value

DFHLS2JS가 해당 활동 로그 및 추적 정보를 기록하는 파일의 완전한 z/OS UNIX 이름입니다. DFHLS2JS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다.

이 파일은 일반적으로 사용하지 않지만 DFHLS2JS 관련 문제점이 발생하는 경우 IBM 서비스 조직이 요청할 수 있습니다.

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0}

웹 서비스 바인딩 파일과 JSON 스키마를 생성할 때 DFHLS2JS가 사용하는 맵핑 레벨을 지정합니다. 다음 옵션을 선택할 수 있습니다.

- 1.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.1** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.1** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.2** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 3.0** 사용 가능한 전체 옵션 세트를 사용하여 JSON 스키마를 생성하려면 이 맵핑 레벨을 사용하십시오.
- 4.0** 이 맵핑 레벨은 CICS TS 5.2 리전에 사용됩니다. 이 맵핑 레벨에서는 COBOL OCCURS DEPENDING ON 필드와 **CHAR-OCCURS** 매개변수를 사용할 수 있습니다.

맵핑 레벨에 대한 자세한 정보는 CICS JSON 지원에 대한 맵핑 레벨을 참조하십시오.

MINIMUM-RUNTIME-

LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0|CURRENT}

웹 서비스 바인딩 파일이 배치될 수 있는 최소 CICS 런타임 환경을 지정합니다. 지정한 다른 매개변수와 일치하지 않는 레벨을 선택하면 오류 메시지를 받게 됩니다. 다음 옵션을 선택할 수 있습니다.

MINIMUM

선택한 매개변수의 경우 CICS의 런타임 레벨 중 가장 낮은 레벨이 자동으로 할당됩니다.

- 1.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 3.0 생성된 웹 서비스 바인딩 파일이 CICS TS 4.1 이상 리전에 배치됩니다.

참고: JSON 지원은 CICS TS 4.2 이상에서만 사용할 수 있습니다.

- 4.0 생성된 웹 서비스 바인딩 파일이 CICS TS 5.2 이상 리전에 배치됩니다. 이 런타임 레벨에서는 **MAPPING-LEVEL** 매개변수에 4.0 이하 맵핑 레벨을 사용할 수 있습니다. 이 레벨에서는 선택적 매개변수를 사용할 수 있습니다.

CURRENT

생성된 웹 서비스 바인딩 파일이 웹 서비스 바인딩 파일을 생성하기 위해 사용하는 런타임 레벨과 동일한 런타임 레벨에서 CICS 리전에 배치됩니다.

PDSLIB=value

처리할 고급 언어 데이터 구조를 포함하는 파티션된 데이터 세트의 이름을 지정합니다. 요청 및 응답에 사용되는 데이터 세트 멤버는 **REQMEM** 및 **RESPMEM** 매개변수에 지정됩니다.

제한사항: 파티션된 데이터 세트 내 레코드는 80바이트의 고정 길이를 갖습니다.

PDSCP=value

REQMEM 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에서 사용되는 코드 페이지를 지정합니다. 여기서 *value*는 CCSID 번호 또는 Java 코드 페이지 번호입니다. 이 매개변수가 지정되지 않으면 z/OS UNIX System Services 코드 페이지를 사용합니다. 예를 들어, **PDSCP=037**을 지정할 수 있습니다.

PGMINT={ CHANNEL | COMMAREA }

서비스 제공자의 경우, CICS가 대상 애플리케이션 프로그램으로 데이터를 전달하는 방식을 지정합니다.

CHANNEL

CICS가 채널 인터페이스를 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

- 맵핑 레벨이 3.0 이하인 경우에는 입출력에 모두 사용되는 하나의 컨테이너만 채널에 포함될 수 있습니다. 컨테이너의 이름을 지정하려면 **CONTID** 매개변수를 사용하십시오. 기본 이름은 DFHWS-DATA입니다.
- 맵핑 레벨 3.0에서는 채널에 여러 컨테이너가 포함될 수 있습니다. **REQUEST-CHANNEL** 및 **RESPONSE-CHANNEL** 매개변수를 사용하십시오. **PDSLIB**, **REQMEM** 또는 **RESPMEM**은 지정하지 마십시오.

COMMAREA

CICS가 통신 영역을 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

대상 애플리케이션 프로그램이 요청을 처리한 경우에는 동일한 메커니즘으로 응답을 리턴해야 합니다. 즉, 요청이 통신 영역에서 수신된 경우에는 응답도 통신 영역에서 리턴되어야 하고 요청이 컨테이너에서 수신된 경우에는 응답도 컨테이너에서 리턴되어야 합니다. CICS가 대상 애플리케이션 프로그램으로 전달하는 통신 영역 또는 컨테이너의 길이는 요청 통신 영역 또는 컨테이너의 길이와 응답 통신 영역 또는 컨테이너의 길이 중 긴 값입니다.

PGMNAME=value

웹 서비스로 노출될 대상 애플리케이션 프로그램에 대한 CICS PROGRAM 자원의 이름을 지정합니다. CICS 웹 서비스 지원은 이 프로그램에 링크됩니다.

REQMEM=value

웹 서비스 요청에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 지정합니다. 서비스 제공자의 경우 웹 서비스 요청은 애플리케이션 프로그램에 입력됩니다.

REQUEST-CHANNEL=value

채널 설명 문서의 이름과 위치를 지정합니다. 채널 설명은 웹 서비스 요청자로부터 JSON 메시지를 수신할 때 웹 서비스 제공자 애플리케이션이 해당 인터페이스에서 사용할 수 있는 컨테이너를 설명합니다. 채널 설명은 CICS 제공 채널 스키마를 준수해야 하는 XML 문서입니다. 자세한 정보는 221 페이지의 『채널 설명 문서 작성』을 참조하십시오.

이 매개변수는 맵핑 레벨 3.0에서만 사용할 수 있습니다.

RESPMEM=value

웹 서비스 응답에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 지정합니다. 서비스 제공자의 경우 웹 서비스 응답은 애플리케이션 프로그램에서 출력됩니다.

RESPONSE-CHANNEL=value

채널 설명 문서의 이름과 위치를 지정합니다. 채널 설명은 웹 서비스 요청자로 JSON 응답 메시지를 보낼 때 웹 서비스 제공자 애플리케이션이 해당 인터페이스에서 사용할 수 있는 컨테이너를 설명합니다. 채널 설명은 CICS 제공 채널 스키마를 준수해야 하는 XML 문서입니다. 자세한 정보는 221 페이지의 『채널 설명 문서 작성』을 참조하십시오.

이 매개변수는 맵핑 레벨 3.0에서만 사용할 수 있습니다.

STRUCTURE=(request,response)

C 및 C++의 경우에만, **REQMEM** 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에 포함된 상위 레벨 구조의 이름을 지정합니다.

request

REQMEM 매개변수가 지정되는 경우 요청을 포함하는 상위 레벨 구조의 이름을 지정합니다. 기본값은 DFHREQUEST입니다.

파티션된 데이터 세트 멤버에는 사용자가 지정하는 이름을 갖는 상위 레벨 구조 또는 이름을 지정하지 않는 경우 DFHREQUEST 구조가 포함되어야 합니다.

response

RESPMEM 매개변수가 지정되는 경우 응답을 포함하는 상위 레벨 구조의 이름을 지정합니다. 기본값은 DFHRESPONSE입니다.

값을 지정하는 경우, 파티션된 데이터 세트 멤버에는 사용자가 지정하는 이름을 갖는 상위 레벨 구조 또는 DFHRESPONSE 구조(이름을 지정하지 않는 경우)가 포함되어야 합니다.

SYNCONRETURN={NO|YES}

원격 웹 서비스가 동기점을 발행할 수 있는지 여부를 지정합니다.

NO 원격 웹 서비스가 동기점을 발행할 수 없습니다. 이 값은 기본값입니다. 원격 웹 서비스가 동기점을 발행하면 ADPL 이상 종료와 함께 실패합니다.

YES 원격 웹 서비스가 동기점을 발행할 수 있습니다. YES를 선택하면 원격 웹 서비스에서 제어가 리턴되는 경우 원격 태스크가 개별 작업 단위로 커밋됩니다. 원격 웹 서비스가 복구 가능 자원을 갱신하고 리턴 후 실패하면 해당 자원에 대한 갱신을 백아웃할 수 없습니다.

TRANSACTION=name

서비스 제공자에서 이 매개변수는 파이프라인을 시작할 수 있는 별명 트랜잭션의 1

- 4자 이름을 지정합니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 TRANSACTION 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ # _ < >

URI=value

이 매개변수는 클라이언트가 웹 서비스에 액세스하기 위해 사용하는 상대 또는 절대 URI를 지정합니다. CICS는 DFHLS2JS로 작성된 웹 서비스 바인딩 파일에서 URIMAP 자원을 생성할 때 지정되는 값을 사용합니다. 이 매개변수는 URIMAP 정의가 적용되는 URI의 경로 컴포넌트를 지정합니다.

USERID=id

서비스 제공자에서 이 매개변수는 웹 클라이언트가 사용할 수 있는 1 - 8자 사용자 ID를 지정합니다. 애플리케이션 생성 응답 또는 웹 서비스의 경우 이 사용자 ID 아래에 별명 트랜잭션이 첨부됩니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 때 URIMAP 자원의 USERID 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ #

WSBIND=value

웹 서비스 바인딩 파일의 완전한 z/OS UNIX 이름입니다. DFHLS2JS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 파일 확장자는 .wsbind입니다.

기타 정보

- DFHLS2JS 실행을 위해 사용되는 사용자 ID는 UNIX System Services를 사용하도록 구성되어야 합니다. 사용자 ID는 CICS z/OS UNIX 파일 구조 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND**, **JSON Schema** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다.
- JCL의 최대 매개변수 길이는 100자입니다. 이 매개변수 길이는 **STDPARM**문을 사용하여 늘릴 수 있습니다. 자세한 정보는 *z/OS UNIX System Services User Guide*를 참조하십시오.

예제

```
//LS2JS JOB 'accounting information',name,MSGCLASS=A
// SET QT='''
//JAVAPROG EXEC DFHLS2JS,
// TMPFILE=&QT.&SYSUID.&QT,
//INPUT.SYSUT1 DD *
```

```

PDSLIB=CICSHLQ.SDFHSAMP
REQMEM=DFH0XCP4
RESPMEM=DFH0XCP4
JSON-SCHEMA-REQUEST=/u/exampleapp/json/example_request.json
JSON-SCHEMA-RESPONSE=/u/exampleapp/json/example_response.json
LANG=COBOL
LOGFILE=/u/exampleapp/wsbind/example.log
MAPPING-LEVEL=4.0
CHAR-VARYING=COLLAPSE
PGMNAME=DFH0XCMN
URI=http://myserver.example.org:8080/exampleApp/example
PGMINT=COMMAREA
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbind/example.wsbind
/*

```

DFHJS2LS: 요청-응답 서비스를 위한 JSON 스키마 대 고급 언어 변환:

DFHJS2LS 프로시저는 JSON 스키마에서 고급 언어 데이터 구조와 웹 서비스 바인딩 파일을 생성합니다. DFHJS2LS는 CICS 애플리케이션 프로그램을 서비스 제공자로 작성하려는 경우 사용할 수 있습니다.

DFHJS2LS와 기호 매개변수, 입력 매개변수와 설명, 작업 예제에 대한 작업 제어 명령문은 이 프로시저를 사용하는 데 도움을 줍니다.

DFHJS2LS JCL 프로시저는 데이터 세트 *HLQ.XDFHINST*에 설치됩니다. 여기서 *HLQ*는 CICS가 설치된 상위 레벨 규정자입니다.

DFHLS2JS 또는 DFHJS2LS 프로시저의 관련 사용 모드는 다음과 같이 요구사항에 따라 다릅니다.

- DFHLS2JS: 링크 가능 인터페이스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 링크 가능 인터페이스를 위한 JSON 스키마 대 고급 언어 변환
- DFHLS2JS: 요청-응답 서비스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 요청-응답 서비스를 위한 JSON 스키마 대 고급 언어 변환
- DFHJS2LS: RESTful 서비스를 위한 JSON 스키마 대 고급 언어 변환

DFHJS2LS의 작업 제어 명령문

JOB 작업을 시작합니다.

EXEC

프로시저 이름(DFHJS2LS)을 지정합니다.

INPUT.SYSUT1 DD

입력을 지정합니다. 입력 매개변수는 일반적으로 입력 스트림에 지정됩니다. 그러나 데이터 세트 또는 파티션된 데이터 세트의 멤버에 정의될 수 있습니다.

기호 매개변수

DFHJS2LS에서는 다음 기호 매개변수가 정의됩니다.

JAVADIR=*path*

DFHJS2LS에서 사용되는 Java 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/에 추가되어 /usr/lpp/*path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

PATHPREF=*prefix*

다른 매개변수 또는 ''(빈 문자열)(접두부를 사용하지 않는 경우)에서 사용되는 z/OS UNIX 디렉토리 경로를 확장하는 접두부를 지정합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **PATHPREF** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

SERVICE=*value*

이 매개변수는 IBM 지원 센터에서 지시하는 경우에만 사용합니다.

TMPDIR=*tmpdir*

DFHJS2LS가 z/OS UNIX에서 임시 작업공간으로 사용하는 디렉토리의 위치를 지정합니다. 작업이 실행되는 사용자 ID는 이 디렉토리에 대한 읽기 및 쓰기 권한이 있어야 합니다.

기본값은 /tmp입니다.

TMPPFILE=*tmpprefix*

DFHJS2LS가 임시 작업공간 파일의 이름을 구성하기 위해 사용하는 접두부를 지정합니다.

기본값은 JS2LS입니다.

USSDIR=*path*

UNIX 시스템 서비스 파일 시스템에서 CICS TS 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/cicsts/에 추가되어 /usr/lpp/cicsts/*path*의 전체 경로 이름을 생성합니다. 기본값을 사용하는 경우 '.'(마침표)로 지정되어야 합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **USSDIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

임시 작업공간

DFHJS2LS는 런타임에 다음 세 개 임시 파일을 작성합니다.

tmpdir/*tmpprefix*.in

tmpdir/*tmpprefix*.out

tmpdir/tmpprefix.err

여기서,

*tmpdir*은 **TMPDIR** 매개변수에 지정된 값입니다.

*tmpprefix*는 **TMPFILE** 매개변수에 지정된 값입니다.

TMPDIR 및 **TMPFILE**이 지정되지 않은 경우 파일의 기본 이름은 다음과 같습니다.

/tmp/JS2LS.in

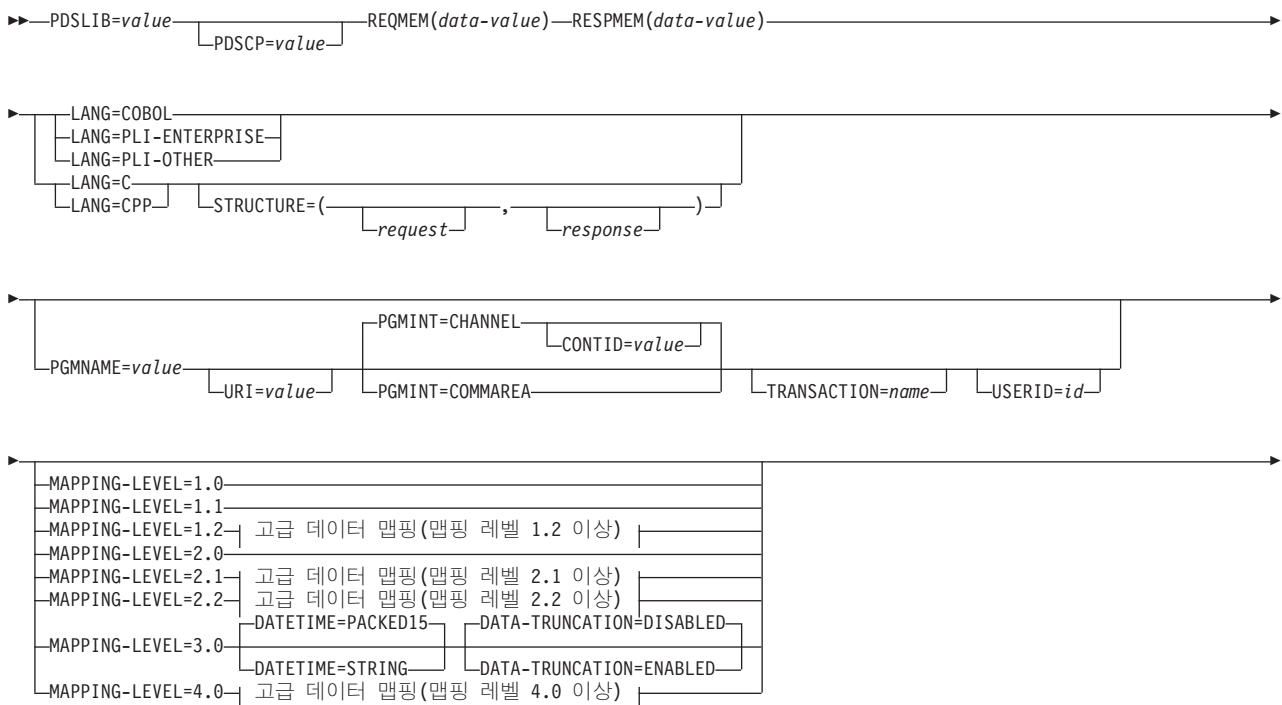
/tmp/JS2LS.out

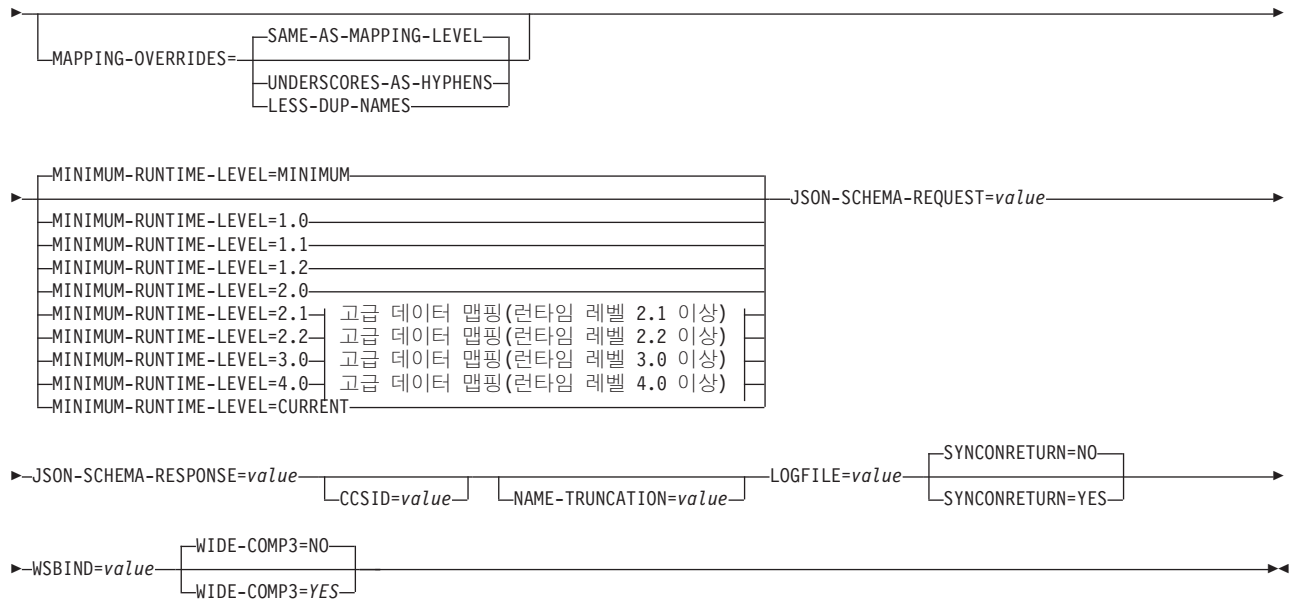
/tmp/JS2LS.err

중요사항: DFHJS2LS는 z/OS UNIX 파일 또는 데이터 세트 멤버에 대한 액세스를 차단하지 않습니다. 따라서 두 개 이상의 DFHJS2LS 인스턴스가 동시에 실행되고 동일한 임시 작업공간 파일을 사용하는 경우, 다른 작업이 작업공간 파일을 사용하는 동안 특정 작업이 해당 파일을 겹쳐쓸 수 있어 예상치 못한 실패를 경험할 수 있습니다.

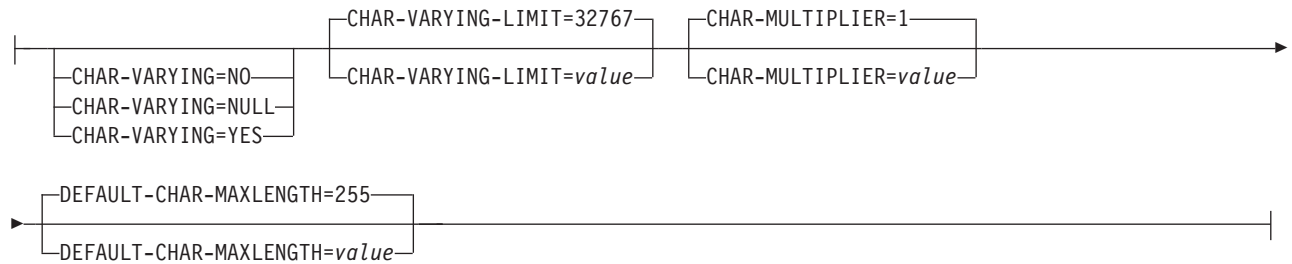
따라서 이러한 상황을 방지하는 이름 지정 규칙과 작동 프로시저를 고안해야 합니다. 예를 들어, 시스템 기호 매개변수 **SYSUID**를 사용하여 개별 사용자에게 고유한 작업공간 파일 이름을 생성할 수 있습니다. 이러한 임시 파일은 작업이 종료되기 전에 삭제됩니다.

DFHJS2LS의 입력 매개변수





고급 데이터 매핑(매핑 레벨 1.2 이상):



고급 데이터 매핑(매핑 레벨 2.1 이상):



매개변수 사용

- 입력 매개변수는 순서에 관계 없이 지정할 수 있습니다.
- 각 매개변수는 행을 바꿔 시작해야 합니다.
- 매개변수는 사용 시 해당 연속 문자와 함께 열 72를 넘어가서는 안됩니다.
열 73 - 80은 공백이어야 합니다.
- 매개변수가 너무 길어 한 행에 맞지 않는 경우에는 행 끝에서 별표(*) 문자를 사용하여 매개변수가 다음 행에서 연속됨을 표시해야 합니다. 별표 앞의 모든 내용은 공백을 포함하여 매개변수의 일부로 간주됩니다. 예를 들어, 다음과 같습니다.

```
WSBIND=wsbinddir*
/app1
```


구문은 다음과 같습니다.

WSBIND=wsbinddir/app1

- 행에서 첫 번째 문자 위치의 # 문자는 주석 문자입니다. 이 행은 무시됩니다.
- 행에서 마지막 문자 위치의 쉼표는 선택적 선 구분 기호이며 무시됩니다.

매개변수 설명

CCSID=value

런타임에 애플리케이션 데이터 구조에서 문자 데이터를 인코딩하는 데 사용되는 CCSID를 지정합니다. 이 매개변수의 값은 **LOCALCCSID** 시스템 초기화 매개변수의 값을 대체합니다. *value*는 Java 및 z/OS Unicode Services User's Guide and Reference에서 지원하는 EBCDIC CCSID여야 합니다. 이 매개변수를 지정하지 않으면 시스템 초기화 매개변수에 지정된 CCSID를 사용하여 애플리케이션 데이터 구조가 인코딩됩니다.

CHAR-MULTIPLIER={1|value}

맵핑 레벨이 1.2 이상인 경우 각 문자에 허용되는 바이트 수를 지정합니다. 이 매개변수의 *value*는 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다. 모든 비숫자 문자 기반 맵핑은 이 승수가 적용됩니다. 2진, 숫자, 존형 및 팩형 10진수 필드는 이 승수의 대상이 아닙니다.

이 매개변수는 예를 들어, DBCS 문자를 사용하려는 경우에 유용할 수 있습니다. 이 경우 런타임 시 모든 2바이트 문자 앞뒤로 잠재적인 SO 및 SI 문자를 위한 공간을 허용하기 위해 승수 3을 선택할 수 있습니다.

CCSID=1200(UTF-16 표시)을 설정하면 **CHAR-MULTIPLIER**에 대한 유일한 올바른 값은 2 또는 4입니다. UTF-16을 사용할 때 기본값은 2입니다. 애플리케이션 데이터가 1개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우 **CHAR-MULTIPLIER=2**를 사용하십시오. 애플리케이션 데이터가 2개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우 **CHAR-MULTIPLIER=4**를 사용하십시오.

참고: **CHAR-MULTIPLIER**를 1로 설정하면 DBCS 문자의 사용이 배제되지 않으며 이를 2로 설정하면 UTF-16 대리 쌍의 사용을 배제하지 않습니다. 그러나 와이드 문자가 주기적으로 사용되면 일부 올바른 값이 할당된 필드에 맞지 않게 됩니다. **CHAR-MULTIPLIER**의 값으로 더 큰 값을 사용할 수록 XML에서 올바른 문자 수보다 더 많은 문자를 할당된 필드에 저장할 수 있습니다. 해당 범위 제한을 주의하여 따라야 합니다.

CHAR-VARYING={NO|NULL|YES}

맵핑 레벨이 1.2 이상인 경우 가변 길이 문자 데이터가 맵핑되는 방식을 지정합니다.

다. 가변 길이 2진 데이터 유형은 항상 컨테이너 또는 가변 구조에 맵핑됩니다. 이 매개변수를 지정하지 않은 경우 지정한 언어에 따라 기본 맵핑이 달라집니다. 다음 옵션을 선택할 수 있습니다.

NO 가변 길이 문자 데이터가 고정 길이 문자열로 맵핑됩니다.

NULL

가변 길이 문자 데이터가 널 종료 문자열로 맵핑됩니다.

YES PL/I의 경우 가변 길이 문자 데이터가 CHAR VARYING 데이터 유형에 맵핑됩니다. COBOL, C, C++ 언어에서는 가변 길이 문자 데이터가 두 가지 관련 요소(데이터-길이 및 데이터)로 구성되는 해당 표시에 맵핑됩니다.

CHAR-VARYING-LIMIT={32767}|value}

맵핑 레벨이 1.2 이상인 경우 언어 구조에 맵핑되는 가변 길이 문자 데이터와 2진 데이터의 최대 크기를 지정합니다. 문자 또는 2진 데이터가 이 매개변수에 지정된 값보다 큰 경우에는 컨테이너에 맵핑되고 생성된 언어 구조에서 컨테이너 이름이 사용됩니다. 값의 범위는 0 - 32,767바이트(기본값)입니다.

CONTID=value

서비스 제공자에서, JSON 메시지를 나타내는 데 사용되는 상위 레벨 데이터 구조를 보유하는 컨테이너의 이름을 지정합니다.

CICS가 대상 애플리케이션 프로그램으로 전달하는 컨테이너의 길이는 요청 컨테이너의 길이와 응답 컨테이너의 길이 중 긴 값입니다.

DATA-TRUNCATION={DISABLED}|ENABLED}

가변 길이 데이터가 고정 길이 필드 구조에서 용인되는지 여부를 지정합니다.

DISABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 거부하고 오류 메시지를 발행합니다.

ENABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 용인하고 누락된 데이터를 널값으로 처리합니다.

DATETIME={PACKED15}|STRING}

JSON 날짜-시간 요소가 언어 구조에 맵핑되는 방식을 지정합니다.

PACKED15

기본적으로 모든 JSON 날짜-시간 요소는 시간소인으로 처리되고 CICS ABSTIME 형식에 맵핑됩니다.

STRING

JSON 날짜-시간 요소가 텍스트로 처리됩니다.

DEFAULT-CHAR-MAXLENGTH={255}|value}

맵핑 레벨이 1.2 이상이고 웹 서비스 설명 문서에 길이가 나타나지 않는 경우 맵핑

문자 내 문자 데이터의 기본 배열 길이를 지정합니다. 이 매개변수의 값은 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다.

INLINE-MAXOCCURS-LIMIT={1|value}

인라인 가변 반복 콘텐츠가 maxItems JSON 스키마 키워드를 기반으로 사용되는지 여부를 지정합니다. 인라인으로 맵핑되는 가변 반복 콘텐츠는 생성된 언어 구조의 나머지와 함께 현재 컨테이너에 배치됩니다. 가변 반복 콘텐츠는 두 파트로, 즉 데이터 발생 회수를 저장하는 카운터와 데이터의 각 발생을 저장하는 배열로 저장됩니다. 가변 반복 콘텐츠의 대체 맵핑은 컨테이너 기반 맵핑으로, 데이터 발생 횟수와 데이터가 배치되는 컨테이너의 이름을 저장합니다. 개별 컨테이너에 데이터를 저장하는 경우 성능에 영향을 주어 인라인 맵핑이 선호될 수 있습니다.

INLINE-MAXOCCURS-LIMIT 매개변수는 맵핑 레벨 2.1 이상에서만 사용할 수 있습니다. **INLINE-MAXOCCURS-LIMIT** 값은 0 - 32,767 범위의 양의 정수일 수 있습니다. 값이 0이면 인라인 맵핑이 사용되지 않음을 나타냅니다. 값이 1이면 선택적 요소가 인라인으로 맵핑됩니다. maxOccurs 속성의 value가 **INLINE-MAXOCCURS-LIMIT**의 value보다 크면 컨테이너 기반 맵핑을 사용하고 그렇지 않으면 인라인 맵핑을 사용합니다.

가변 반복 목록을 인라인으로 맵핑하는지 여부를 결정할 때 반복 데이터의 단일 항목 길이를 고려해야 합니다. 길이가 긴 인스턴스가 거의 없는 경우에는 컨테이너 기반 맵핑이 선호됩니다. 길이가 짧은 인스턴스가 많은 경우에는 인라인 맵핑이 선호됩니다.

JSON-SCHEMA-REQUEST=value

필수 매개변수입니다.

값은 요청 JSON 스키마가 저장되는 UNIX System Services 위치를 나타냅니다.

JSON-SCHEMA-RESPONSE=value

필수 매개변수입니다.

값은 응답 JSON 스키마가 저장되는 UNIX System Services 위치를 나타냅니다.

LANG=COBOL

고급 언어 구조의 프로그래밍 언어가 COBOL인 것으로 지정합니다.

LANG=PLI-ENTERPRISE

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I인 것으로 지정합니다.

LANG=PLI-OTHER

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I 이외의 PL/I 레벨인 것으로 지정합니다.

LANG=C

고급 언어 구조의 프로그래밍 언어가 C인 것으로 지정합니다.

LANG=CPP

고급 언어 구조의 프로그래밍 언어가 C++인 것으로 지정합니다.

LOGFILE=value

DFHJS2LS가 해당 활동 로그 및 추적 정보를 기록하는 파일의 완전한 z/OS UNIX 이름입니다. DFHJS2LS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다.

이 파일은 일반적으로 사용하지 않지만 DFHJS2LS 관련 문제점이 발생하는 경우 IBM 서비스 조직이 요청할 수 있습니다.

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0}

웹 서비스 바인딩 파일과 언어 구조를 생성할 때 DFHJS2LS가 사용하는 맵핑 레벨을 지정합니다. 다음 옵션을 선택할 수 있습니다.

- 1.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.1** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.1** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.2** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 3.0** 사용 가능한 전체 옵션 세트를 사용하여 JSON 스키마를 생성하려면 이 맵핑 레벨을 사용하십시오.
- 4.0** UTF-16을 사용하려는 경우 CICS TS 5.2 리전에 이 맵핑 레벨을 사용하십시오.

맵핑 레벨에 대한 자세한 정보는 CICS 지원에 대한 맵핑 레벨을 참조하십시오.

MAPPING-OVERRIDES={SAME-AS-MAPPING-LEVEL|UNDERScores-AS-HYPHENS|LESS-DUP-NAMES}

언어 구조를 생성할 때 지정된 맵핑 레벨에 대해 기본 동작이 대체되는지 여부를 지정합니다.

LESS-DUP-NAMES

이 매개변수는 이름 끝에 `_value`가 있는 비구조적 구조 필드 이름을 생성하여 필드를 직접 참조할 수 있습니다. 예를 들어, 다음 PLI 언어 구조에

서 MAPPING-OVERRIDES=LESS-DUP-NAMES가 지정되는 경우 레벨 12 필드 streetName에는 접미부 _value가 추가됩니다.

```
09 streetName,
   12 streetName                CHAR(255) VARYING
   UNALIGNED,
   12 filler                    BIT (7),
   12 attr_nil_streetName_value BIT (1),
```

결과적으로 구조는 다음과 같습니다.

```
09 streetName,
   12 streetName_value          CHAR(255) VARYING
   UNALIGNED,
   12 filler                    BIT (7),
   12 attr_nil_streetName_value BIT (1),
```

SAME-AS-MAPPING-LEVEL

이 매개변수는 맵핑 레벨과 동일한 스타일의 언어 구조를 생성합니다. 이는 기본값입니다.

UNDERSCORES-AS-HYPHENS

COBOL에서만 사용됩니다. 이 매개변수는 JSON 스키마의 밑줄을 문자 X가 아닌 하이픈으로 변환하여 생성된 COBOL 언어 구조의 가독성을 향상시킵니다. 필드 이름 충돌이 발생하면 필드가 고유할 수 있도록 필드에 번호가 지정됩니다. 자세한 정보는 개발 애플리케이션에서 COBOL에 대한 JSON 스키마 매핑을 참조하십시오.

MINIMUM - RUNTIME -

LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0|CURRENT}

웹 서비스 바인딩 파일이 배치될 수 있는 최소 CICS 런타임 환경을 지정합니다. 지정한 다른 매개변수와 일치하지 않는 레벨을 선택하면 오류 메시지를 받게 됩니다. 다음 옵션을 선택할 수 있습니다.

MINIMUM

선택한 매개변수의 경우 CICS의 런타임 레벨 중 가장 낮은 레벨이 자동으로 할당됩니다.

- 1.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.1** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

2.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

2.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

3.0 생성된 웹 서비스 바인딩 파일이 CICS TS 4.1 이상 리전에 배치됩니다.

참고: JSON 지원은 CICS TS 4.2 이상에서만 사용할 수 있습니다.

4.0 생성된 웹 서비스 바인딩 파일이 CICS TS 5.2 이상 리전에 배치됩니다. 이 런타임 레벨에서는 **MAPPING-LEVEL** 매개변수에 4.0 이하 맵핑 레벨을 사용할 수 있습니다. 이 레벨에서는 선택적 매개변수를 사용할 수 있습니다.

CURRENT

생성된 웹 서비스 바인딩 파일이 웹 서비스 바인딩 파일을 생성하기 위해 사용하는 런타임 레벨과 동일한 런타임 레벨에서 CICS 리전에 배치됩니다.

NAME-TRUNCATION={LEFT|RIGHT}

JSON 이름이 왼쪽 또는 오른쪽부터 잘리는지 여부를 지정합니다. CICS 웹 서비스 지원은 지정된 고급 언어에 적합한 길이로 JSON 이름을 자릅니다. 기본적으로 이름은 오른쪽부터 잘립니다.

PDSCP=value

REQMEM 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에서 사용되는 코드 페이지를 지정합니다. 여기서 *value*는 CCSID 번호 또는 Java 코드 페이지 번호입니다. 이 매개변수가 지정되지 않으면 z/OS UNIX System Services 코드 페이지를 사용합니다. 예를 들어, **PDSCP=037**을 지정할 수 있습니다.

PDSLIB=value

생성된 고급 언어를 포함하는 파티션된 데이터 세트의 이름을 지정합니다. 요청 및 응답에 사용되는 데이터 세트 멤버는 **REQMEM** 및 **RESPMEM** 매개변수에 지정됩니다.

PGMINT={CHANNEL|COMMAREA}

서비스 제공자의 경우, CICS가 대상 애플리케이션 프로그램으로 데이터를 전달하는 방식을 지정합니다.

CHANNEL

CICS가 채널 인터페이스를 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

COMMAREA

CICS가 통신 영역을 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

대상 애플리케이션 프로그램이 요청을 처리한 경우에는 동일한 메커니즘으로 응답을 리턴해야 합니다. 즉, 요청이 통신 영역에서 수신된 경우에는 응답도 통신 영역

에서 리턴되어야 하고 요청이 컨테이너에서 수신된 경우에는 응답도 컨테이너에서 리턴되어야 합니다. CICS가 대상 애플리케이션 프로그램으로 전달하는 통신 영역 또는 컨테이너의 길이는 요청 통신 영역 또는 컨테이너의 길이와 응답 통신 영역 또는 컨테이너의 길이 중 긴 값입니다.

PGMNAME=value

CICS PROGRAM 자원의 이름을 지정합니다.

DFHJS2LS가 서비스 제공자에서 사용되는 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우 이 매개변수를 제공해야 합니다. 이 매개변수는 웹 서비스로 노출되는 애플리케이션 프로그램의 자원 이름을 지정합니다.

DFHJS2LS가 서비스 요청자에서 사용되는 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우에는 이 매개변수를 생략하십시오.

REQMEM=value

DFHJS2LS가 웹 서비스 요청(애플리케이션 프로그램의 입력 데이터)에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 사용하는 1 - 6자 접두부를 지정합니다.

DFHJS2LS는 접두부에 01을 추가하여 파티션된 데이터 세트 멤버의 이름을 생성합니다.

RESPMEM=value

DFHJS2LS가 웹 서비스 응답(애플리케이션 프로그램의 출력 데이터)에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 사용하는 1 - 6자 접두부를 지정합니다.

DFHJS2LS는 접두부에 01을 추가하여 파티션된 데이터 세트 멤버의 이름을 생성합니다.

STRUCTURE=(request,response)

C 및 C++의 경우에만, 요청 및 응답 구조의 이름이 생성되는 방식을 지정합니다.

생성된 요청 및 응답 구조에는 이름 *request01* 및 *response01*이 지정됩니다.

두 이름 중 하나 또는 두 이름이 모두 생략되는 경우, 구조 이름은 사용자가 지정하는 **REQMEM** 및 **RESPMEM** 매개변수에서 생성된 파티션된 데이터 세트 멤버 이름과 같습니다.

SYNCONRETURN={NO|YES}

원격 웹 서비스가 동기점을 발행할 수 있는지 여부를 지정합니다.

NO 원격 웹 서비스가 동기점을 발행할 수 없습니다. 이 값은 기본값입니다. 원격 웹 서비스가 동기점을 발행하면 ADPL 이상 종료와 함께 실패합니다.

YES 원격 웹 서비스가 동기점을 발행할 수 있습니다. YES를 선택하면 원격 웹 서비스에서 제어가 리턴되는 경우 원격 태스크가 개별 작업 단위로 커밋

됩니다. 원격 웹 서비스가 복구 가능 자원을 갱신하고 리턴 후 실패하면 해당 자원에 대한 갱신을 백아웃할 수 없습니다.

TRANSACTION=name

서비스 제공자에서 이 매개변수는 파이프라인을 시작할 수 있는 별명 트랜잭션의 1 - 4자 이름을 지정합니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 TRANSACTION 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ # _ < >

URI=value

서비스 제공자에서, 이 매개변수는 클라이언트가 웹 서비스에 액세스하기 위해 사용하는 상대 URI를 지정합니다. CICS는 DFHJS2LS로 작성된 웹 서비스 바인딩 파일에서 URIMAP 자원을 생성할 때 지정되는 값을 사용합니다. 이 매개변수는 URIMAP 정의가 적용되는 URI의 경로 컴포넌트를 지정합니다.

USERID=id

서비스 제공자에서 이 매개변수는 웹 클라이언트가 사용할 수 있는 1 - 8자 사용자 ID를 지정합니다. 애플리케이션 생성 응답 또는 웹 서비스의 경우 이 사용자 ID 아래에 별명 트랜잭션이 첨부됩니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 때 URIMAP 자원의 USERID 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3={NO|YES}

COBOL에서만 사용됩니다. COBOL 언어 구조에서 팩형 10진수 가변 길이의 최대 크기를 제어합니다.

NO DFHJS2LS가 COBOL 언어 구조 유형 COMP-3을 생성할 때 팩형 10진수 가변 길이를 18로 제한합니다. 팩형 10진수 크기가 18보다 큰 경우, 지정된 유형이 총 18자로 제한됨을 나타내는 DFHPI9022W 메시지가 발행됩니다.

YES DFHJS2LS가 COBOL 언어 구조 유형 COMP-3을 생성할 때 최대 크기 31을 지원합니다.

WSBIND=value

웹 서비스 바인딩 파일의 완전한 z/OS UNIX 이름입니다. DFHJS2LS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 기본 파일 확장자는 .wsbind입니다.

기타 정보

- DFHJS2LS가 실행되는 사용자 ID는 UNIX System Services를 사용하도록 구성되어야 합니다. 사용자 ID는 CICS z/OS UNIX 파일 구조 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND**, **WSDL** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다.
- JCL의 최대 매개변수 길이는 100자입니다. 이 길이는 **STDPARM**문을 사용하여 늘릴 수 있습니다. 자세한 정보는 *z/OS UNIX System Services User Guide*를 참조하십시오.

예제

```
//JS2LS JOB  'accounting information',name,MSGCLASS=A
//  SET QT='''
//JAVAPROG EXEC DFHJS2LS,
// TMPFILE=&QT.&SYSUID.&QT
/INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
REQMEM=CPYBK1
RESPMEM=CPYBK2
JSON-SCHEMA-REQUEST=example.json
JSON-SCHEMA-RESPONSE=example.json
LANG=COBOL
LOGFILE=/u/exampleapp/wsbinding/example.log
MAPPING-LEVEL=4.0
CHAR-VARYING=NULL
INLINE-MAXOCCURS-LIMIT=2
PGMNAME=DFH0XCMN
URI=exampleApp/example
PGMINT=COMMAREA
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbinding/example.wsbinding
/*
```

DFHJS2LS: RESTful 서비스를 위한 JSON 스키마 대 고급 언어 변환:

DFHJS2LS 프로시저는 JSON 스키마에서 고급 언어 데이터 구조와 웹 서비스 바인딩 파일을 생성합니다. DFHJS2LS는 RESTful JSON 서비스 제공자 작성을 준비할 때 사용할 수 있습니다.

DFHJS2LS와 기호 매개변수, 입력 매개변수와 설명, 작업 예제에 대한 작업 제어 명령문은 이 프로시저를 사용하는 데 도움을 줍니다.

DFHJS2LS JCL 프로시저는 데이터 세트 *HLQ.XDFHINST*에 설치됩니다. 여기서 *HLQ*는 CICS가 설치된 상위 레벨 규정자입니다.

DFHLS2JS 또는 DFHJS2LS 프로시저의 관련 사용 모드는 다음과 같이 요구사항에 따라 다릅니다.

- DFHLS2JS: 링크 가능 인터페이스를 위한 고급 언어 대 JSON 스키마 변환

- DFHJS2LS: 링크 가능 인터페이스를 위한 JSON 스키마 대 고급 언어 변환
- DFHLS2JS: 요청-응답 서비스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 요청-응답 서비스를 위한 JSON 스키마 대 고급 언어 변환
- DFHJS2LS: RESTful 서비스를 위한 JSON 스키마 대 고급 언어 변환

DFHJS2LS의 작업 제어 명령문

JOB 작업을 시작합니다.

EXEC

프로시저 이름(DFHJS2LS)을 지정합니다.

INPUT.SYSUT1 DD

입력을 지정합니다. 입력 매개변수는 일반적으로 입력 스트림에 지정됩니다. 그러나 데이터 세트 또는 파티션된 데이터 세트의 멤버에 정의될 수 있습니다.

기호 매개변수

DFHJS2LS에서는 다음 기호 매개변수가 정의됩니다.

JAVADIR=*path*

DFHJS2LS에서 사용되는 Java 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/에 추가되어 /usr/lpp/*path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

PATHPREFIX=*prefix*

다른 매개변수 또는 ''(빈 문자열)(접두부를 사용하지 않는 경우)에서 사용되는 z/OS UNIX 디렉토리 경로를 확장하는 접두부를 지정합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **PATHPREFIX** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

SERVICE=*value*

이 매개변수는 IBM 지원 센터에서 지시하는 경우에만 사용합니다.

TMPDIR=*tmpdir*

DFHJS2LS가 z/OS UNIX에서 임시 작업공간으로 사용하는 디렉토리의 위치를 지정합니다. 작업이 실행되는 사용자 ID는 이 디렉토리에 대한 읽기 및 쓰기 권한이 있어야 합니다.

기본값은 /tmp입니다.

TMPPFILE=*tmpprefix*

DFHJS2LS가 임시 작업공간 파일의 이름을 구성하기 위해 사용하는 접두부를 지정합니다.

기본값은 JS2LS입니다.

USSDIR=path

UNIX 시스템 서비스 파일 시스템에서 CICS TS 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/cicsts/에 추가되어 /usr/lpp/cicsts/path의 전체 경로 이름을 생성합니다. 기본값을 사용하는 경우 '.'(마침표)로 지정되어야 합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **USSDIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

임시 작업공간

DFHJS2LS는 런타임에 다음 세 개 임시 파일을 작성합니다.

```
tmpdir/tmpprefix.in  
tmpdir/tmpprefix.out  
tmpdir/tmpprefix.err
```

여기서,

*tmpdir*은 **TMPDIR** 매개변수에 지정된 값입니다.

*tmpprefix*는 **TMPFILE** 매개변수에 지정된 값입니다.

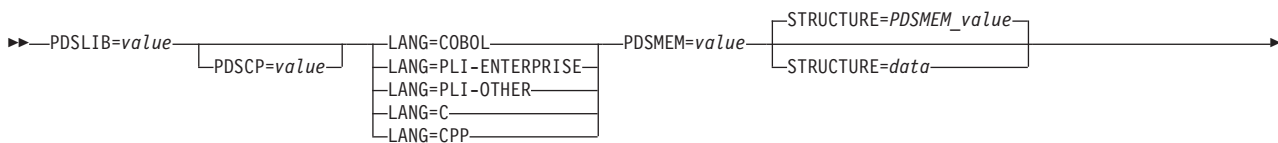
TMPDIR 및 **TMPFILE**이 지정되지 않은 경우 파일의 기본 이름은 다음과 같습니다.

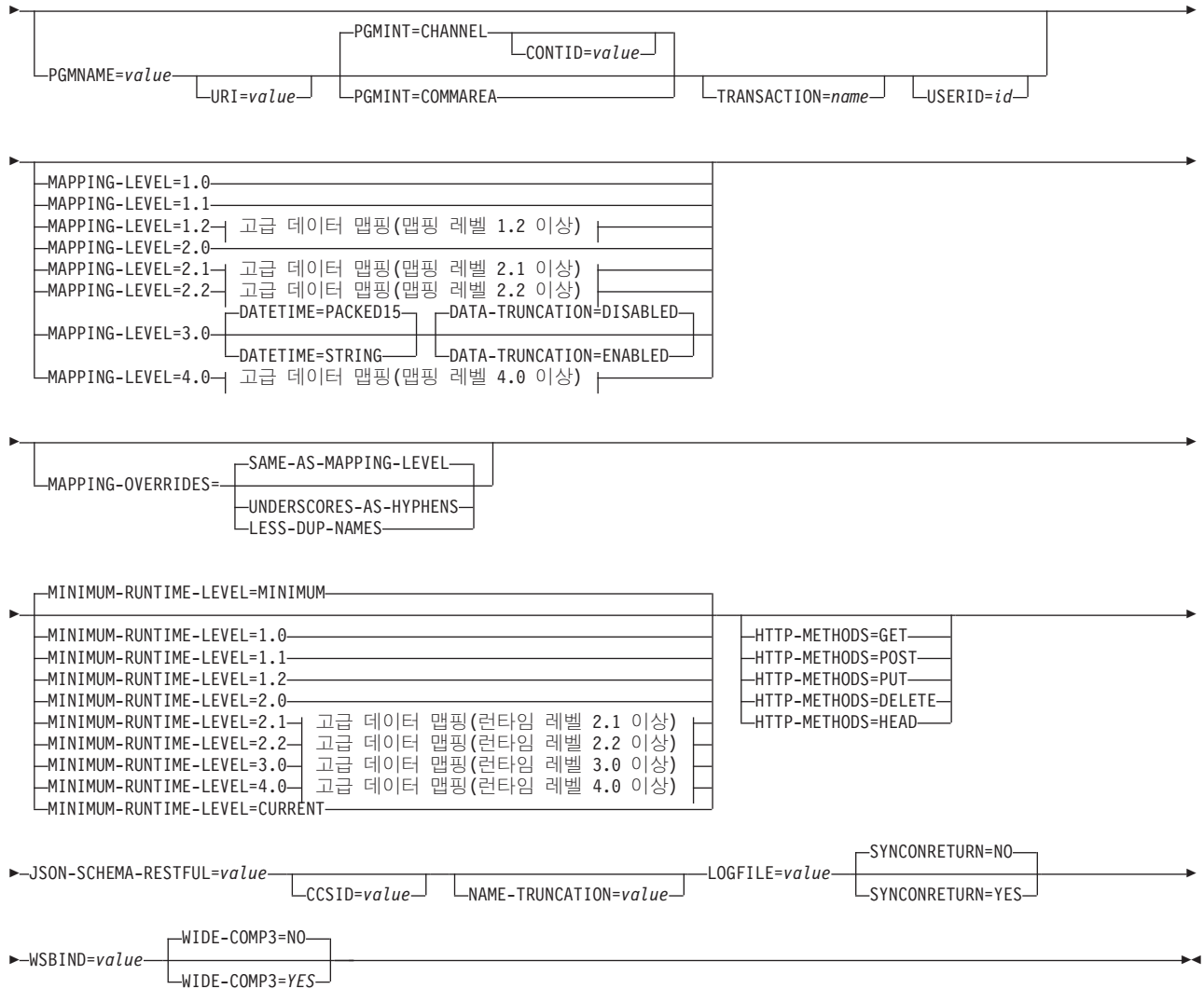
```
/tmp/JS2LS.in  
/tmp/JS2LS.out  
/tmp/JS2LS.err
```

중요사항: DFHJS2LS는 z/OS UNIX 파일 또는 데이터 세트 멤버에 대한 액세스를 차단하지 않습니다. 따라서 두 개 이상의 DFHJS2LS 인스턴스가 동시에 실행되고 동일한 임시 작업공간 파일을 사용하는 경우, 다른 작업이 작업공간 파일을 사용하는 동안 특정 작업이 해당 파일을 겹쳐쓸 수 있어 예상치 못한 실패를 경험할 수 있습니다.

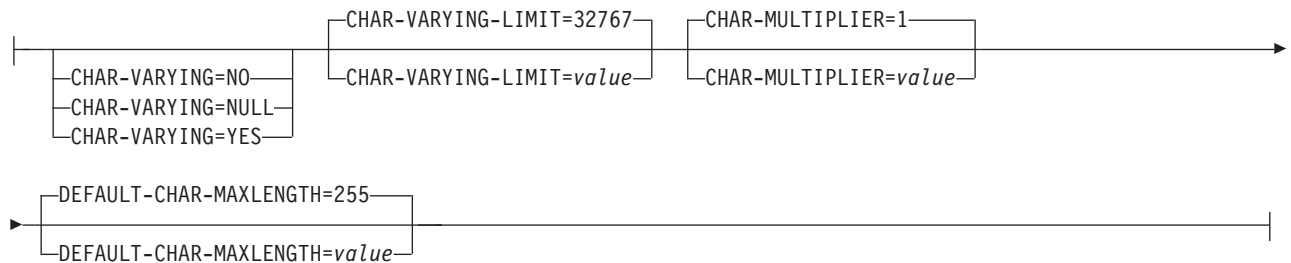
따라서 이러한 상황을 방지하는 이름 지정 규칙과 작동 프로시저를 고안해야 합니다. 예를 들어, 시스템 기호 매개변수 **SYSUID**를 사용하여 개별 사용자에게 고유한 작업공간 파일 이름을 생성할 수 있습니다. 이러한 임시 파일은 작업이 종료되기 전에 삭제됩니다.

DFHJS2LS의 입력 매개변수





고급 데이터 맵핑(맵핑 레벨 1.2 이상):



고급 데이터 맵핑(맵핑 레벨 2.1 이상):



매개변수 사용

- 입력 매개변수는 순서에 관계 없이 지정할 수 있습니다.
- 각 매개변수는 행을 바꿔 시작해야 합니다.
- 매개변수는 사용 시 해당 연속 문자와 함께 열 72를 넘어가서는 안됩니다.
열 73 - 80은 공백이어야 합니다.
- 매개변수가 너무 길어 한 행에 맞지 않는 경우에는 행 끝에서 별표(*) 문자를 사용하여 매개변수가 다음 행에서 연속됨을 표시해야 합니다. 별표 앞의 모든 내용은 공백을 포함하여 매개변수의 일부로 간주됩니다. 예를 들어, 다음과 같습니다.

```
WSBIND=wsbinddir*  
/app1
```

구문은 다음과 같습니다.

```
WSBIND=wsbinddir/app1
```

- 행에서 첫 번째 문자 위치의 # 문자는 주석 문자입니다. 이 행은 무시됩니다.
- 행에서 마지막 문자 위치의 쉼표는 선택적 선 구분 기호이며 무시됩니다.

매개변수 설명

CCSID=*value*

런타임에 애플리케이션 데이터 구조에서 문자 데이터를 인코딩하는 데 사용되는 CCSID를 지정합니다. 이 매개변수의 값은 **LOCALCCSID** 시스템 초기화 매개변수의 값을 대체합니다. *value*는 Java 및 z/OS Unicode Services User's Guide and Reference에서 지원하는 EBCDIC CCSID여야 합니다. 이 매개변수를 지정하지 않으면 시스템 초기화 매개변수에 지정된 CCSID를 사용하여 애플리케이션 데이터 구조가 인코딩됩니다.

CHAR-MULTIPLIER={1|*value*}

맵핑 레벨이 1.2 이상인 경우 각 문자에 허용되는 바이트 수를 지정합니다. 이 매개변수의 *value*는 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다. 모든 비숫자 문자 기반 맵핑은 이 승수가 적용됩니다. 2진, 숫자, 존형 및 팩형 10진수 필드는 이 승수의 대상이 아닙니다.

이 매개변수는 예를 들어, DBCS 문자를 사용하려는 경우에 유용할 수 있습니다. 이 경우 런타임 시 모든 2바이트 문자 앞뒤로 잠재적인 SO 및 SI 문자를 위한 공간을 허용하기 위해 승수 3을 선택할 수 있습니다.

CCSID=1200(UTF-16 표시)을 설정하면 **CHAR-MULTIPLIER**에 대한 유일한 올바른 값은 2 또는 4입니다. UTF-16을 사용할 때 기본값은 2입니다. 애플리케이션 데이터가 1개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우 **CHAR-MULTIPLIER=2**를 사용하십시오. 애플리케이션 데이터가 2개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우 **CHAR-MULTIPLIER=4**를 사용하십시오.

참고: **CHAR-MULTIPLIER**를 1로 설정하면 DBCS 문자의 사용이 배제되지 않으며 이를 2로 설정하면 UTF-16 대리 쌍의 사용을 배제하지 않습니다. 그러나 와이드 문자가 주기적으로 사용되면 일부 올바른 값이 할당된 필드에 맞지 않게 됩니다. **CHAR-MULTIPLIER**의 값으로 더 큰 값을 사용할 수록 XML에서 올바른 문자 수보다 더 많은 문자를 할당된 필드에 저장할 수 있습니다. 해당 범위 제한을 주의하여 따라야 합니다.

CHAR-VARYING=NO|NULL|YES

맵핑 레벨이 1.2 이상인 경우 가변 길이 문자 데이터가 맵핑되는 방식을 지정합니다. 가변 길이 2진 데이터 유형은 항상 컨테이너 또는 가변 구조에 맵핑됩니다. 이 매개변수를 지정하지 않은 경우 지정한 언어에 따라 기본 맵핑이 달라집니다. 다음 옵션을 선택할 수 있습니다.

NO 가변 길이 문자 데이터가 고정 길이 문자열로 맵핑됩니다.

NULL

가변 길이 문자 데이터가 널 종료 문자열로 맵핑됩니다.

YES PL/I의 경우 가변 길이 문자 데이터가 CHAR VARYING 데이터 유형에 맵핑됩니다. COBOL, C, C++ 언어에서는 가변 길이 문자 데이터가 두 가지 관련 요소(데이터-길이 및 데이터)로 구성되는 해당 표시에 맵핑됩니다.

CHAR-VARYING-LIMIT=32767|value

맵핑 레벨이 1.2 이상인 경우 언어 구조에 맵핑되는 가변 길이 문자 데이터와 2진 데이터의 최대 크기를 지정합니다. 문자 또는 2진 데이터가 이 매개변수에 지정된 값보다 큰 경우에는 컨테이너에 맵핑되고 생성된 언어 구조에서 컨테이너 이름이 사용됩니다. 값의 범위는 0 - 32,767바이트(기본값)입니다.

CONTID=value

서비스 제공자에서, JSON 메시지를 나타내는 데 사용되는 상위 레벨 데이터 구조를 보유하는 컨테이너의 이름을 지정합니다.

CICS가 대상 애플리케이션 프로그램으로 전달하는 컨테이너의 길이는 요청 컨테이너의 길이와 응답 컨테이너의 길이 중 긴 값입니다.

DATA-TRUNCATION={DISABLED|ENABLED}

가변 길이 데이터가 고정 길이 필드 구조에서 용인되는지 여부를 지정합니다.

DISABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 거부하고 오류 메시지를 발행합니다.

ENABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 용인하고 누락된 데이터를 널값으로 처리합니다.

DATE TIME=PACKED15|STRING

JSON 날짜-시간 요소가 언어 구조에 맵핑되는 방식을 지정합니다.

PACKED15

기본적으로 모든 JSON 날짜-시간 요소는 시간소인으로 처리되고 CICS ABSTIME 형식에 맵핑됩니다.

STRING

JSON 날짜-시간 요소가 텍스트로 처리됩니다.

DEFAULT-CHAR-MAXLENGTH=255|value

맵핑 레벨이 1.2 이상이고 웹 서비스 설명 문서에 길이가 나타나지 않는 경우 맵핑 문자 내 문자 데이터의 기본 배열 길이를 지정합니다. 이 매개변수의 값은 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다.

HTTP-METHODS={GET|POST|PUT|DELETE|HEAD},{GET|POST|PUT|DELETE|HEAD},*
선택적 매개변수입니다.

기본값은 GET, POST, PUT, DELETE를 설정하는 것입니다. 이는 애플리케이션이 네 가지 기본 RESTful 조작을 지원함을 DFHJS2LS에 알려줍니다.

값이 제공되는 경우, DFHJS2LS는 명시적으로 지정된 HTTP 메소드만 허용되는 WSBInd 파일을 빌드합니다.

애플리케이션이 HEAD 메소드를 구현하려는 경우에는 의도적으로 선택해야 합니다. 기본적으로 DFHJS2LS는 애플리케이션이 수신 HTTP HEAD 메소드를 지원하지 않는 것으로 가정합니다.

JSON 클라이언트가 지원되지 않는 HTTP 메소드를 사용하려고 시도하면 CICS가 HTTP 405 응답으로 응답합니다.

INLINE-MAXOCCURS-LIMIT=1|value

인라인 가변 반복 콘텐츠가 maxItems JSON 스키마 키워드를 기반으로 사용되는지 여부를 지정합니다. 인라인으로 맵핑되는 가변 반복 콘텐츠는 생성된 언어 구조의 나머지와 함께 현재 컨테이너에 배치됩니다. 가변 반복 콘텐츠는 두 파트로, 즉 데이터 발생 회수를 저장하는 카운터와 데이터의 각 발생을 저장하는 배열로 저장됩니다. 가변 반복 콘텐츠의 대체 맵핑은 컨테이너 기반 맵핑으로, 데이터 발생 횟수와 데이터가 배치되는 컨테이너의 이름을 저장합니다. 개별 컨테이너에 데이터를 저장하는 경우 성능에 영향을 주어 인라인 맵핑이 선호될 수 있습니다.

INLINE-MAXOCCURS-LIMIT 매개변수는 맵핑 레벨 2.1 이상에서만 사용할 수 있습니다. **INLINE-MAXOCCURS-LIMIT** 값은 0 - 32,767 범위의 양의 정수일 수 있습니다. 값이 0이면 인라인 맵핑이 사용되지 않음을 나타냅니다. 값이 1이면 선택적 요소가 인라인으로 맵핑됩니다. maxOccurs 속성의 value가

INLINE-MAXOCCURS-LIMIT의 value보다 크면 컨테이너 기반 맵핑을 사용하고 그렇지 않으면 인라인 맵핑을 사용합니다.

가변 반복 목록을 인라인으로 맵핑하는지 여부를 결정할 때 반복 데이터의 단일 항목 길이를 고려해야 합니다. 길이가 긴 인스턴스가 거의 없는 경우에는 컨테이너 기반 맵핑이 선호됩니다. 길이가 짧은 인스턴스가 많은 경우에는 인라인 맵핑이 선호됩니다.

JSON-SCHEMA-RESTFUL=*value*

필수 매개변수입니다.

값은 응답 JSON 스키마가 저장되는 UNIX System Services 위치를 나타냅니다.

LANG=COBOL

고급 언어 구조의 프로그래밍 언어가 COBOL인 것으로 지정합니다.

LANG=PLI-ENTERPRISE

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I인 것으로 지정합니다.

LANG=PLI-OTHER

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I 이외의 PL/I 레벨인 것으로 지정합니다.

LANG=C

고급 언어 구조의 프로그래밍 언어가 C인 것으로 지정합니다.

LANG=CPP

고급 언어 구조의 프로그래밍 언어가 C++인 것으로 지정합니다.

LOGFILE=*value*

DFHJS2LS가 해당 활동 로그 및 추적 정보를 기록하는 파일의 완전한 z/OS UNIX 이름입니다. DFHJS2LS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다.

이 파일은 일반적으로 사용하지 않지만 DFHJS2LS 관련 문제점이 발생하는 경우 IBM 서비스 조직이 요청할 수 있습니다.

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0}

웹 서비스 바인딩 파일과 언어 구조를 생성할 때 DFHJS2LS가 사용하는 맵핑 레벨을 지정합니다. 다음 옵션을 선택할 수 있습니다.

- 1.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.1** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

- 2.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 3.0 사용 가능한 전체 옵션 세트를 사용하여 JSON 스키마를 생성하려면 이 맵핑 레벨을 사용하십시오.
- 4.0 UTF-16을 사용하려는 경우 CICS TS 5.2 리전에 이 맵핑 레벨을 사용하십시오.

맵핑 레벨에 대한 자세한 정보는 애플리케이션 개발에서 CICS JSON 지원자에 대한 맵핑 레벨을 참조하십시오.

MAPPING-OVERRIDES={SAME-AS-MAPPING-LEVEL|UNDERSCORES-AS-HYPHENS|LESS-DUP-NAMES}

언어 구조를 생성할 때 지정된 맵핑 레벨에 대해 기본 동작이 대체되는지 여부를 지정합니다.

LESS-DUP-NAMES

이 매개변수는 이름 끝에 `_value`가 있는 비구조적 구조 필드 이름을 생성하여 필드를 직접 참조할 수 있습니다. 예를 들어, 다음 PLI 언어 구조에서 **MAPPING-OVERRIDES=LESS-DUP-NAMES**가 지정되는 경우 레벨 12 필드 `streetName`에는 접미부 `_value`가 추가됩니다.

```
09 streetName,
   12 streetName                CHAR(255) VARYING
   UNALIGNED,
   12 filler                    BIT (7),
   12 attr_nil_streetName_value BIT (1),
```

결과적으로 구조는 다음과 같습니다.

```
09 streetName,
   12 streetName_value          CHAR(255) VARYING
   UNALIGNED,
   12 filler                    BIT (7),
   12 attr_nil_streetName_value BIT (1),
```

SAME-AS-MAPPING-LEVEL

이 매개변수는 맵핑 레벨과 동일한 스타일의 언어 구조를 생성합니다. 이는 기본값입니다.

UNDERSCORES-AS-HYPHENS

COBOL에서만 사용됩니다. 이 매개변수는 JSON 스키마의 밑줄을 문자 `X`가 아닌 하이픈으로 변환하여 생성된 COBOL 언어 구조의 가독성을 향상시킵니다. 필드 이름 충돌이 발생하면 필드가 고유할 수 있도록 필드에 번호가 지정됩니다. 자세한 정보는 개발 애플리케이션에서 COBOL에 대한 JSON 스키마 맵핑을 참조하십시오.

MINIMUM - RUNTIME -

LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0|CURRENT}

웹 서비스 바인딩 파일이 배치될 수 있는 최소 CICS 런타임 환경을 지정합니다. 지정한 다른 매개변수와 일치하지 않는 레벨을 선택하면 오류 메시지를 받게 됩니다. 다음 옵션을 선택할 수 있습니다.

MINIMUM

선택한 매개변수의 경우 CICS의 런타임 레벨 중 가장 낮은 레벨이 자동으로 할당됩니다.

- 1.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 3.0 생성된 웹 서비스 바인딩 파일이 CICS TS 4.1 이상 리전에 배치됩니다.

참고: JSON 지원은 CICS TS 4.2 이상에서만 사용할 수 있습니다.

- 4.0 생성된 웹 서비스 바인딩 파일이 CICS TS 5.2 이상 리전에 배치됩니다. 이 런타임 레벨에서는 **MAPPING-LEVEL** 매개변수에 4.0 이하 맵핑 레벨을 사용할 수 있습니다. 이 레벨에서는 선택적 매개변수를 사용할 수 있습니다.

CURRENT

생성된 웹 서비스 바인딩 파일이 웹 서비스 바인딩 파일을 생성하기 위해 사용하는 런타임 레벨과 동일한 런타임 레벨에서 CICS 리전에 배치됩니다.

NAME-TRUNCATION={LEFT|RIGHT}

JSON 이름이 왼쪽 또는 오른쪽부터 잘리는지 여부를 지정합니다. CICS 웹 서비스 지원은 지정된 고급 언어에 적합한 길이로 JSON 이름을 자릅니다. 기본적으로 이름은 오른쪽부터 잘립니다.

PDSCP=value

PDSMEM 매개변수에 지정된 파티션된 데이터 세트 멤버에서 사용되는 코드를 지정

합니다. 여기서 *value*는 CCSID 번호 또는 Java 코드 페이지 번호입니다. 이 매개 변수가 지정되지 않으면 z/OS UNIX System Services 코드 페이지를 사용합니다. 예를 들어, **PDSCP=037**을 지정할 수 있습니다.

PDSLIB=*value*

생성된 고급 언어를 포함하는 파티션된 데이터 세트의 이름을 지정합니다. 요청 및 응답에 사용되는 데이터 세트 멤버는 **PDSMEM** 매개변수에 지정됩니다.

PDSMEM=*value*

DFHJS2LS가 고급 언어 구조를 포함할 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 사용하는 1 - 6자 접두부를 지정합니다.

DFHJS2LS는 접두부에 01을 추가하여 파티션된 데이터 세트 멤버를 생성합니다.

PGMINT=CHANNEL|COMMAREA

서비스 제공자의 경우, CICS가 대상 애플리케이션 프로그램으로 데이터를 전달하는 방식을 지정합니다.

CHANNEL

CICS가 채널 인터페이스를 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

COMMAREA

CICS가 통신 영역을 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

대상 애플리케이션 프로그램이 요청을 처리한 경우에는 동일한 메커니즘으로 응답을 리턴해야 합니다. 즉, 요청이 통신 영역에서 수신된 경우에는 응답도 통신 영역에서 리턴되어야 하고 요청이 컨테이너에서 수신된 경우에는 응답도 컨테이너에서 리턴되어야 합니다. CICS가 대상 애플리케이션 프로그램으로 전달하는 통신 영역 또는 컨테이너의 길이는 요청 통신 영역 또는 컨테이너의 길이와 응답 통신 영역 또는 컨테이너의 길이 중 긴 값입니다.

PGMNAME=*value*

CICS PROGRAM 자원의 이름을 지정합니다.

DFHJS2LS가 서비스 제공자에서 사용되는 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우 이 매개변수를 제공해야 합니다. 이 매개변수는 웹 서비스로 노출되는 애플리케이션 프로그램의 자원 이름을 지정합니다.

DFHJS2LS가 서비스 요청자에서 사용되는 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우에는 이 매개변수를 생략하십시오.

STRUCTURE=*name*

C 및 C++의 경우에만 구조 이름이 생성되는 방식을 지정합니다.

생성된 구조에는 이름 *name01* 이 지정됩니다.

이름이 생략되는 경우, 구조 이름은 사용자가 지정하는 **PDSMEM** 매개변수에서 생성되는 파티션된 데이터 세트 멤버 이름과 같습니다.

SYNCONRETURN=**{NO|YES}**

원격 웹 서비스가 동기점을 발행할 수 있는지 여부를 지정합니다.

NO 원격 웹 서비스가 동기점을 발행할 수 없습니다. 이 값은 기본값입니다. 원격 웹 서비스가 동기점을 발행하면 ADPL 이상 종료와 함께 실패합니다.

YES 원격 웹 서비스가 동기점을 발행할 수 있습니다. YES를 선택하면 원격 웹 서비스에서 제어가 리턴되는 경우 원격 태스크가 개별 작업 단위로 커밋됩니다. 원격 웹 서비스가 복구 가능 자원을 갱신하고 리턴 후 실패하면 해당 자원에 대한 갱신을 백아웃할 수 없습니다.

TRANSACTION=*name*

서비스 제공자에서 이 매개변수는 파이프라인을 시작할 수 있는 별명 트랜잭션의 1 - 4자 이름을 지정합니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 TRANSACTION 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ # _ < >

URI=*value*

서비스 제공자에서, 이 매개변수는 클라이언트가 웹 서비스에 액세스하기 위해 사용하는 상대 URI를 지정합니다. CICS는 DFHJS2LS로 작성된 웹 서비스 바인딩 파일에서 URIMAP 자원을 생성할 때 지정되는 값을 사용합니다. 이 매개변수는 URIMAP 정의가 적용되는 URI의 경로 컴포넌트를 지정합니다. URI 끝에 와일드카드 *를 사용하는 경우에는 URI 값 뒤에 쉼표가 와야 합니다.

USERID=*id*

서비스 제공자에서 이 매개변수는 웹 클라이언트가 사용할 수 있는 1 - 8자 사용자 ID를 지정합니다. 애플리케이션 생성 응답 또는 웹 서비스의 경우 이 사용자 ID 아래에 별명 트랜잭션이 첨부됩니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 때 URIMAP 자원의 USERID 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3=**NO|YES**

COBOL에서만 사용됩니다. COBOL 언어 구조에서 팩형 10진수 가변 길이의 최대 크기를 제어합니다.

NO DFHJS2LS가 COBOL 언어 구조 유형 COMP-3을 생성할 때 팩형 10진

수 가변 길이를 18로 제한합니다. 팩형 10진수 크기가 18보다 큰 경우, 지정된 유형이 총 18자로 제한됨을 나타내는 DFHPI9022W 메시지가 발행됩니다.

YES DFHJS2LS가 COBOL 언어 구조 유형 COMP-3을 생성할 때 최대 크기 31을 지원합니다.

WSBIND=*value*

웹 서비스 바인딩 파일의 완전한 z/OS UNIX 이름입니다. DFHJS2LS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 기본 파일 확장자는 .wsbind입니다.

기타 정보

- DFHJS2LS가 실행되는 사용자 ID는 UNIX System Services를 사용하도록 구성되어야 합니다. 사용자 ID는 CICS z/OS UNIX 파일 구조 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND**, **WSDL** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다.
- JCL의 최대 매개변수 길이는 100자입니다. 이 길이는 **STDPARM**문을 사용하여 늘릴 수 있습니다. 자세한 정보는 *z/OS UNIX System Services User Guide*를 참조하십시오.

예제

```
//JS2LS JOB 'accounting information',name,MSGCLASS=A
// SET QT='''
//JAVAPROG EXEC DFHJS2LS,
// TMPFILE=&QT.&SYSUID.&QT
/INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
PDSMEM=CPYBK2
JSON-SCHEMA-RESTFUL=example.json
LANG=COBOL
LOGFILE=/u/exampleapp/wsbind/example.log
MAPPING-LEVEL=4.0
CHAR-VARYING=NULL
INLINE-MAXOCCURS-LIMIT=2
PGMNAME=DFH0XCMN
URI=exampleApp/example/*,
PGMINT=COMMAREA
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbind/example.wsbind
/*
```

JSON 지원을 사용하여 JSON 서비스 제공자 애플리케이션 작성

JSON 스키마 v4(초안)를 준수하는 JSON 스키마 또는 고급 언어 데이터 구조에서 서비스 제공자 애플리케이션을 작성할 수 있습니다. CICS JSON 지원은 서비스 제공자 설정에서 CICS 애플리케이션을 배치하는 데 도움을 줍니다.

이 태스크 정보

지원을 사용하여 CICS 애플리케이션을 서비스 제공자로 배치할 때 다음 두 가지 옵션이 있습니다.

- JSON 스키마를 시작하고 지원을 사용하여 언어 데이터 구조를 생성합니다.

기존 웹 서비스 설명에 부합되는 서비스 제공자를 구현하려는 경우에는 이 옵션을 사용하십시오.

- 언어 데이터 구조를 시작하고 지원을 사용하여 JSON 스키마를 생성합니다.

기존 프로그램을 JSON 서비스로 노출시키려는 경우에는 이 옵션을 사용하십시오.

관련 정보:

 JSON 스키마 v4(초안)

JSON 스키마에서 서비스 제공자 애플리케이션 작성:

CICS JSON 지원을 사용하여 JSON 스키마로부터 서비스 제공자 애플리케이션을 작성할 수 있습니다.

시작하기 전에

서비스 제공자 애플리케이션을 작성하려면 먼저 다음 조건이 만족되어야 합니다.

- 웹 서비스 설명이 z/OS의 UNIX 파일에 있어야 하고 CICS 리전에서 적절한 제공자 모드 파이프라인을 작성해야 합니다.
- OMVS에 DFHJS2LS를 실행하는 사용자 ID를 정의해야 합니다.
- 사용자 ID는 z/OS UNIX 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND** 및 **JSON-SCHEMA-REQUEST**와 **JSON-SCHEMA-RESPONSE** 및 **JSON-SCHEMA-RESTFUL** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- Java를 실행할 수 있도록 사용자 ID에 충분한 저장영역을 할당해야 합니다. 지원되는 모든 Java 버전을 사용할 수 있습니다. 기본적으로 DFHJS2LS는 **JAVADIR** 매개변수에 지정된 Java 버전을 사용합니다.

이 태스크 정보

JSON 지원을 사용하여 서비스 제공자 애플리케이션의 JSON 스키마에서 언어 구조를 작성할 수 있습니다.

프로시저

1. DFHJS2LS 일괄처리 프로그램을 사용하여 웹 서비스 바인딩 파일 및 한 개 이상의 언어 데이터 구조를 생성하십시오. DFHJS2LS에는 사용자 애플리케이션이 필요로 하는 바인딩 파일 및 언어 구조를 작성할 수 있도록 사용자에게 유연성을 제공

하는 대형 선택적 매개변수 세트가 포함되어 있습니다. 웹 서비스에 대해 기존 애플리케이션을 사용으로 설정할 때 이러한 옵션을 고려하십시오.

- 데이터를 서비스 제공자 애플리케이션 프로그램에 전달하기 위해 CICS가 사용하는 메커니즘은? 채널을 사용하여 컨테이너로 데이터를 전달하거나 COMMAREA를 사용할 수 있습니다. 채널 및 컨테이너가 권장됩니다. 이를 **PGMINT** 매개변수로 지정하십시오.
- 생성할 언어는? DFHJS2LS는 COBOL, C/C++ 또는 PL/I 언어 데이터 구조를 생성할 수 있습니다. **LANG** 매개변수를 사용하여 언어를 지정하십시오.
- 사용할 맵핑 레벨은? 맵핑 레벨이 높을수록 런타임 시 문자 및 2진 데이터의 헵들링에 사용 가능한 제어 및 지원이 많아집니다. 일부 선택적 매개변수는 높은 맵핑 레벨에서만 사용 가능합니다. 사용 가능한 가장 높은 레벨의 맵핑을 사용하도록 권장됩니다. **MAPPING-LEVEL** 매개변수로 맵핑 레벨을 지정하십시오.
- 웹 서비스 요청자가 사용하게 할 URI는? **URI** 매개변수를 사용하여 상대 URI를 지정하십시오(예: URI=/my/test/webservice). 이 값은 URIMAP 자원을 작성할 때 CICS에서 사용됩니다.
- 웹 서비스 요청 및 응답을 실행할 트랜잭션 및 사용자 ID는? 서비스 요청자에 대한 응답을 구성하기 위해 별명 트랜잭션을 사용하여 애플리케이션을 실행할 수 있습니다. 별명 트랜잭션은 사용자 ID 아래 첨부됩니다. 이를 **TRANSACTION** 및 **USERID** 매개변수로 지정하십시오. 이러한 값은 URIMAP 자원을 작성할 때 사용됩니다. 특정 트랜잭션을 사용하지 않으려는 경우에는 이러한 매개변수를 사용하지 마십시오.

DFHJS2LS를 제출할 때 CICS는 웹 서비스 바인딩 파일을 생성하고 이 파일을 **WSBIND** 매개변수로 지정한 위치에 배치합니다. 언어 구조는 **PDSLIB** 매개변수로 지정한 파티션된 데이터 세트에 배치됩니다.

2. 생성된 웹 서비스 바인딩 파일을 웹 서비스 애플리케이션에 사용할 제공자 모드 PIPELINE 자원의 pickup 디렉토리에 복사하십시오. 바인딩 파일은 2진 모드로 복사해야 합니다.
3. 생성된 언어 구조와 인터페이스할 서비스 제공자 애플리케이션 프로그램을 작성하고 필수 비즈니스 로직을 구현하십시오.
4. **PIPELINE SCAN** 명령을 사용하여 WEBSERVICE 자원 및 URIMAP 자원을 동적으로 작성하십시오.
 - WEBSERVICE 자원은 CICS의 웹 서비스 바인딩 파일을 요약하고 런타임 시 사용됩니다.
 - URIMAP 자원은 CICS에 WEBSERVICE 자원과 특정 URI를 연관시키기 위한 정보를 제공합니다.

또는 직접 자원을 정의할 수 있지만 권장되지 않습니다.

결과

DFHJS2LS 제출에 문제점이 있거나 자원이 올바르게 설치되지 않는 경우 문제점 해결에서 JSON 지원자 문제점 해결의 내용을 참조하십시오.

 JSON 스키마 v4

 JSON 스키마 유효성 검증 v0

데이터 구조에서 서비스 제공자 애플리케이션 작성:

CICS 웹 서비스 지원을 사용하여 고급 언어 데이터 구조에서 서비스 제공자 애플리케이션을 작성할 수 있습니다.

시작하기 전에

서비스 제공자 애플리케이션을 작성하기 전에 설정이 다음 전제조건을 준수하는지 확인하십시오.

- 고급 언어 데이터 구조가 다음 기준을 충족시켜야 합니다.
 - 데이터 구조는 소스 프로그램(예: COBOL 카피북)과 별도로 정의되어야 합니다.
 - PL/I 또는 COBOL 애플리케이션 프로그램이 입력 및 출력에 서로 다른 데이터 구조를 사용하는 경우 데이터 구조는 파티션된 데이터 세트에 있는 두 개의 다른 멤버에서 정의되어야 합니다. 입력 및 출력에 같은 구조를 사용하는 경우 이 구조는 단일 멤버에서 정의되어야 합니다.

C 및 C++에서 데이터 구조는 파티션된 데이터 세트에서 같은 멤버에 있을 수 있습니다.

- 언어 구조는 파티션된 데이터 세트에서 사용 가능해야 하며 CICS 리전에서 적절한 PIPELINE 자원을 작성해야 합니다.
- OMVS에 DFHLS2WS가 실행에 사용하는 사용자 ID를 정의해야 합니다.
- 사용자 ID는 z/OS UNIX 및 PDS 라이브러리에 대한 읽기 권한 및 **LOGFILE**, **WSBIND** 및 **JSON-SCHEMA-REQUEST**와 **JSON-SCHEMA-RESPONSE** 출력 매개변수에서 지정되는 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다. 지원되는 모든 Java 버전을 사용할 수 있습니다. 기본적으로 DFHLS2JS는 **JAVADIR** 매개변수에 지정된 Java 버전을 사용합니다.

프로시저

상위 레벨 데이터 구조에서 서비스 제공자 애플리케이션을 작성하려면 다음 단계를 따르십시오.

1. 서비스 제공자 애플리케이션 인터페이스에서 채널 및 다수의 컨테이너를 사용하는 경우 JSON로 인터페이스를 설명하는 채널 설명 문서를 작성하십시오. 채널 설명 문서는 z/OS UNIX의 적절한 디렉토리에 저장해야 합니다. CICS는 이 문서를 사용하여 채널의 여러 컨테이너에서 JSON 메시지를 구성하고 구성 해제합니다. 또는 한 채널에서 한 개의 컨테이너를 사용하고 채널 설명 문서를 작성하지 않을 수 있습니다.

채널 설명 문서를 작성하는 방법에 대한 자세한 정보는 221 페이지의 『채널 설명 문서 작성』의 내용을 참조하십시오.

2. DFHLS2JS 일괄처리 프로그램을 사용하여 언어 구조에서 웹 서비스 설명 및 웹 서비스 바인딩 파일을 생성하십시오. DFHLS2JS 일괄처리 프로그램은 *HLQ.XDFHINST* 에서 찾을 수 있으며 여기서 *HLQ*는 CICS를 설치한 위치입니다. DFHLS2JS에는 사용자 애플리케이션이 필요로 하는 바인딩 파일 및 언어 구조를 작성할 수 있도록 사용자에게 유연성을 제공하는 대형 선택적 매개변수 세트가 포함되어 있습니다. 기존 애플리케이션에 대해 웹 서비스를 사용으로 설정할 때 다음 옵션을 고려하십시오.

- 데이터를 서비스 제공자 애플리케이션 프로그램에 전달하기 위해 CICS가 사용하게 하려는 메커니즘은? 채널을 사용하여 컨테이너로 데이터를 전달하거나 COMMAREA를 사용할 수 있습니다. **PGMINT** 매개변수를 사용하여 메커니즘을 지정하십시오. 애플리케이션 인터페이스가 채널 및 다수의 컨테이너를 사용하는 경우 **REQUEST-CHANNEL** 매개변수 및 선택적으로 **RESPONSE-CHANNEL**을 지정하십시오. 이러한 매개변수는 맵핑 레벨이 3.0 이상인 경우에만 사용할 수 있습니다.
- 사용할 맵핑 레벨은? 맵핑 레벨이 높을수록 런타임 시 문자 및 2진 데이터의 핸들링에 사용 가능한 제어 및 지원이 많아집니다. 일부 선택적 매개변수는 높은 맵핑 레벨에서만 사용 가능합니다. **MAPPING-LEVEL** 매개변수에 사용 가능한 가장 높은 레벨의 맵핑을 지정해야 합니다.
- 웹 서비스가 사용하게 할 URI는? **URI** 매개변수를 사용하여 절대 URI를 지정하십시오(예: **URI**=http://www.example.org:80/my/test/webservice). 이 주소의 상대 파트인 /my/test/webservice는 URIMAP 자원을 작성할 때 사용됩니다.

DFHLS2JS를 제출할 때 CICS는 웹 서비스 바인딩 파일을 생성하고 이 파일을 **WSBIND** 매개변수로 지정한 위치에 배치합니다. 생성된 JSON 스키마는 **JSON-SCHEMA-REQUEST** 및 **JSON-SCHEMA-RESPONSE** 매개변수로 지정된 위치에 배치됩니다.

3. 생성된 JSON 스키마를 검토하십시오.

이러한 스키마는 JSON 웹 서비스의 입력 및 출력 데이터 형식을 정의하는 데 사용됩니다. 애플리케이션 개발자는 JSON 웹 서비스와 상호작용하기 위해 애플리케이션을 작성할 때 이러한 스키마를 사용해야 합니다.

참고: 생성된 스키마를 변경하면 생성된 웹 서비스 바인딩 파일 WSBind를 무효화합니다.

스키마를 변경하려는 경우(예: 스키마 내에서 필드의 이름 바꾸기) DFHJS2LS를 사용하여 새 웹 서비스 바인딩 파일 및 새 언어 구조 세트를 생성해야 합니다. 새 언어 구조를 사용하려면 CICS의 애플리케이션 프로그램을 변경해야 합니다.

4. 웹 서비스 바인딩 파일을 웹 서비스 애플리케이션에 사용할 제공자 모드 파이프라인의 픽업 디렉토리에 복사하십시오. 웹 서비스 바인딩 파일은 2진 모드로 복사해야 합니다.
5. **PIPELINE SCAN** 명령을 사용하여 WEBSERVICE 자원 및 URIMAP 자원을 동적으로 작성하십시오.
 - WEBSERVICE 자원은 CICS의 웹 서비스 바인딩 파일을 포함하고 런타임 시 사용됩니다.
 - URIMAP 자원은 CICS에 WEBSERVICE 자원과 특정 URI를 연관시키기 위한 정보를 제공합니다.

또는 직접 자원을 정의할 수 있습니다.

결과

서비스 제공자 애플리케이션의 작성이 완료됩니다.

DFHLS2JS 제출에 문제점이 있거나 자원이 올바르게 설치되지 않는 경우 문제점 해결에서 JSON 지원자 문제점 해결의 내용을 참조하십시오.

다음에 수행할 작업

서비스에 액세스할 웹 서비스를 개발하는 모든 사용자가 사용할 수 있도록 웹 서비스 설명을 공개하십시오.

채널 설명 문서 작성:

서비스 제공자 애플리케이션이 여러 컨테이너와의 채널 인터페이스를 사용하는 경우 채널 설명 문서를 작성합니다.

이 태스크 정보

XML 편집기를 사용하여 채널 설명 문서를 작성하십시오. 채널 설명에 대한 스키마는 channel.xsd라고 하며 /usr/lpp/cicsts/cicsts53/schemas/channel 디렉토리(여

가서, /usr/lpp/cicsts/cicsts53는 z/OS UNIX에서 CICS 파일의 기본 설치 디렉토리임)에 있습니다.

프로시저

1. <채널> 요소 및 CICS 채널 네임스페이스를 사용하여 XML 문서를 작성하십시오.

```
<channel name="myChannel" xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```

2. 애플리케이션 프로그램 인터페이스가 채널에서 사용하는 모든 컨테이너에 <컨테이너> 요소를 추가하십시오. 이름, 유형 및 속성을 사용하여 각 컨테이너를 설명해야 합니다. 다음 예제는 서로 다른 속성 값을 가지는 여섯 개의 컨테이너를 보여줍니다.

```
<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>
```

이 구조 요소는 파티션된 데이터 세트 멤버에 있는 언어 구조로 콘텐츠가 정의됨을 표시합니다.

3. z/OS UNIX에서 XML 문서를 저장하십시오.

채널 스키마

채널 설명 문서는 다음 스키마를 준수해야 합니다.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.ibm.com/xmlns/prod/CICS/channel"
  xmlns:tns="http://www.ibm.com/xmlns/prod/CICS/channel" elementFormDefault="qualified">
  <element name="channel"> 1
    <complexType>
      <sequence>
        <element name="container" maxOccurs="unbounded" "unbounded" minOccurs="0"> 2
          <complexType>
            <sequence>
              <element name="structure" minOccurs="0"> 3
                <complexType>
                  <attribute name="location" type="string" use="required"/>
                  <attribute name="structure" type="string" use="optional"/>
                </complexType>
              </element>
            </sequence>
            <attribute name="name" type="tns:name16Type" use="required"/>
            <attribute name="type" type="tns:typeType" use="required"/>
            <attribute name="use" type="tns:useType" use="required"/>
          </complexType>
        </element>
      </sequence>
      <attribute name="name" type="tns:name16Type" use="optional" />
    </complexType>
  </element>
</simpleType name="name16Type">
```

```

        <restriction base="string">
            <maxLength value="16"/>
        </restriction>
    </simpleType>
    <simpleType name="typeType">
        <restriction base="string">
            <enumeration value="char"/>
            <enumeration value="bit"/>
        </restriction>
    </simpleType>
    <simpleType name="useType">
        <restriction base="string">
            <enumeration value="required"/>
            <enumeration value="optional"/>
        </restriction>
    </simpleType>
</schema>

```

1. 이 요소는 CICS 채널을 나타냅니다.
2. 이 요소는 채널 내 CICS 컨테이너를 나타냅니다.
3. 구조는 '비트' 모드 컨테이너와 함께만 사용할 수 있습니다. '위치' 속성은 컨테이너의 콘텐츠를 맵핑하는 파일의 위치를 표시합니다. '구조' 속성은 C 및 C++에서 사용되어 구조의 이름을 표시할 수 있습니다.

다음에 수행할 작업

DFHLS2JS를 실행하여 웹 서비스 제공자 애플리케이션에 대한 맵핑 및 JSON 스키마를 작성하십시오. DFHLS2JS는 채널 설명 문서에서 컨테이너가 지정되는 순서대로 채널의 맵핑을 JSON 스키마에 넣습니다.

생성된 JSON 스키마 사용자 정의:

DFHLS2JS에 의해 생성되는 JSON 스키마에는 공개 전 변경하기에 적합할 수 있는 일부 자동 생성 콘텐츠가 있습니다. JSON 스키마를 사용자 정의하면 웹 서비스 바인딩 파일이 재생성될 수 있으며 일부 경우에는 래퍼 프로그램이 작성될 수 있습니다.

이 태스크 정보

다음 단계에 따라 생성된 JSON 스키마를 사용자 정의하십시오.

프로시저

1. HTTPS 지원을 광고하려면 DFHLS2JS의 **URI** 매개변수를 사용하여 절대 URI를 설정하십시오.
2. 웹 서비스의 네트워크 위치를 제공하려면 DFHLS2JS의 **URI** 매개변수를 사용하여 절대 URI를 설정하십시오.
3. JSON 스키마에서 자동으로 생성된 이름이 사용자의 용도에 적합한지 여부를 고려해 보십시오. 특성의 이름을 바꿀 수 있습니다.

이러한 값은 클라이언트 프로그램을 코드화할 프로그래밍 방식 인터페이스 일부를 구성합니다. 생성된 이름의 의미가 분명하지 않으면 장기간에 걸친 애플리케이션 코드의 유지보수가 좀 더 어려울 수 있습니다.

이러한 값을 변경하는 경우 DFHJS2LS를 사용하여 웹 서비스 바인딩 파일을 재생성해야 합니다. 생성되는 언어 구조는 기존 애플리케이션과 호환 가능하지 않을 가능성이 높기 때문에 애플리케이션 변경이 필요할 수 있습니다. 생성된 언어 구조를 검토하고 4 단계에서 논의한 대로 랩퍼 프로그램 작성을 고려해 보십시오.

4. JSON 스키마에서 노출된 COMMAREA 필드가 적합한지 고려하십시오. JSON 클라이언트 개발자에게 유용하지 않은 필드 제거를 고려할 수 있습니다.

- 출력 값에만 사용되는 필드는 입력 데이터 구조를 맵핑하는 스키마에서 제거할 수 있습니다.
- 필터 필드입니다.
- 자동으로 생성되는 어노테이션입니다.

이러한 변경사항을 작성하는 경우 DFHJS2LS를 사용하여 웹 서비스 바인딩 파일을 재생성해야 합니다. 생성되는 새 언어 구조가 원래 언어 구조와 호환 가능하지 않기 때문에 랩퍼 프로그램을 작성하여 새 표시에서 이전 표시로 데이터를 맵핑해야 합니다. 이 랩퍼 프로그램은 대상 애플리케이션 프로그램에 대해 **EXEC CICS LINK** 명령을 수행한 다음 리턴된 데이터를 맵핑해야 합니다.

이 사용자 정의 레벨에서는 가장 많은 노력이 필요하지만 JSON 클라이언트 개발자에게 가장 유용한 프로그래밍 방식 인터페이스가 생성됩니다.

결과

비즈니스 요구사항에 맞게 사용자 정의된 JSON 스키마 및 CICS에 비즈니스 요구사항을 구현하는 프로그램을 가지고 있습니다.

RESTful 웹 서비스 제공자 애플리케이션 작성:

RESTful JSON 웹 서비스의 CICS 구현은 SOAP 웹 서비스의 CICS 구현과 유사합니다. 대부분의 개념 및 아키텍처는 공유되지만 CICS에서는 JSON 스키마 사용이 필요합니다.

이 태스크 정보

RESTful JSON 웹 서비스 구현에는 다음 태스크가 포함됩니다.

프로시저

1. 애플리케이션 인터페이스를 생성하십시오.

입력:

- RESTful 웹 서비스에 대한 데이터 모델을 정의하는 JSON 스키마

출력:

- JSON 스키마를 맵핑하는 언어 구조(예: COBOL 카피북)
- WSBind 파일

DFHJS2LS 유틸리티를 실행하고 적절한 입력 매개변수를 지정하십시오. 이러한 매개변수는 다음과 같습니다.

- JSON 스키마의 위치
- 지원되는 메소드 목록(GET, PUT, POST 및 DELETE는 기본적으로 사용으로 설정됨)
- 서비스가 배치되는 URI
- 서비스를 구현하는 애플리케이션 PROGRAM의 이름
- 필수 데이터 맵핑 매개변수

2. 이 인터페이스를 사용하는 애플리케이션을 작성하십시오.

입력:

- 224 페이지의 1 단계의 언어 구조
- 구현할 RESTful 조작에 대한 인식

출력:

- CICS에 대한 배치에 적합한 프로그램

다음은 수행하는 프로그램을 작성하십시오.

참고: DFHJS2LS에 대한 **CONTID** 매개변수에 자체 컨테이너의 이름을 제공하는 경우 다음 단계에서 언급될 때마다 DFHWS-DATA 대신 이 컨테이너를 사용해야 합니다.

- URI를 검사하여 자원의 ID를 확인하십시오. CICS는 URI의 관련 컴포넌트를 식별하는 데 도움이 되는 여러 컨테이너를 제공합니다. 이러한 컨테이너는 표 2에 설명되어 있습니다. 다음 예제는 URI에 대한 각 컨테이너의 콘텐츠를 보여줍니다.

<http://www.example.org:10000/JSONServices/CustomDetails/13388?action=query>

일치한 URIMAP의 경로가 /JSONServices/CustomDetails/*인 경우

표 2. DFHWS-URI 컨테이너. DFHWS-URI 컨테이너

컨테이너	콘텐츠	예제
구성에서 DFHWS-URI	전체 URI	http://www.example.org:10000/JSONServices/CustomDetails/13388?action=query

표 2. DFHWS-URI 컨테이너 (계속). DFHWS-URI 컨테이너

컨테이너	컨텐츠	예제
	URIMAP과 일치한 URI 경로 부분	/JSONServices/ CustomerDetails/*
구성에서 DFHWS-URI-RESID	제거된 URIMAP과 일치하는 부분이 포함된 URI 경로(자원 ID)	13388
구성에서 DFHWS-URI-QUERY	조회 문자열	action=query

- b. URI를 유효성 검증하십시오. 문제점이 있을 경우 CICS에 문제점을 보고하고 종료하십시오.
- c. DFHHTTPMETHOD 컨테이너를 조회하여 구동되고 있는 메소드를 판별하십시오. 자세한 정보는 구성에서 DFHHTTPMETHOD의 내용을 참조하십시오.
- d. POST(작성) 메소드(필요한 경우)의 경우:
 - DFHWS-DATA 컨테이너에서 입력 데이터를 읽으십시오.
 - 224 페이지의 1단계에서 생성된 언어 구조를 사용하여 데이터를 해석하십시오.
 - 데이터를 유효성 검증하십시오. 문제점이 있을 경우 CICS에 문제점을 보고하고 종료하십시오.
 - 자원 작성에 필요한 애플리케이션 특정 처리를 수행하십시오.
 - 선택적으로 DFHRESPONSE 컨테이너를 작성하여 클라이언트에게 새 자원의 ID를 알리십시오. 이 컨테이너의 컨텐츠는 CICS에 의해 변환되는 것이 아니라 HTTP 응답으로 직접 전송됩니다. DFHWS-DATA 컨테이너는 무시됩니다.
- e. GET(조회) 또는 HEAD(조회) 메소드가 필요한 경우 *resource*를 나타내는 데이터를 DFHWS-DATA 컨테이너에 작성하십시오.
- f. PUT(설정) 메소드가 필요한 경우:
 - DFHWS-DATA 컨테이너에서 입력 데이터를 읽으십시오.
 - 224 페이지의 1단계에서 생성된 언어 구조를 사용하여 데이터를 해석하십시오.
 - 데이터를 유효성 검증하십시오. 문제점이 있을 경우 CICS에 문제점을 보고하고 종료하십시오.
 - 자원 갱신에 필요한 애플리케이션 특정 처리를 수행하십시오.
- g. DELETE 메소드가 필요하면 자원을 삭제하는 데 필요한 애플리케이션 특정 처리를 수행하십시오.

참고: CICS에 의해 구현되는 RESTful 데이터 모델은 기본적으로 PUT, POST 또는 DELETE 메소드에 대해 응답 본문을 전송하지 않습니다. RESTful 애플리케이션은 일반적으로 HTTP 상태 코드를 사용하여 성공 또는 실패를 나타내십시오. 애플리케이션이 정상적으로 완료되면 CICS가 HTTP 응답 200(OK)을 전송합니다. 오

류 응답 전송에 대한 자세한 정보는 애플리케이션 개발에서 애플리케이션 오류 보고의 내용을 참조하십시오. PUT, POST 또는 DELETE 메소드에 대해 응답 본문을 전송하려는 경우 DFHRESPONSE 컨테이너를 작성해야 합니다. 있는 경우 CICS는 추가 처리 없이 HTTP 본문으로 이 컨테이너의 콘텐츠를 전송합니다. CICS는 이러한 메소드에 대한 응답 처리에서 DFHWS-DATA 컨테이너를 무시합니다.

3. 아티팩트를 배치하십시오.

입력:

- 224 페이지의 1 단계의 WSBind 파일
- 225 페이지의 2 단계의 CICS 프로그램
- JSON에 대해 구성되는 CICS 제공자 모드 파이프라인 자원

출력:

- 배치된 RESTful JSON 웹 서비스

일반적인 방식으로 CICS에 프로그램을 배치하십시오.

다음 중 하나를 수행하십시오.

- WSBind 파일을 파이프라인의 '픽업' 디렉토리에 배치한 다음 **PIPELINE SCAN** 명령을 발행하여 WEBSERVICE 및 URIMAP 자원을 작성하십시오.
- WEBSERVICE 자원 및 연관된 URIMAP 자원을 수동으로 정의하고 설치하십시오. URIMAP은 URI를 PIPELINE 및 WEBSERVICE 둘 다와 연관시켜야 합니다.

4. 서비스를 테스트하십시오.

입력:

- JSON 스키마
- RESTful 웹 서비스의 URI
- 3단계에서 배치된 서비스
- 선택한 JSON 테스트 클라이언트

출력:

- 성공적으로 처리한 요청

선택한 테스트 클라이언트를 사용하여 CICS에 JSON 요청을 전송하십시오.

예상치 못한 응답을 수신하는 경우 문제점 판별을 시도하십시오. 자세한 정보는 JSON 요청 문제점 해결의 내용을 참조하십시오.

관련 정보:

 RESTful 서비스

RESTful 웹 서비스 제공자 애플리케이션에 대한 디자인 고려사항:

이 주제에서는 JSON용 RESTful 웹 서비스 제공자 애플리케이션을 계획 및 디자인할 때 고려해야 할 몇 가지 문제를 설명합니다.

자원의 컬렉션

RESTful API를 위한 공통 디자인은 자원의 컬렉션 검색을 지원하는 것입니다. 예를 들어, 다음과 같이 일련의 오브젝트를 리턴하는 서비스가 있을 수 있습니다.

```
GET /Services/CustomerDetails?Surname=Cooper
```

이 요청은 성이 "Cooper"인 모든 CustomerDetails 오브젝트에 대한 정보를 리턴할 것으로 예상됩니다. 다음과 같이 좀 더 구체적인 URI를 사용하면 개별 CustomerDetails 오브젝트가 리턴될 수도 있습니다.

```
GET /Services/CustomerDetails/Customer27
```

이 예제에서 Customer27은 특정 고객의 기본 키입니다. 이 두 번째 조회의 출력은 CustomerDetails 오브젝트의 인스턴스가 됩니다. 첫 번째 조회의 출력은 덜 명확합니다. 이 조회에서는 CustomerDetails 오브젝트의 목록을 리턴하거나 CustomerDetails 오브젝트에 대한 URI 목록(클라이언트가 개별 검색을 계속할 수 있음)을 리턴할 수 있습니다. 두 규칙 모두 공통입니다.

CICS에서 컬렉션을 구현하려면 데이터 인스턴스의 목록 또는 URI의 목록을 설명하는 JSON 스키마를 작성하십시오. 그런 다음 서비스를 빌드하고 평소와 같이 이 서비스를 구현할 수 있습니다. 이 예제에서 GET 메소드만 구현하도록 선택할 수 있습니다. 클라이언트가 전체 데이터 세트에서 페이지를 앞으로 넘기거나 뒤로 넘길 수 있도록 페이지 매기기 서비스 구현을 고려할 수 있습니다. 예를 들어, 다음과 같습니다.

```
GET /Services/CustomerDetails?startRecord=200&endRecord=225
```

CICS에서 두 개의 URIMAP 자원(및 두 개의 WEBSERVICE 자원)이 필요할 가능성이 높습니다. 컬렉션에 대해 루트 URI 구조를 맵핑하는 URIMAP 및 인스턴스에 대해 와일드카드화된 URIMAP. 예를 들어, 다음과 같습니다.

```
URIMAP1: Path=/Services/CustomerDetails WEBSERVICE=CollectionService
URIMAP2: Path=/Services/CustomerDetails/* WEBSERVICE=InstanceService
```

캐시 관리

RESTful 아키텍처는 표준 HTTP 캐시 관리 기법과의 통합을 권장합니다. 이러한 통합을 통해 GET 요청의 결과가 네트워크에서 캐시될 수 있기 때문에 서버에 대한 부담을 줄일 수 있습니다. 이를 수행하기 위한 메커니즘에는 GET 요청에 대해 리턴되는 데이터의 만기 날짜/시간 설정이 포함됩니다.

CICS에는 애플리케이션 제어 캐시 만기를 지원하기 위한 일반적인 용도의 메커니즘이 없지만 EXEC CICS WEB WRITE HTTPHEADER API를 사용하여 적합한 HTTP

헤더를 추가하도록 파이프라인 핸들러 프로그램을 작성할 수 있습니다. 애플리케이션 프로그램도 이와 유사한 작업을 수행할 수 있지만 HTTP 요청을 수신하는 같은 CICS 리전에서 호스팅되는 경우에만 해당됩니다.

애플리케이션 오류 보고:

이 주제를 읽고 RESTful JSON 웹 서비스 제공자 애플리케이션이 클라이언트에게 오류를 보고하는 방법을 이해하십시오.

하향식 시나리오에서는 애플리케이션이 오류 조건을 보고해야 할 가능성이 높습니다. 예를 들어, 오류는 『Account Number not recognized』일 수 있습니다. 이 요구사항은 하향식 시나리오에서 고유합니다. 상향식 개발에서는 애플리케이션에 데이터 필드에서 인코딩되는 오류 보고 메커니즘이 있거나 애플리케이션이 이상 종료됩니다. 하향식 시나리오에서 JSON 스키마는 오류를 보고하는 필드를 거의 정의하지 않기 때문에 대안이 필요합니다.

SOAP 기반 웹 서비스의 경우 **EXEC CICS SOAPFAULT** API를 사용하여 이 문제점을 처리합니다. SOAP 결함 메시지는 JSON에 존재하지 않습니다. 대신 DFHHTTPSTATUS 컨테이너를 사용하여 JSON 애플리케이션에 대한 애플리케이션 발견 오류를 보고할 수 있습니다. 자세한 정보는 구성에서 DFHHTTPSTATUS의 내용을 참조하십시오.

참고: 애플리케이션은 필요한 경우 DFHRESPONSE 컨테이너 또는 기타 제어 컨테이너를 사용하여 좀 더 세부적인 오류 응답을 제공할 수도 있습니다.

JSON 웹 서비스 제한사항

이 참조 자료를 사용하여 JSON 웹 서비스에서 지원되지 않는 기능을 이해할 수 있습니다.

다음 기능은 지원되지 않습니다.

- JSON이 포함된 요청자 모드 파이프라인은 지원되지 않습니다.
- 스키마에 대한 JSON 데이터의 런타임 유효성 검증이 지원되지 않습니다. JSON 페이로드와 함께 사용되는 WEBSERVICE 자원의 VALIDATION 속성 값은 무시됩니다.
- JSON 데이터(Badgerfish 또는 맵핑된 규칙)에서 네임스페이스 사용은 지원되지 않습니다.
- CICS에 전송된 JSON 페이로드는 UTF-8로 인코딩해야 합니다. 다른 인코딩은 지원되지 않습니다. 마찬가지로, CICS에 의해 전송된 JSON은 항상 UTF-8로 인코딩됩니다.
- JSON 파이프라인을 사용한 WebSphere MQ 전송은 지원되지 않습니다.
- 공급업체 변환기 프로그램을 JSON 변환기에 사용하는 것은 지원되지 않습니다.

- SOAP 웹 서비스 애플리케이션을 위해 JSON 파이프라인에서 작성된 WSBind 파일의 재사용은 지원되지 않습니다. JSON 서비스 제공자 애플리케이션에서 사용할 WSBind 파일은 JSON 지원에 의해 생성되어야 합니다.
- CICS가 JSON 페이로드를 변환할 때 JSON 페이로드가 일부 필수 콘텐츠를 잃는 경우 애플리케이션 프로그램에 전달 시 데이터 구조 내 해당 필드가 초기화되지 않습니다.
- CICS는 따옴표로 묶인 정수 값 외에 부호 있는 긴 정수($2^{63} - 1$)의 최대값을 초과하는 정수 값을 변환할 수 없습니다.
- 단순 데이터 유형 또는 배열의 사용은 JSON 스키마의 루트에서 지원되지 않습니다. JSON 오브젝트는 단순 데이터 유형 및 배열로 구성할 수 있지만 JSON 오브젝트를 설명하는 데 JSON 스키마가 필요합니다.
- maxItems 값이 1인 JSON 스키마에서 배열이 선언되는 경우 런타임 시 JSON을 생성할 때 CICS는 이 배열을 단순 문자열 또는 정수로 직렬화합니다.

중요사항: JSON 특성 이름에 대해 지원되는 유일한 문자는 첫 번째 문자에 대해서는 A-Z a-z _ : 이고 모든 후속 문자에 대해서는 A-Z a-z 0-9 _ : . -입니다.

현재 Axis2 웹 서비스 지원에는 애플리케이션 및 사용자 정의의 개발 및 배치를 위한 다수의 옵션이 있습니다. 다음 옵션은 지원되지 않습니다.

- 사용자 제공 애플리케이션 핸들러 - CICS 제공 애플리케이션 핸들러 클래스 `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler`를 사용해야 합니다.
- 사용자 작성 Axis2 Java 애플리케이션
- SOAPFAULT 및 WS-Addressing API는 JSON 파이프라인에서 사용할 수 없습니다.

컨테이너 제한사항

JSON 요청을 처리할 때 일부 파이프라인 컨테이너는 채워지지 않음에 유의하십시오. 자세한 정보는 구성의 파이프라인에서 사용된 컨테이너의 내용을 참조하십시오.

RESTful 웹 서비스의 차이점

INQUIRE PIPELINE에서

- SOAPLEVEL은 NOTSOAP을 리턴함
- MTOMNOXOPST, MTOMST, SENDMTOMST, SOAPRNUM, SOAPVNUM, XOPDIRECTST 및 XOPSUPPORTST 속성은 사용되지 않습니다.

INQUIRE WEBSERVICE에서

- ARCHIVEFILE, BINDING, VALIDATIONST, XOPDIRECTST 및 XOPSUPPORTST 속성은 사용되지 않습니다.
- WSDLFILE은 WEBSERVICE와 연관된 JSON 스키마 파일의 이름을 리턴합니다.

WEBSERVICE 자원에서

- ARCHIVEFILE 및 VALIDATION 매개변수는 사용되지 않으며 해당 값은 무시됩니다.
- WSDLFILE은 WEBSERVICE와 연관된 JSON 스키마 파일의 이름입니다.

웹 서비스에 대한 CICS 자원

PIPELINE, WEBSERVICE, URIMAP 및 TCPIPSERVICE 자원은 CICS의 웹 서비스를 지원합니다.

PIPELINE

PIPELINE 자원 정의가 모든 웹 서비스에 필요합니다. 이는 서비스 요청 및 응답 시에 작동하는 메시지 핸들러 프로그램에 대한 정보를 제공합니다. 일반적으로, 단일 PIPELINE 자원 정의는 많은 애플리케이션에서 사용할 수 있는 인프라를 정의합니다. 메시지 핸들러에 대한 정보가 간접적으로 제공됩니다. PIPELINE 자원 정의는 핸들러 및 해당 구성에 대한 XML 설명을 포함하는 z/OS UNIX 파일의 이름을 지정합니다.

서비스 요청자를 위해 작성되는 PIPELINE 자원은 서비스 제공자와 그 반대를 위해 사용될 수 없습니다. 두 종류의 PIPELINE 정의는 CONFIGFILE 속성에 지정되는 파이프라인 구성 파일의 콘텐츠에 의해 구별됩니다. 서비스 제공자의 경우 최상위 레벨 요소는 <provider_pipeline>이고, 서비스 요청자의 경우는 <requester_pipeline>입니다.

WEBSERVICE

애플리케이션 데이터 구조와 SOAP 메시지 사이의 맵핑이 CICS 웹 서비스 지원을 사용하여 생성될 때만 WEBSERVICE 자원 정의가 필요합니다. 이는 웹 서비스 설정에 배치된 CICS 애플리케이션 프로그램에 대한 런타임 환경의 측면을 정의합니다.

CICS에서 WEBSERVICE 자원에 대한 일반적 자원 정의 메커니즘을 제공하지만, PIPELINE 자원 정의에 대한 픽업 디렉토리가 스캐닝될 때 일반적으로 웹 서비스 바인딩 파일에서 자동으로 작성됩니다. 이는 PIPELINE 자원이 설치되거나 PERFORM PIPELINE SCAN 명령의 결과로 발생할 수 있습니다. 이 경우, WEBSERVICE 자원에 적용된 속성은 웹 서비스 바인딩 파일에서 발생하며, 이는 웹 서비스 지원에 의해 작성됩니다. 바인딩 파일의 정보는 웹 서비스 설명에서 발생하거나, 웹 서비스 지원의 매개변수로서 공급됩니다.

서비스 요청자를 위해 작성되는 WEBSERVICE 자원은 서비스 제공자에 대해 사용될 수 없고, 그 반대도 마찬가지입니다. 두 종류의 WEBSERVICE 자원은 자원 정의에서 PROGRAM 속성에 의해 구별됩니다. 서비스 제공자의 경우, 속성이 지정되어야 합니다. 서비스 요청자의 경우, 이는 생략되어야 합니다.

URIMAP

URIMAP 정의는 요청을 제공하는 기타 자원(예: PIPELINE 자원)에 인바운드 웹 서비스 요청의 URI를 맵핑하는 정보를 포함할 때 서비스 제공자에 필요합니다. 서비스 요청자 사용자 ID 정보가 HTTP 인증 헤더에서 서비스 제공자로 전달된다는 것을 URIMAP 자원 정의가 지정하므로 사용자가 HTTP 기본 인증을 사용 중이면 이 URIMAP 정의가 필요합니다.

두 번째 선택적 URIMAP 정의는 WSDL 검색을 위해 서비스 제공자에 존재할 수 있습니다. 이 URIMAP 자원 정의에는 웹 서비스와 연관된 문서 또는 WSDL 문서에 대한 인바운드 요청의 URI를 맵핑하는 정보가 포함됩니다.

CICS 웹 서비스 지원을 사용하여 배치되는 서비스 제공자의 경우, CICS에서 일반적인 자원 정의 메커니즘을 제공하지만, 일반적으로 픽 디렉토리가 스캐닝될 때 URIMAP 자원이 자동으로 작성됩니다. 이 스캔은 PIPELINE 자원이 설치되거나 PERFORM PIPELINE SCAN 명령의 결과로 발생할 수 있습니다. 특정 URI와 WEBSERVICE 자원을 연관시키기 위해 CICS에 정보를 제공하는 URIMAP 자원은 필수 자원입니다. 이 자원에 대한 속성은 픽업 디렉토리에서 웹 서비스 바인딩 파일에 의해 지정됩니다. 특정 URI와 WSDL 문서 또는 WSDL 아카이브 파일을 연관시키기 위해 CICS에 정보를 제공하는 URIMAP 자원은 선택적 자원이고 WSDL 파일 또는 WSDL 아카이브 파일이 픽업 디렉토리에 존재하는 경우에 작성됩니다. 웹 서비스 제공자에 대한 URIMAP 자원 작성에 관한 자세한 정보는 150 페이지의 『웹 서비스 지원을 사용하여 웹 서비스 제공자 작성』의 내용을 참조하십시오.

서비스 요청자의 경우, PIPELINE 자원이 설치되거나 PERFORM PIPELINE SCAN 명령의 결과로 설치될 때 CICS는 URIMAP 자원을 자동으로 작성하지 않습니다. 서비스 요청자는 요청할 때 URIMAP 자원을 사용하도록 요청되지 않습니다. 애플리케이션 프로그램에서 아웃바운드 요청의 URI를 직접적으로 지정할 수 있습니다. 그러나, 클라이언트 요청에 대한 URIMAP 자원을 작성하고 서비스 요청자가 URI를 제공하기 위해 URIMAP 자원을 사용하는 경우, 다음의 이점을 얻게 됩니다.

- 시스템 관리자는 연결의 엔드포인트에 대한 변경사항을 관리할 수 있습니다. 따라서, 서비스 제공자의 URI가 변경되면 사용자 애플리케이션을 다시 컴파일할 필요 없습니다.
- CICS가 사용 후 URIMAP 자원으로 열린 연결을 계속 열어 놓게 하고 동일한 서비스를 호출하는 다른 애플리케이션 또는 후속 요청에 대한 애플리케이션에 의한 재사용을 위해 풀에 이를 위치시키도록 선택할 수 있습니다. 연결 풀링은 SOCKETCLOSE 속성 세트를 가진 URIMAP 자원을 지정할 때에만 사용 가능합니다. 연결 풀링의 성능 장점에 대한 자세한 정보는 성능 개선에서 HTTP 클라이언트 성능에 대한 연결 풀링의 내용을 참조하십시오.

특정 방법으로 URIMAP 자원 속성을 구성하면 첨부된 사용자 트랜잭션에 의해 직접적으로 처리되고 웹 첨부 태스크를 무시하는 인바운드 요청을 사용할 수 있도록 합니다. 자세한 정보는 성능을 개선하기 위해 HTTP 요청이 직접 접속된 사용자 트랜잭션에 의해 처리됩니다.의 내용을 참조하십시오.

TCPIPSERVICE

TCPIPSERVICE 정의는 HTTP 전송을 사용하는 서비스 제공자에 필요합니다. 인바운드 요청이 수신되는 포트에 대한 정보를 포함합니다.

특별한 애플리케이션 프로그램을 지원하는 데 필요한 자원은 다음 기준에 따라 다릅니다.

- 애플리케이션 프로그램이 서비스 제공자 또는 서비스 요청자인지 여부.
- 애플리케이션이 CICS 웹 서비스 지원으로 배치되는지 여부.

서비스 요청자 또는 제공자	사용된 CICS 웹 서비스 자원	필수 PIPELINE	필 수 WEBSERVICE	필수 URIMAP	필 수 TCPIPSERVICE
제공자	예	예	예(그러나, 노트 1 참조)	예(그러나, 노트 1 참조)	노트 2의 내용을 참조하십시오
제공자	아니오	예	아니오	예	노트 2의 내용을 참조하십시오
요청자	예	예	예	노트 3의 내용을 참조하십시오	아니오
요청자	아니오	예	아니오	3	아니오

참고사항:

1. CICS 웹 서비스 지원이 애플리케이션 프로그램을 배치하는데 사용될 때, PIPELINE의 픽업 디렉토리가 스캐닝될 때 WEBSERVICE와 두 개의 URIMAP 자원이 자동으로 작성될 수 있습니다. 특정 URI와 WEBSERVICE 자원을 연관시키기 위해 첫 번째 URIMAP 자원이 요청되고 CICS에 정보를 제공합니다. 외부 요청자가 URI를 사용하여 WSDL 아카이브 파일 또는 WSDL 문서를 발견할 수 있도록 특정 URI를 WSDL 아카이브 파일 또는 WSDL 문서와 연관시키기 위해 두 번째 URIMAP 자원은 선택사항이고 CICS에 정보를 제공합니다. PIPELINE 스캔의 픽업 디렉토리는 PIPELINE 자원이 설치되거나 PERFORM PIPELINE SCAN 명령의 결과로 설치될 때 발생합니다.
2. TCPIPSERVICE 자원은 HTTP 전송이 사용될 때 필요합니다. WebSphere MQ 전송이 사용될 때, TCPIPSERVICE 자원은 필요하지 않습니다.
3. URIMAP 자원은 서비스 요청자에 대해 선택적이고 CICS 웹 서비스 지원은 자동으로 하나를 생성하지 않습니다. 사용하려는 서비스 요청자에 대한 자체 URIMAP 자원을 정의할 때, 연결 풀링을 구현하고 서비스 제공자를 위해 URI에 대한 변경사항을 관리합니다.

특정 방법으로 TCPIP 자원 속성을 구성하면 첨부된 사용자 트랜잭션에 의해 직접적으로 처리되고 웹 첨부 태스크를 무시하는 인바운드 요청을 사용할 수 있도록 합니다. 자세한 정보는 HTTP 요청은 직접적으로 첨부된 사용자 트랜잭션에 의해 처리됩니다.의 내용을 참조하십시오.

일반적으로, CICS 시스템에서 많은 웹 서비스 애플리케이션을 배치할 때, 자원의 각 유형 중 하나보다 더 가지고 있습니다. 이 경우, 애플리케이션 사이에서 일부 자원을 공유할 수 있습니다. 각 웹 서비스 파일 또는 자원은 기타 유형의 하나 이상의 CICS 자원과 연관됩니다.

표 3. 각 웹 서비스 파일 및 자원과 연관된 기타 CICS 자원

웹 서비스 파일 또는 자원	연관된 자원
파이프라인 구성 파일	파일을 언급하는 둘 이상의 PIPELINE 자원.
PIPELINE	- PIPELINE 자원을 언급하는 둘 이상의 URIMAP 자원. - PIPELINE 자원을 언급하는 둘 이상의 URIMAP 자원. - PIPELINE 자원의 픽업 디렉토리에 있는 둘 이상의 웹 서비스 바인딩 파일.
웹 서비스 바인딩 파일	- 자동으로 바인딩 파일에서 생성되는 하나 URIMAP 자원. 서비스 제공자를 위한 URIMAP 자원을 추가적으로 정의할 수 있고 서비스 요청자를 위한 URIMAP 자원을 정의할 수 있습니다. - 자동으로 바인딩 파일에서 생성되는 하나의 WEBSERVICE 자원. 필요한 경우 WEBSERVICE 자원을 추가적으로 정의할 수 있습니다.
WEBSERVICE	둘 이상의 URIMAP 자원. WEBSERVICE 자원이 자동으로 서비스 제공자에 대한 바인딩 파일로부터 생성되는 경우, CICS는 하나의 해당 URIMAP 자원을 생성합니다. 서비스 제공자를 위한 URIMAP 자원을 추가적으로 정의할 수 있고 서비스 요청자를 위한 URIMAP 자원을 정의할 수 있습니다.
URIMAP	URIMAP 자원에서 명시적으로 이름 지정될 때 하나의 TCPIPService 자원.
TCPIPService	여러 URIMAP 자원.

웹 서비스 발견

제공자 모드 웹 서비스와 연관된 WSDL 문서는 자동으로 웹에 공개됩니다.

웹 서비스 호스팅 환경에는 원격 클라이언트(일반적으로 웹 브라우저를 사용하는 애플리케이션 개발자)가 접미부 `?wsdl`인 웹 서비스 URI를 사용하여 웹 서비스 WSDL을 조회할 수 있는 규칙이 존재합니다. 이 규칙에 따라 일반 WSDL 저장소가 없어도 관심있는 대상들에게 WSDL을 쉽게 분배할 수 있습니다. 이 규칙은 CICS로 구현됩니다.

예를 들어, CICS로 웹 서비스를 호스팅하고 다음 URI로 공개할 수 있습니다.

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

연관된 WSDL 문서는 웹 브라우저를 사용하여 다음 URI를 요청하여 복구될 수 있습니다.

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

서비스 제공자를 위한 WSDL 문서를 공개하면 URIMAP 자원을 사용하여 발견할 수 있습니다. 각 PIPELINE 자원을 설치할 때마다 CICS가 PIPELINE 자원의 WSDIR 속성에 지정된 디렉토리(픽업 디렉토리)를 스캔합니다. 이 디렉토리에 WSDL 아카이브 파일 또는 WSDL 문서가 포함되는 경우에는 두 번째 URIMAP 자원이 설치됩니다. 이 새 URIMAP 자원은 WSDL 아카이브 파일 또는 WSDL 문서를 특정 URI와 연관시키기 위한 정보를 CICS에 제공하며 이에 따라 외부 요청자가 해당 URI를 사용하여 WSDL 아카이브 파일 또는 WSDL 문서를 발견할 수 있습니다. 이 URI는 WEBSERVICE와 연관된 URI와 경로가 같으며 접미부 `?wsdl`이 추가되었습니다.

WSDL 아카이브 파일은 하나 이상의 WSDL 문서를 포함할 수 있습니다. 픽업 디렉토리에 WSDL 아카이브 파일과 WSDL 문서가 포함되는 경우에는 URI가 WSDL 아카이브만 리턴합니다. 지원되는 아카이브 파일 형식은 .zip 파일 유형입니다. SPI 및 CEMT를 사용하여 WSDL 아카이브 파일 또는 WSDL 문서를 발견할 수도 있습니다. WSDL 아카이브 파일의 WSDL 문서는 SOAP 메시지 유효성 검증에 사용할 수 있습니다.

WebSphere MQ 전송을 사용하도록 CICS 구성

CICS에서 SOAP 웹 서비스에 WebSphere MQ 전송을 사용하려면 그에 따라 CICS 리전을 구성해야 합니다.

이 태스크 정보

참고: JSON 웹 서비스에 Websphere MQ 전송을 사용할 수 없습니다.

프로시저

1. WebSphere MQ 라이브러리 `thlqual.SCSQAUTH`를 CICS 프로시저의 STEPLIB 연결에 포함시키십시오. 이 라이브러리를 CICS 라이브러리 뒤에 포함시켜 올바른 코드가 사용되는지 확인하십시오. `thlqual`은 WebSphere MQ 라이브러리를 위한 상위 레벨 규정자입니다.

2. 다음 WebSphere MQ 라이브러리를 CICS 프로시저의 DFHRPL 연결에 포함시키십시오. 이러한 라이브러리를 CICS 라이브러리 뒤에 포함시켜 올바른 코드가 사용 되는지 확인하십시오.

thlqual.SCSQCICS

thlqual.SCSQLLOAD

thlqual.SCSQAUTH

*thlqual*은 WebSphere MQ 라이브러리를 위한 상위 레벨 규정자입니다. CICS-WebSphere MQ API 교차 엑시트(CSQCAPX)를 사용하는 경우 프로그램에 대한 로드 모듈을 포함하는 라이브러리의 이름도 추가하십시오. SCSQCICS 라이브러리는 WebSphere MQ 제공 샘플을 실행하려는 경우에만 필요합니다. 그렇지 않은 경우에 이 라이브러리는 CICS 프로시저에서 제거할 수 있습니다.

3. CICS 리전의 MQCONN 자원을 설치하십시오. MQCONN 자원은 기본 WebSphere MQ 큐 관리자의 이름 또는 연결의 큐 공유 그룹과 같은 CICS 와 WebSphere MQ 간 연결 속성을 지정합니다. 자세한 정보는 구성에서 MQCONN 자원 설정의 내용을 참조하십시오.
4. CICS 초기화 시 CICS-WebSphere MQ 연결이 자동으로 시작되도록 CICS 시스템 초기화 매개변수 **MQCONN=YES**를 지정하십시오.

CICS가 WebSphere MQ에 대한 연결을 시작할 수 있으려면 MQCONN 자원 정의가 설치되어야 합니다. CICS 초기화에서 연결을 자동으로 시작할 때, 초기 또는 콜드 스타트의 경우, MQCONN 자원 정의는 **GRPLIST** 시스템 초기화 매개변수에 의해 이름 지정되는 목록에 이름 지정된 그룹 중 하나에 있어야 합니다. CICS의 워밍업 또는 긴급 시작의 경우, MQCONN 자원 정의는 이전 CICS 실행의 끝에 설치되어야 합니다.

5. 원격 CICS 시스템에 대한 리전 간 통신(IRC)이 있는 CICS 시스템에서 CICS-WebSphere MQ 어댑터를 사용하는 경우 어댑터를 시작하기 전에 CICS 시스템 초기화 매개변수 **IRCSTRT=YES**를 지정하여 IRC 기능이 OPEN인지 확인하십시오. IRC 액세스 방법이 교차 메모리로 정의되는 경우 (즉, **ACCESSMETHOD(XM)**) IRC 기능은 OPEN이어야 합니다.
6. 큐 관리자 및 CICS에서 사용되는 코드화된 문자 세트 ID(CCSID) 및 UTF-8과 UTF-16 코드 페이지가 z/OS 변환 서비스로 구성되는지 확인하십시오. CICS 코드 페이지는 **LOCALCCSID** 시스템 초기화 매개변수에서 지정됩니다.
7. 다음과 같이 CICS CSD를 갱신하십시오.
 - a. CICS의 이전 릴리스와 CSD를 공유하지 않는 경우 CSD에서 그룹 CSQCAT1 및 CSQCKB와 해당 그룹 또는 해당 그룹 항목의 사본을 제거하십시오. 그룹 CSQCAT1에서 CKQQ TDQUEUE도 삭제해야 합니다. 이제 CKQQ에 대한 정의가 CICS CSD 그룹 DFHDCTG에서 제공됩니다.

- b. 이전 CICS 릴리스와 CSD를 공유하지 않는 경우 CSQCAT1 및 CSQCKB와 해당 그룹 또는 해당 콘텐츠의 사본은 CICS TS 4.1 또는 CICS TS 3.2에 설치되지 않습니다. 그룹 CSQCAT1에서 CKQQ TDQUEUE도 삭제해야 합니다. 이제 CKQQ에 대한 정의가 CICS CSD 그룹 DFHDCTG에서 제공됩니다. CICS TS 3.2 이전 CICS TS 릴리스에서 DFHLIST를 설치한 후에 CSQCAT1 및 CSQCKB 그룹을 그룹 목록의 일부로 설치하여 그룹 DFHMQ를 대체하고 필수 정의를 올바르게 설치하십시오.
8. 데드 레터 큐, 기본 전송 큐 및 CICS-WebSphere MQ 어댑터 오브젝트의 WebSphere MQ 정의를 갱신하십시오. 샘플 CSQ4INYG를 사용할 수 있지만 CICS 릴리스의 MQINI 자원 정의에 있는 기본 시작 큐 이름과 일치시키기 위해 시작 큐 이름을 변경해야 될 수도 있습니다. 큐 관리자 시작 프로시저의 CSQINP2 DD 연결에서 이 멤버를 사용하거나 CSQUTIL 유틸리티의 COMMAND 함수에 대한 입력으로 이 멤버를 사용하여 필수 DEFINE 명령을 실행할 수 있습니다. CSQUTIL 유틸리티를 사용하는 것이 더 좋습니다. 이 경우 WebSphere MQ를 재시작할 때마다 이러한 오브젝트를 재정의하지 않아도 되기 때문입니다.

WebSphere MQ 전송

CICS는 서비스 제공자 및 서비스 요청자 역할에서 모두 WebSphere MQ 전송을 사용하여 SOAP 메시지를 수신하고 WebSphere MQ로 보낼 수 있습니다.

CICS는 서비스 제공자로서, WebSphere MQ 트리거링을 사용하여 애플리케이션 큐에서 SOAP 메시지를 처리합니다. 트리거링은 시작 큐와 로컬 큐를 사용하여 실행됩니다. 로컬(애플리케이션) 큐 정의에는 다음 정보가 포함됩니다.

- 트리거 메시지가 생성되는 시점에 대한 기준. 예를 들어, 첫 번째 메시지가 로컬 큐에 도달할 때 또는 로컬 큐에 도달하는 모든 메시지. CICS SOAP 처리의 경우, 첫 번째 메시지가 로컬 큐에 도달할 때 트리거링이 발생하도록 지정합니다.

로컬 큐 정의는 또한 트리거 데이터가 대상 애플리케이션으로 전달되는 것으로 지정하며, CICS SOAP 처리(트랜잭션 CPIL)의 경우 인바운드 메시지와 함께 전달되지 않는 경우 사용할 기본 대상 URL을 지정합니다.

- 프로세스 정의를 식별하는 프로세스 이름. 프로세스 정의는 메시지 처리 방법을 설명합니다. CICS SOAP 처리의 경우 CPIL 트랜잭션을 지정하십시오.
- 트리거 메시지를 보내야 하는 시작 큐의 이름.

로컬 큐에서 메시지가 도달하면 큐 관리자가 트리거 메시지를 생성하고 지정된 시작 큐로 보냅니다. 트리거 메시지에는 프로세스 정의의 정보가 포함됩니다. 트리거 모니터는 시작 큐에서 트리거 메시지를 검색하며 CPIL 트랜잭션이 로컬 큐에서 메시지 처리를 시작하도록 예약합니다. 트리거링에 대한 자세한 정보는 제품 개요에서 태스크 시작기 또는 트리거 모니터(CKTI)를 참조하십시오.

로컬 큐에서 메시지가 도달하면 트리거 모니터(WebSphere MQ가 제공)가 로컬 큐에서 메시지를 처리하고 CICS SOAP 파이프라인이 큐에서 SOAP 메시지를 처리하도록 구동시키는 CPIL 트랜잭션을 스케줄하도록 CICS를 구성할 수 있습니다.

CICS가 WebSphere MQ에서 수신되는 SOAP 메시지에 대한 응답을 구성할 때 보고서 옵션 MQRO_PASS_CORREL_ID가 설정되지 않은 경우에는 상관 ID 필드가 입력 메시지의 메시지 ID로 채워집니다. 이 보고서 옵션이 설정된 경우에는 응답에 대한 입력 메시지에서 상관 ID가 전파됩니다.

서비스 요청자로서, 아웃바운드 요청 시 대상 웹 서비스에 대한 응답이 특정 응답 큐에서 리턴되는 것으로 지정할 수 있습니다.

두 가지 경우 모두 CICS 및 WebSphere MQ에는 필수 자원과 큐를 정의하기 위한 구성이 필요합니다.

서비스 제공자에서 로컬 큐 정의

서비스 제공자에서 WebSphere MQ 전송을 사용하려면 처리될 때까지 요청 메시지를 저장하는 한 개 이상의 로컬 큐 및 요청 메시지를 처리하는 CICS 트랜잭션을 지정하는 한 개의 트리거 프로세스를 정의해야 합니다.

프로시저

1. 시작 큐를 정의하십시오. 다음 명령을 사용하십시오.

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

여기서, *initiation_queue*는 CICS 리전에 대한 MQINI 자원 정의의 INITQNAME 속성에 대해 지정된 값과 같습니다. MQINI는 MQCONN 자원을 설치할 때 CICS가 작성하는 묵시적 자원입니다.

2. 각 로컬 요청 큐에 대해 QLOCAL 오브젝트를 정의하십시오. 다음 명령을 사용하십시오.

```
DEFINE
QLOCAL('queuename')
DESCR('description')
PROCESS(processname)
INITQ('initiation_queue')
TRIGGER
TRIGTYPE(FIRST)
```

```
TRIGDATA('default_target_service')
BOTHRESH(nnn)
BOQNAME('requeuename')
```

여기서,

- *queuename*은 로컬 큐 이름입니다.
 - *processname*은 트리거 이벤트가 발생하면 큐 관리자가 시작하는 애플리케이션을 식별하는 프로세스 인스턴스의 이름입니다. 각 QLOCAL 오브젝트에 대해 동일한 이름을 지정하십시오.
 - *initiation_queue*는 사용할 시작 큐(예: CICS 리전에 대한 MQINI 정의에서 지정된 시작 큐)의 이름입니다.
 - *default_target_service*는 요청에서 서비스가 지정되지 않은 경우에 사용되어야 할 기본 대상 서비스입니다. 대상 서비스의 양식은 '/string'이고 URIMAP 정의의 경로(예: '/SOAP/test/test1')와 일치 비교하는 데 사용됩니다. 첫 번째 문자는 '/'이어야 합니다.
 - *nnn*은 시도한 재시도 횟수입니다.
 - *requeuename*은 실패한 메시지가 전송되는 큐의 이름입니다.
3. 트리거 프로세스를 지정하는 PROCESS 오브젝트를 정의하십시오. 다음 명령을 사용하십시오.

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

여기서,

*processname*은 프로세스의 이름이고 요청 큐를 정의할 때 사용되는 이름과 같아야 합니다.

시작 큐에 대한 작업

다음 인터페이스를 사용하여 시작 큐의 이름에 대해 조회할 수 있습니다.



CICS Explorer



CICS Explorer 운영 보기

Websphere MQ 시작 큐 보기의 **Name** 속성을 사용하십시오.

CICSplex SM

 MQINI 운영 보기

CEMT

 INQUIRE MQINI 명령

CICS SPI

 INQUIRE MQINI 명령

서비스 요청자에서 로컬 큐 정의

서비스 요청자의 아웃바운드 요청에 WebSphere MQ 전송을 사용할 때 대상 웹 서비스에 대해 응답이 사전정의된 응답 큐에서 리턴되어야 함을 URI에서 지정할 수 있습니다. 이를 수행하는 경우 QLOCAL 오브젝트를 사용하여 각 응답 큐를 정의해야 합니다.

이 태스크 정보

요청과 연관된 URI가 응답 큐를 지정하지 않는 경우 CICS는 응답에 동적 큐를 사용합니다.

프로시저

옵션: 사전정의된 응답 큐를 지정하는 각 QLOCAL 오브젝트를 정의하려면 다음 명령을 사용하십시오.

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

여기서,

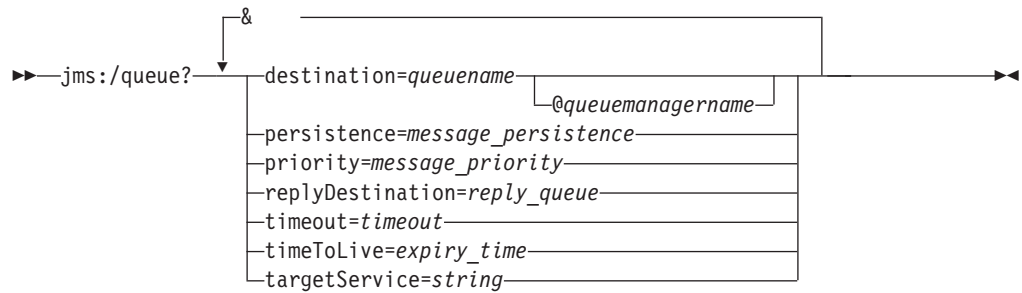
*reply_queue*는 로컬 큐 이름입니다.

*nnn*은 시도될 재시도 횟수입니다.

WebSphere MQ 전송을 위한 URI

서비스 요청자와 서비스 제공자 사이의 통신을 위해 WebSphere MQ를 사용하는 경우, 대상의 URI는 대상을 큐로 식별하고 WebSphere MQ가 요청 및 응답을 처리해야 하는 방식을 지정하는 정보를 포함하는 양식입니다.

구문



CICS는 다음 옵션을 사용합니다. 다른 웹 서비스 제공자는 여기서 설명하지 않는 추가 옵션을 사용할 수 있습니다. 전체 URI는 서비스 제공자로 전달되지만 CICS는 지원되지 않고 URI로 코딩되는 옵션을 모두 무시합니다. CICS는 옵션 이름의 대소문자를 구분하지 않습니다. 그러나 이 URI 스타일을 지원하는 일부 다른 구현은 대소문자를 구분합니다.

destination=queueename [**@queueanagername**]

*queueename*은 대상 큐 관리자에서 입력 큐의 이름입니다.

*queueanagername*은 대상 큐 관리자의 이름입니다.

persistence=message_persistence

다음 옵션 중 하나를 지정합니다.

0 기본 큐 지속성으로 지속성이 정의됩니다.

1 메시지가 지속되지 않습니다.

2

메시지가 지속됩니다.

옵션이 지정되지 않거나 잘못 지정된 경우에는 기본 큐 지속성을 사용합니다.

priority=message_priority

메시지 우선순위를 지정합니다. CICS는 메시지 우선순위에 대해 0 - 9 범위의 정수 값을 지원합니다. 우선순위가 가장 높은 메시지에는 9가 지정되고 우선순위가 가장 낮은 메시지에는 0이 지정됩니다. 또는 대상 큐에 정의된 기본 우선순위를 사용하려면 **-1**을 지정합니다.

replyDestination=reply_queue

응답 메시지에 사용될 큐를 지정합니다. 이 옵션이 지정되지 않으면 CICS가 응답 메시지에 동적 큐를 사용합니다. 이 옵션을 사용하기 전에 QLOCAL 오브젝트에서 응답 큐를 정의해야 합니다.

timeout=timeout

서비스 요청자가 응답을 대기하는 제한시간(밀리초). 값 0이 지정되거나 이 옵션이 생략되면 요청 제한시간이 초과되지 않습니다.

timeToLive=expiry-time

요청 만료 시간(밀리초)을 지정합니다. 옵션이 지정되지 않거나 잘못 지정되면 요청이 만료되지 않습니다.

targetService=string

대상 서비스를 식별합니다. CICS가 서비스 제공자인 경우, 대상 서비스는 '/string' 양식이어야 합니다. CICS가 URIMAP과의 일치 확인을 시도할 때 경로로 사용하기 때문입니다. 지정되지 않으면 서비스 제공자의 입력 큐에서 TRIGDATA에 지정된 값이 사용됩니다.

이 예제는 WebSphere MQ 전송에 대한 URI를 보여줍니다.

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

"connectionFactory" 및 "initialContextFactory"에 대한 정보는 *WebSphere MQ* 제품 문서를 참조하십시오.

지속적 메시지를 지원하도록 CICS 구성

CICS는 WebSphere MQ 전송 프로토콜을 사용하여 CICS 리전에 배치되는 웹 서비스 제공자 애플리케이션으로 지속적 메시지를 전송하기 위한 지원을 제공합니다.

이 태스크 정보

CICS는 BTS(Business Transaction Services)를 사용하여 CICS 시스템 장애가 발생할 경우 지속적 메시지가 복구되는지 확인합니다. BTS가 올바르게 작동하게 하려면 다음 단계를 수행하십시오.

프로시저

1. IDCAMS를 사용하여 MVS™에 대한 저장소 파일 및 로컬 요청 큐를 정의하십시오. 파일 정의를 위해 STRINGS에 적절한 값을 지정해야 합니다. 기본값 1로는 충분하지 않을 가능성이 높고 대신 10을 사용하도록 권장됩니다.
2. CICS에 대한 저장소 파일 및 로컬 요청 큐를 정의하십시오. CICS에 대한 로컬 요청 큐를 정의하는 방법의 세부사항은 238 페이지의 『서비스 제공자에서 로컬 큐 정의』에 설명되어 있습니다. 파일 정의에서 STRINGS에 적절한 값을 지정해야 합니다. 기본값 1로는 충분하지 않을 가능성이 높고 대신 10을 사용하도록 권장됩니다.
3. 저장소 파일 이름을 FILE 옵션의 값으로 사용하여 이름이 DFHMQSOA인 PROCESSTYPE 자원을 정의하십시오.
4. 지속적 메시지를 처리하는 동안 첫 번째 묵시적 동기점이 요청되기 전에 프로그램이 **EXEC CICS SYNCPOINT** 명령을 발행하는지 확인하십시오. 예를 들어, **EXEC CICS CREATE TDQUEUE**와 같은 SPI 명령을 사용하면 묵시적으로 동기점을 가져옵니다.

EXEC CICS SYNCPOINT 명령을 실행하면 지속적 메시지가 성공적으로 처리되었는지 확인합니다. 프로그램이 묵시적으로 동기점을 가져오려고 시도하기 전에 명시적으로 동기점을 요청하지 않는 경우 ASP7 이상 종료가 발행됩니다.

결과

다음에 수행할 작업

단방향 요청 메시지의 경우 웹 서비스가 이상 종료되거나 취소되면 트랜잭션 또는 프로그램이 실패한 요청을 재시도하거나 적절하게 실패를 보고할 수 있도록 충분한 정보가 보존됩니다. 이 복구 트랜잭션 또는 프로그램을 제공해야 합니다. 세부사항은 『지속적 메시지 처리』의 내용을 참조하십시오.

지속적 메시지 처리

웹 서비스 요청이 WebSphere MQ 지속적 메시지에서 수신되면 CICS가 프로세스 유형 DFHMQSOA로 고유 BTS 프로세스를 작성합니다. 인바운드 요청과 관련된 데이터는 프로세스와 연관된 BTS 데이터 컨테이너에서 캡처됩니다.

그런 다음 프로세스가 비동기로 실행되도록 스케줄됩니다. 웹 서비스가 완료되고 커밋되면 CICS가 BTS 프로세스를 삭제합니다. SOAP 결함이 생성되면 웹 서비스 요청자로 리턴되는 경우가 포함됩니다.

오류 처리

필수 BTS 프로세스를 작성할 때 오류가 발생하면 웹 서비스 트랜잭션이 이상 종료되고 인바운드 웹 서비스 요청이 처리되지 않습니다. BTS를 사용할 수 없으면 메시지 DFHPI0117이 발행되며 기존 채널 기반 컨테이너 메커니즘을 사용하여 BTS 없이 CICS가 계속됩니다.

웹 서비스 처리가 시작되거나 완료되기 전에 CICS 실패가 발생하면 BTS 복구로 CICS가 재시작될 때 프로세스 스케줄이 변경됩니다.

웹 서비스가 비정상적으로 종료되고 백아웃되면 BTS 프로세스가 ABENDED 상태로 완료된 것으로 표시됩니다. 응답이 필요한 요청 메시지의 경우에는 웹 서비스 요청자로 SOAP 결함이 리턴됩니다. BTS 프로세스가 취소되고 CICS가 실패한 요청에 대한 정보를 보유하지 않습니다. CICS는 임시 데이터 큐 CSBA의 경우 메시지 DFHBA0104를 발행하고 임시 데이터 큐 CPIO의 경우에는 메시지 DFHPI0117을 발행합니다.

단방향 메시지의 경우에는 실패에 대한 정보를 요청자로 리턴할 수 있는 방법이 없으므로 BTS 프로세스가 COMPLETE ABENDED 상태로 유지됩니다. CICS는 임시 데이터 큐 CSBA의 경우 메시지 DFHBA0104를 발행하고 임시 데이터 큐 CPIO의 경우에는 DFHPI0116을 발행합니다.

CBAM 트랜잭션을 사용하여 COMPLETE ABENDED 프로세스를 표시하거나, DFHMQSOA의 COMPLETE ABENDED 프로세스를 확인하는 복구 트랜잭션을 제공하고 적절한 조치를 수행할 수 있습니다.

예를 들어, 복구 트랜잭션은 다음 조치를 수행할 수 있습니다.

1. **RESET ACQPROCESS** 명령을 사용하여 BTS 프로세스를 재설정합니다.
2. **RUN ASYNC** 명령을 발행하여 실패 웹 서비스를 재시도합니다. 프로세스에서 다른 데이터-컨테이너의 재시도 횟수를 유지하여 반복된 실패를 방지할 수 있습니다.
3. 연관된 데이터-컨테이너의 정보를 사용하여 문제점을 보고합니다.

DFHMQORIGINALMSG 데이터-컨테이너는 RFH2 헤더를 포함할 수 있는 WebSphere MQ에서 수신된 메시지를 포함합니다.

DFHMQMSG 데이터 컨테이너는 RFH2 헤더가 제거된 WebSphere MQ 메시지를 포함합니다.

DFHMQDLQ 데이터-컨테이너는 원래 메시지와 연관된 데드 레터 큐의 이름을 포함합니다.

DFHMQCONT 데이터-컨테이너는 원래 메시지에 대한 **MQ GET**와 관련이 있는 WebSphere MQ MQMD 제어 블록을 포함합니다.

웹 서비스 지원과 WSRR 간의 상호 운용성

CICS 웹 서비스 지원은 WSRR(IBM WebSphere Service Registry and Repository)과 상호 운용될 수 있습니다. 요청하는 웹 서비스를 보다 빠르게 찾고 제공하는 웹 서비스의 버전 제어를 강제 실행하려면 WSRR을 사용하십시오.

DFHLS2WS 및 DFHWS2LS 모두에는 WSRR과 상호 운용하기 위한 매개변수가 포함됩니다. DFHLS2WS에는 또한 WSRR의 WSDL 문서에 사용자 정의된 고유한 메타데이터를 추가할 수 있는 선택적 매개변수가 포함됩니다.

웹 서비스 지원이 WSRR과 안전하게 통신하도록 하기 위해 보안 소켓 레이어(SSL) 암호화를 사용할 수 있습니다. DFHLS2WS 및 DFHWS2LS에 모두 SSL 암호화를 위한 매개변수가 포함됩니다.

웹 서비스 지원과 WSRR에 SSL을 사용하려면 『웹 서비스 지원 및 WSRR에서 SSL을 사용하는 방법의 예제』를 참조하십시오.

웹 서비스 지원 및 WSRR에서 SSL을 사용하는 방법의 예제

SSL(Secure Socket Layer) 암호화를 사용하여 웹 서비스 지원과 WSRR(IBM WebSphere Service Registry and Repository) 서버 사이에서 안전하게 상호 운용할 수 있습니다. SSL 암호화를 사용하려면 키 저장소 및 신뢰 저장소가 필요하고 웹 서비스 지원에 대한 특정 매개변수도 지정해야 합니다.

이 태스크 정보

웹 서비스 지원과 WSRR 간 상호작용을 위해 SSL을 사용하려면 다음 단계를 완료하십시오.

프로시저

1. 개인 키 및 공개 키 인증서(PKC)를 위한 키 저장소를 작성하십시오.
 - a. IBM Key Management Utility(iKeyman)와 같은 키 구성 프로그램을 사용하여 키 저장소를 작성할 수 있습니다.
 - b. DFHWS2LS 또는 DFHLS2WS의 **SSL-KEYSTORE** 매개변수에 작성한 키 저장소의 완전한 이름을 지정하십시오.
 - c. 옵션: DFHWS2LS 또는 DFHLS2WS의 **SSL-KEYPWD** 매개변수에 작성한 키 저장소의 비밀번호를 지정하십시오.
2. 모든 신뢰 루트 인증 기관(CA) 인증서를 위한 신뢰 저장소를 작성하십시오. 이러한 인증서는 인바운드 공개 키 인증서의 신뢰를 확립하는 데 사용됩니다.
 - a. IBM Key Management Utility(iKeyman)와 같은 키 구성 프로그램을 사용하여 신뢰 저장소를 작성할 수 있습니다.
 - b. DFHWS2LS 또는 DFHLS2WS의 **SSL-TRUSTSTORE** 매개변수에 작성한 신뢰 저장소의 완전한 이름을 지정하십시오.
 - c. 옵션: DFHWS2LS 또는 DFHLS2WS의 **SSL-TRUSTPWD** 매개변수에 작성한 신뢰 저장소의 비밀번호를 지정하십시오.
3. 웹 서비스 지원이 SSL 암호화를 사용하여 WSRR과 통신할 수 있는지 테스트하십시오.
 - a. IBM WebSphere Application Server가 제공하는 샘플 파일을 사용하여 WSRR과 웹 서비스 지원을 테스트할 수 있습니다.
 - WebSphere Application Server가 제공하는 샘플 키 저장소는 DummyClientKeyFile.jks 및 DummyServerKeyFile.jks입니다.
 - WebSphere Application Server가 제공하는 샘플 신뢰 저장소는 DummyClientTrustFile.jks 및 DummyServerTrustFile.jks입니다.
 - b. 샘플 키의 키 및 신뢰 저장소 파일을 바꾸십시오. 이러한 키는 WebSphere Application Server와 함께 제공되고 보안을 위해 비공개입니다.

결과

이제 웹 서비스 지원이 SSL 암호화를 사용하여 네트워크 전체에서 안전하게 WSRR과 통신할 수 있습니다.

웹 서비스 인프라

CICS 리전의 CICS 애플리케이션은 웹 서비스 파이프라인을 사용하여 해당 리전에 대해 외부적인 애플리케이션에 서비스를 제공하거나 애플리케이션에서 서비스를 요청할 수 있습니다. CICS가 서비스 제공자일 때, CICS 애플리케이션은 외부 애플리케이션에 서비스를 제공합니다. CICS가 서비스 요청자일 때, 외부 애플리케이션은 CICS 애플리케이션에 서비스를 제공합니다. 웹 서비스 파이프라인은 사용 가능한 곳에서 zEnterprise Application Assist Processor(zAAP)를 사용하기 위해 구성될 수 있습니다.

서비스 제공자로서 CICS

외부 서비스 요청자에 서비스를 제공하기 위한 CICS의 경우, 이는 서비스 요청을 수신해야 하고 파이프라인을 통해 이를 대상 애플리케이션 프로그램으로 전달해야 합니다. 애플리케이션으로부터의 응답은 동일한 파이프라인을 통해 서비스 요청자에게 리턴됩니다.

그림 21에서는 CICS가 Java 파이프라인을 사용하는 서비스 제공자일 때 외부 서비스 요청자로부터 요청을 처리하기 위해 필요한 아키텍처 및 자원의 예제 구성을 표시합니다.

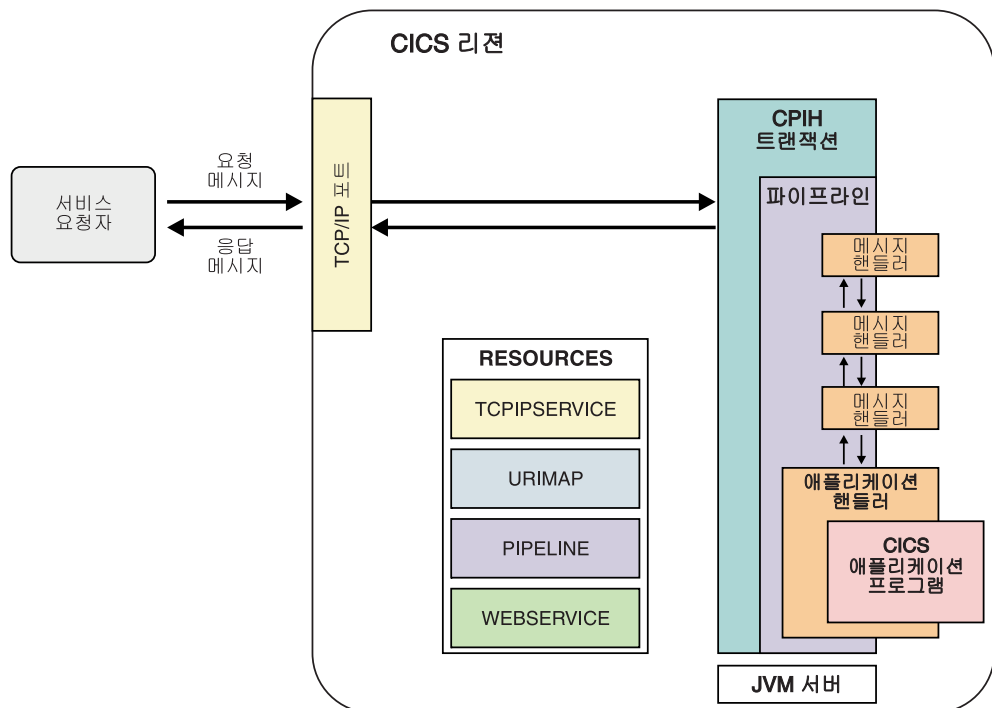


그림 21. 서비스 제공자에 대한 아키텍처 및 자원

요청을 처리하기 위해 CICS는 다음 조작을 수행해야 합니다.

1. 서비스 요청자로부터 요청을 수신하십시오.

TCPIPSERVICE 자원은 요청 수신을 위한 포트를 지정합니다. 이 포트는 CICS 제공 소켓 리스너 트랜잭션(CSOL)에 의해 모니터링됩니다.

2. 요청을 검사하고 대상 애플리케이션 프로그램과 관련된 콘텐츠를 추출하십시오.

요청 메시지가 적절한 포트에서 수신될 때, URIMAP 자원 정의는 해당 USAGE 속성을 PIPELINE로 설정하도록 하고 해당 PATH 속성을 요청에서 찾은 URI로 설정하도록 하는 URIMAP 정의에 대해 스캐닝됩니다. 적절한 URIMAP 정의가 발견되면, URIMAP 정의의 PIPELINE 및 WEBSERVICE 속성에서 PIPELINE 및 WEBSERVICE 정의가 사용됩니다. URIMAP 정의의 TRANSACTION 속성은 파이프라인을 처리하기 위해 첨부되어야 하는 트랜잭션의 이름을 판별합니다. 기본적으로, CPIH 트랜잭션이 사용됩니다. URIMAP 정의는 사용할 PIPELINE 및 WEBSERVICE 자원을 또한 식별합니다. 이러한 자원은 CICS가 수행하는 처리를 제어합니다.

3. 요청으로부터 추출된 데이터를 전달하여 애플리케이션 프로그램을 호출하십시오.

파이프라인 및 애플리케이션 핸들러의 메시지 핸들러는 서비스 제공자 애플리케이션 프로그램이 예상하는 애플리케이션 언어 구조로 요청 메시지를 변환합니다. 프로그램이 이 입력을 처리하고 애플리케이션 핸들러에 응답을 리턴합니다.

4. 애플리케이션 프로그램이 리턴하는 데이터를 사용하여 응답을 구성하고 서비스 요청자로 응답을 전송하십시오.

애플리케이션 핸들러 및 메시지 핸들러는 서비스 제공자 애플리케이션으로부터 수신한 응답 메시지를 원래 요청의 형식의 메시지로 변환합니다. 이 메시지는 서비스 요청자로 다시 전송됩니다.

파이프라인 내의 일부 처리는 파이프라인이 적절하게 구성되면 zEnterprise Application Assist Processor(zAAP)를 사용하여 수행될 수 있습니다. 자세한 정보는 249 페이지의 『Java 기반 SOAP 파이프라인』의 내용을 참조하십시오.

서비스 요청자로서 CICS

외부 서비스를 호출하기 위한 CICS의 경우, 애플리케이션 프로그램은 파이프라인을 통해 전달되는 요청을 대상 서비스에 전송합니다. 서비스로부터의 응답은 샘플 파이프라인을 통해 애플리케이션 프로그램으로 리턴됩니다.

248 페이지의 그림 22에서는 Java 파이프라인을 사용하여 CICS 리전에 대해 외부인 서비스 제공자로부터 데이터에 대한 CICS 애플리케이션 프로그램에서 요청을 처리하기 위해 필요한 자원 및 아키텍처의 예제 구성을 표시합니다.

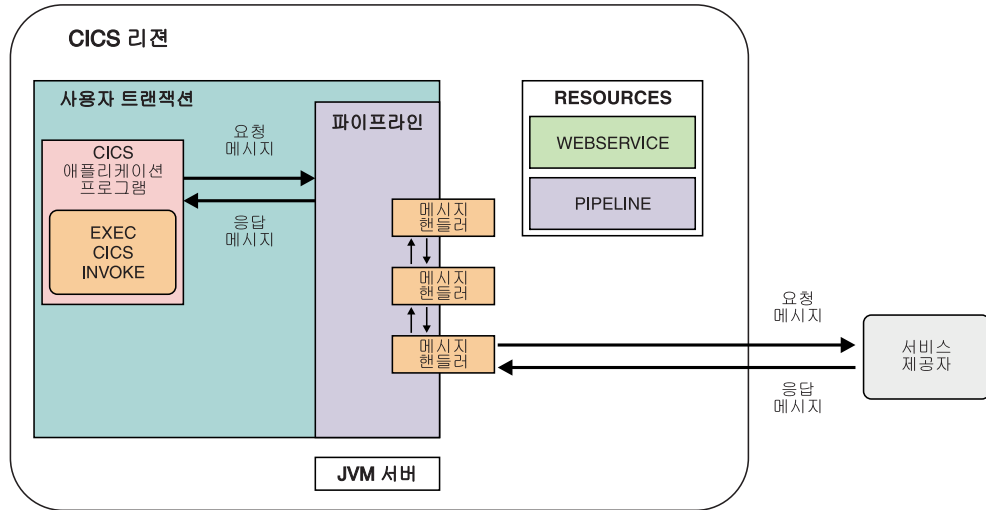


그림 22. 서비스 요청자에 대한 아키텍처 및 자원

요청을 처리하기 위해 CICS는 다음 조작을 수행해야 합니다.

1. 애플리케이션 프로그램이 제공하는 데이터를 사용하여 요청을 빌드하십시오.

CICS 애플리케이션 프로그램이 CICS 리전에 대해 외부적인 서비스 제공자에 대한 요청을 시작할 때, 요청자 애플리케이션은 EXEC CICS INVOKE SERVICE 명령을 호출합니다. EXEC CICS INVOKE SERVICE 명령은 파이프라인을 호출합니다. 파이프라인은 서비스 제공자가 처리할 수 있는 언어로 애플리케이션 언어 구조를 변환합니다(예: SOAP 메시지).

2. 요청을 서비스 제공자에 전송하십시오.

CICS는 HTTP 또는 WebSphere MQ를 사용하여 원격 서비스 제공자에 요청 메시지를 전송합니다.

3. 서비스 제공자로부터 응답을 수신하십시오.

서비스 제공자 응답 메시지가 수신될 때, CICS는 파이프라인에 메시지를 다시 전달합니다.

4. 응답을 검사하고 원래 애플리케이션 프로그램과 관련된 콘텐츠를 추출하십시오.

파이프라인은 애플리케이션 언어 구조로 서비스 제공자 응답 메시지를 변환합니다. 이는 애플리케이션 프로그램으로 전달됩니다. 제어가 애플리케이션 프로그램으로 리턴됩니다.

파이프라인 내의 일부 처리는 파이프라인이 적절하게 구성되면 zEnterprise Application Assist Processor(zAAP)를 사용하여 수행될 수 있습니다. 자세한 정보는 249 페이지의 『Java 기반 SOAP 파이프라인』의 내용을 참조하십시오.

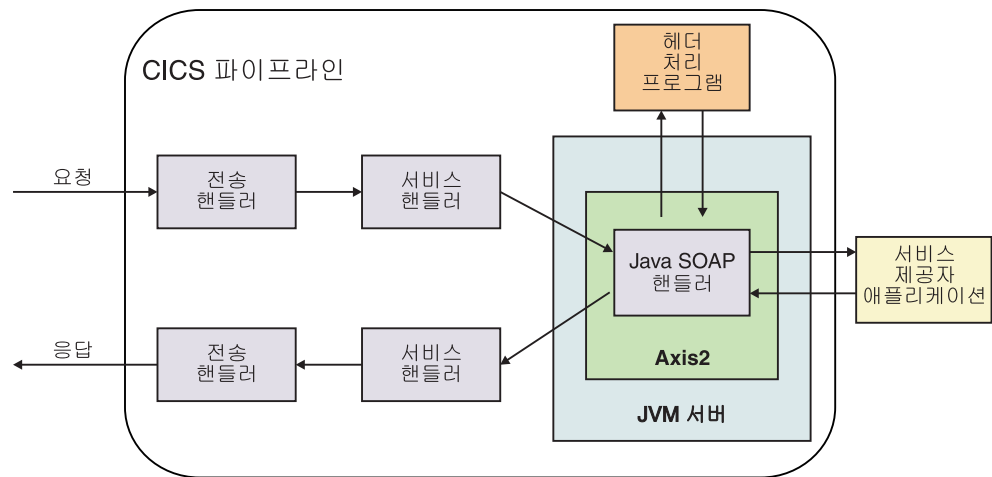
Java 기반 SOAP 파이프라인

CICS는 제공자 및 요청자 파이프라인에서 웹 서비스 요청을 처리하기 위해 Axis2 Java 기반 SOAP 엔진을 사용하여 지원합니다. Axis2가 Java를 사용하므로 SOAP 처리는 zEnterprise Application Assist Processor(zAAP)에 대한 오프로드에 적합합니다.

Axis2는 Apache 기초의 개방형 소스 웹 서비스 엔진이고 Java 환경에서 SOAP 메시지를 처리하기 위해 CICS가 제공됩니다. 파이프라인 구성 파일에 Java SOAP 핸들러를 추가하고 Axis2 처리를 다루기 위해 JVM 서버를 작성하여 Axis2를 사용하도록 선택할 수 있습니다.

Axis2 사용은 파이프라인을 사용하는 기존 웹 서비스에 대한 바인딩 파일을 재생성할 필요가 없습니다. 응답 시간은 Axis2를 사용할 때 더 느려질 수 있지만, zAAP에 SOAP 처리를 오프로드할 수 있습니다. zAAP에 오프로드하는 것에 대한 자세한 정보는 시작 하기의 CICS에 Java 지원의 내용을 참조하십시오.

CICS가 서비스 제공자일 때, Java 기반 터미널 핸들러는 Axis2를 사용하여 요청 메시지에 대한 SOAP 엔벨로프를 구문 분석합니다. 헤더 처리 프로그램을 사용하여 SOAP 메시지와 연관된 SOAP 헤더를 처리할 수 있습니다. Axis2는 SOAP 응답 메시지도 구성합니다. 이 프로세스는 다음 다이어그램에 표시됩니다.



CICS가 서비스 요청자일 때, 파이프라인의 Java 기반 초기 핸들러는 Axis2를 사용하여 요청 메시지에 대한 SOAP 엔벨로프를 생성합니다. 헤더 처리 프로그램을 사용하여 SOAP 메시지와 연관된 SOAP 헤더를 처리할 수 있습니다. Axis2는 SOAP 응답 메시지도 구문 분석합니다.

웹 서비스 애플리케이션 및 Java

제공자 모드 SOAP 파이프라인의 경우, 요청 및 응답 메시지가 애플리케이션 핸들러를 사용하여 웹 서비스 애플리케이션과 파이프라인의 터미널 핸들러 사이에 전달됩니다. 요

청이 애플리케이션에 의해 사용될 수 있도록 애플리케이션 핸들러는 SOAP 요청의 본문을 처리합니다. 애플리케이션 핸들러는 애플리케이션에서 리턴된 데이터를 사용하여 응답도 생성합니다. 사용자 파이프라인의 터미널 핸들러가 Java 기반 메시지 핸들러인 경우, 제공된 DFHPITP 애플리케이션 핸들러를 지정하는 것이 아니라 파이프라인 구성 파일에서 제공된 Axis2 애플리케이션 핸들러를 지정할 수 있습니다. 애플리케이션 핸들러 처리가 zAAP에 대해 오프로드될 수 있습니다. 애플리케이션 핸들러에 대한 자세한 정보는 327 페이지의 『애플리케이션 핸들러』의 내용을 참조하십시오.

요청자 모드 SOAP 파이프라인의 경우, 웹 서비스 애플리케이션은 **EXEC CICS INVOKE SERVICE** 명령을 사용하여 파이프라인을 호출합니다. 그런 다음, 요청 및 응답 메시지는 파이프라인에서 웹 서비스 애플리케이션과 초기 핸들러 사이에 전달됩니다. 파이프라인에서 초기 핸들러로서 Java 기반 핸들러를 지정하려면 zAAP에 대한 이 프로세스를 오프로드할 수 있도록 하면서 **EXEC CICS INVOKE SERVICE** 명령이 Axis2에 의해 처리됩니다. 첫 번째 핸들러가 Java 기반 핸들러가 아닌 경우, **EXEC CICS INVOKE SERVICE** 명령이 CICS에 의해 처리됩니다.

JVM 서버에서 Axis2 처리

Axis2에는 JVM 서버가 필요하고, 이는 CICS에서 JVMSERVER 자원으로 표시됩니다. JVM 서버는 단일 JVM의 다른 Java 프로그램에서 여러 병행 요청을 처리할 수 있는 런타임 환경입니다. JVM 서버에 대한 클래스 경로에는 Axis2 Java 아카이브 파일이 포함되어야 합니다. JVM 프로파일에서 JAVA_PIPELINE 옵션을 지정하여 클래스 경로에 모든 필수 JAR 파일을 자동으로 추가할 수 있습니다. 파이프라인 구성 파일은 Axis2를 지원하기 위해 구성되는 JVMSERVER 자원을 가리켜야 합니다.

JVM 서버에 대한 자세한 정보는 시작하기의 CICS에 Java 지원의 내용을 참조하십시오.

Axis2 헤더 핸들러

기존 헤더 처리 프로그램을 사용할 수 있지만, SOAP 헤더를 처리하기 위해 Java에서 Axis2 핸들러를 작성하는 것이 더 효율적입니다. 이러한 핸들러는 JVM 서버에서 실행되고 따라서 오프로드에 적합합니다. Axis2 핸들러 작성에 대한 자세한 정보는 사용자 자체 Axis2 모듈 작성을 참조하십시오.

헤더 핸들러 프로그램은 Axis2 API를 사용하여 Axis2 환경, SOAP 메시지 및 개별 웹 서비스를 수정하거나 이와 상호작용할 수 있습니다. CICS가 엔진을 올바르게 실행할 수 없다는 것을 의미하는 방법으로 Axis2를 변경하는 것처럼 이러한 API를 사용하여 Axis2를 사용자 정의하지 마십시오. Axis2 핸들러는 CICS가 Axis2를 사용하는 방법과 호환 가능한 방법으로 Axis2 환경과 상호작용하는 경우에만 지원됩니다.

Axis2 저장소

Axis2는 저장소를 사용하여 모든 해당 구성 파일, 서비스 및 모듈을 저장합니다. CICS에서는 z/OS UNIX의 `usshome/lib/pipeline/repository` 디렉토리에 기본 저장소를 제공합니다. 여기서, `usshome`은 **USSHOME** 시스템 초기화 매개변수의 값입니다.

기본 저장소에는 Axis2를 사용하기 위해 CICS가 요청하는 구성 파일(`axis2.xml`)이 포함됩니다. 이 파일은 저장소의 `/conf` 서브디렉토리에 있습니다. 사용자 자체 저장소를 작성하는 경우, Axis2에 대해 작업하도록 이 파일을 CICS에 대한 저장소로 복사해야 합니다.

핸들러 프로그램을 등록하지 않으면 `axis2.xml` 파일을 편집하지 마십시오. 이 파일은 CICS의 내부 파트로 관리됩니다. 따라서, IBM 지원 센터에 의해 수행하지 않는다면 이 파일에 다른 변경사항을 작성하지 않아야 합니다.

웹 서비스에 대한 데이터 형식

다른 CICS 기술은 동등하게 부합하는 스펙이지만 실제로는 다른 JSON 및 XML을 생성할 수 있습니다. 데이터의 유효성을 검증하기 위해 검사를 적용하는 주문의 결과로 이는 다른 방식으로 입력 메시지에 발견되는 오류도 보고할 수 있습니다.

CICS는 JSON 및 XML 데이터를 자동으로 변환하기 위해 여러 다른 기술을 사용합니다. 여기에는 CICS용 z/OS 연결(Java 및 비Java 변형 모두), Axis2 및 PIPELINE 자원이 포함됩니다. 이러한 기술은 관련 스펙에 의해 설명되는 것처럼 외부 형식으로 JSON 및 XML 데이터를 생성합니다.

동등하게 부합하는 스펙이지만 실제로는 다른 데이터를 표시하기 위한 여러 방식이 있을 수 있습니다. CICS 기술은 부합하는 데이터를 항상 생성하지만, 이들 사이에 실제 차이가 있을 수 있습니다. 예를 들어, Java와 JSON에 대한 CICS용 비Java z/OS 연결 옵션 사이를 교환하는 경우, 생성된 JSON에서 작은 차이점을 감지할 수 있습니다.

그러한 차이점에는 실패 조건에서 보고되는 다른 오류 메시지, 공백 문자가 삽입되는 방법의 차이점, 숫자 데이터에 대한 대체(그러나 동등한) 표시 및 특수 문자가 이스케이프되는 방법의 변화가 포함됩니다. 이 네이처의 추가 변경사항은 교정 유지보수를 CICS에 적용하는 것의 결과로서 또는 CICS의 새 릴리스를 사용하여 소개될 수 있습니다.

JSON 클라이언트와 같은 파트너 시스템은 이 네이처의 스펙 부합 변화를 용인하기 위해 작성되어야 합니다. 파트너는 가끔 JSON 및 XML과의 상호작용을 위해 넓게 사용되는 협력 산업 라이브러리와 기술을 사용합니다. 그러한 라이브러리는 그와 같은 작은 형식화 차이점을 자동으로 처리합니다. 그러나, 덜 부합하는 파트너 시스템이 다양한 CICS 기술에서 서식 차이점을 감지하고 이에 다르게 응답할 수 있습니다. 따라서, 표준 기반 구문 분석기의 이점 없이 JSON 또는 XML에 대해 직접적으로 작업하는 애플리케이션을 작성 중인 경우 주의가 요구됩니다.

JSON 요청에 대한 서비스 제공자로서의 CICS

외부 JSON 클라이언트에 서비스를 제공하기 위한 CICS의 경우, 이는 요청을 수신해야 하고 파이프라인을 통해 이를 대상 애플리케이션 프로그램으로 전달해야 합니다. 애플리케이션으로부터의 응답은 동일한 파이프라인을 통해 JSON 클라이언트로 리턴됩니다.

JSON 요청을 위해 서비스 제공자로서 CICS를 구성하는 3가지 방법이 있습니다.

- CICS용 z/OS 연결 사용.

CICS용 z/OS 연결은 JSON 기술을 사용하여 z/OS 자산(예: CICS 프로그램)에 액세스하기 위한 기술입니다. z/OS용 z/OS 연결 및 CICS용 z/OS 연결과 CICSJava 파이프라인 사이의 일부 차이점에 대한 자세한 정보는 의 내용을 참조하십시오.

- CICS Java 파이프라인 사용.

CICS Java 파이프라인은 JSON을 사용하여 CICS 프로그램에 액세스하도록 허용하기 위해 CICS의 이전 버전에서 제공되는 기술입니다. CICS Java 파이프라인에 대한 자세한 정보는 구성에서의 Java 기반 SOAP 파이프라인의 내용을 참조하십시오.

- CICS Liberty JVM 서버의 JAX-RS 및 JSON 기능을 직접적으로 사용.

CICS는 JSON 데이터를 수신하고 이를 CICS 애플리케이션 프로그램에 의해 이해되는 구조화된 애플리케이션 데이터로 변환합니다. CICS 애플리케이션으로부터의 응답은 아웃바운드 응답에 대한 JSON 페이로드로 변환됩니다. 변환에는 메시지의 구문 분석이 필요합니다. CICS용 z/OS 연결을 사용하는 경우, Java 기반 JSON 구문 분석기 또는 비Java 등가를 사용하는 옵션을 가지고 있습니다. 구성 옵션이 CICS용 z/OS 연결 파이프라인 구성 파일에 지정됩니다. CICS용 z/OS 연결 구성에 대한 자세한 정보는 의 내용을 참조하십시오. CICS Java 파이프라인을 사용하는 경우, 구문 분석은 JVM 서버 내에서 Java를 사용하여서만 수행됩니다. 다른 구성 의사결정의 함축은 다음과 같습니다.

- Java를 사용하여 JSON을 구문 분석하면 처리는 사용 가능한 경우 zEnterprise Application Assist Processor(zAAP)에 대한 오프로드에 적합합니다. 처리를 오프로드하면 비용 효과를 가져올 수 있습니다.
- CICS용 z/OS 연결 비Java 구문 분석기를 사용하는 경우, 일부 워크로드가 성능 및 처리량 이점을 가져올 수 있습니다. 이점이 될 수 있는 워크로드에 대한 자세한 정보는 이 릴리스로 사용 가능한 성능 자료를 참조하십시오. 비Java 구문 분석기를 사용할지라도 대부분의 CICS용 z/OS 연결 인프라 처리는 zEnterprise Application Assist Processor(zAAP)에 오프로드하기 위해 적합합니다.

CICS용 z/OS 연결을 사용하여 JSON 요청에 대한 서비스 제공자로 CICS
 외부 JSON 클라이언트에 서비스를 제공하기 위한 CICS의 경우, 이는 CICS용 z/OS
 연결로 요청을 수신하고, JSON 메시지를 변형시키며 이를 대상 애플리케이션 프로그
 램으로 전달해야 합니다. 애플리케이션으로부터의 응답은 동일한 메커니즘을 통해 JSON
 클라이언트로 리턴됩니다.

그림 23에서는 CICS가 CICS용 z/OS 연결을 사용하는 서비스 제공자일 때 외부 JSON
 클라이언트에서 요청을 처리하기 위해 필요한 자원 및 아키텍처의 예제 구성이 표시됩
 니다.

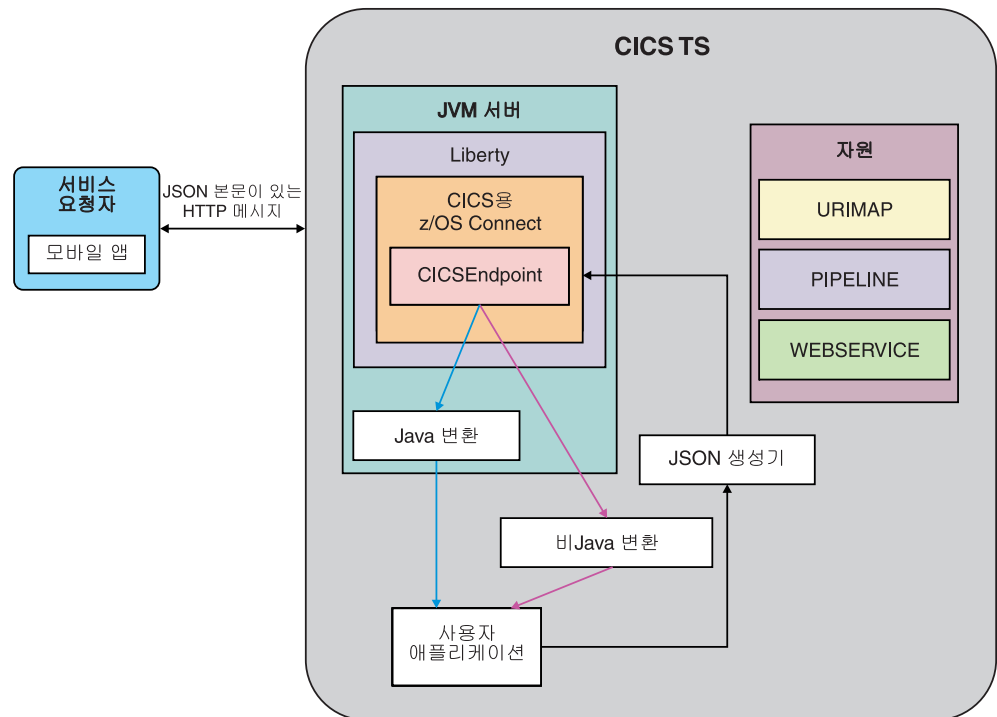


그림 23. CICS용 z/OS 연결을 사용하는 JSON 서비스 제공자에 대한 아키텍처 및 자원

CICS용 z/OS 연결을 가진 JSON 요청 처리

요청을 처리하기 위해 CICS는 다음 조작을 완료합니다.

1. 서비스 요청자로부터 요청을 수신하십시오.

WebSphere Liberty는 요청을 수신하고 이를 CICS용 z/OS 연결에 전달합니다.

2. CICS는 JVMSERVER로 설정된 USAGE 속성으로 URIMAP 정의를 스캐닝하여
 요청에 대한 URIMAP 자원을 해결합니다. URIMAP는 사용되는 WEBSERVICE
 자원을 식별합니다. 새 CICS 태스크는 URIMAP으로부터 트랜잭션 ID를 사용하
 여 요청을 처리하기 위해 시작됩니다. 기본적으로, CPIH 트랜잭션이 사용됩니다.

WEBSERVICE 자원은 CICS가 수행하는 처리를 제어합니다. 특히, WEBSERVICE 자원에 의해 지정되는 WSBInd 파일이 JSON과 구조화된 애플리케이션 데이터 사이에 데이터 변환에 사용됩니다. JSON 웹 서비스에 대한 WSBInd 파일은 유틸리티 DFHLS2JS 및 DFHJS2LS를 사용하여 생성됩니다.

참고: 스키마에 대한 JSON 데이터의 런타임 유효성 검증이 지원되지 않습니다. JSON 페이로드와 함께 사용되는 WEBSERVICE 자원의 VALIDATION 속성 값은 무시됩니다.

적용하는 제한사항에 대한 정보는 JSON 웹 서비스 제한의 내용을 참조하십시오.

3. CICS용 z/OS 연결은 구성되는 방법에 따라 요청을 처리합니다. CICS용 z/OS 연결이 글로벌 인터셉터를 사용하기 위해 구성되는 경우, 인터셉터가 이 처리 동안 실행됩니다.
4. CICSEndpoint가 제어를 수신합니다. JSON 페이로드는 파이프라인 구성 파일에 지정된 구성 옵션에 따라 구조화된 애플리케이션 데이터로 변형됩니다. 변환이 수행되는 방법에는 두 가지 옵션이 있습니다.
 - Java 변환은 JVM 서버 내에서 수행됩니다. 이 처리는 하나가 사용 가능한 경우 zAAP 프로세서에 대한 오프로드에 책임이 있습니다.
 - 비Java 변환은 JVM 서버의 밖에 수행되고, 특정 워크로드에 성능과 처리량 수익성을 제공할 수 있습니다. 이점이 될 수 있는 워크로드에 대한 자세한 정보는 이 릴리스로 사용 가능한 성능 자료를 참조하십시오.

맵핑은 WSBInd 파일의 정보에 따라 수행됩니다. 변환으로부터의 결과물은 둘 다 경우에 동등합니다.

5. CICS가 애플리케이션 프로그램에 링크되고 변환된 데이터를 전달합니다. 프로그램이 이 입력을 처리하고 JSON 생성기에 응답을 리턴합니다.
6. JSON 생성기는 5단계로부터 결과물을 사용하여 JSON 응답 메시지를 생성합니다. 이 메시지는 CICS용 z/OS 연결을 통해 서비스 요청자에 다시 전송됩니다.

z/OS Connect for CICS에서는 표준 RESTful 메소드를 지원하는 RESTful 인터페이스를 제공합니다.

POST

PUT

GET

DELETE

HEAD

적당한 곳에서, 이 인터페이스는 보통 JSON 요청이 구성되는 것과 동일한 방식으로 파이프라인이 구성되는 변환 및 생성기를 사용합니다. RESTful JSON 웹 서비스에 대한 자세한 정보는 시작하기의 RESTful JSON 웹 서비스 개념의 내용을 참조하십시오.

대부분의 CICS용 z/OS 연결 인프라 처리는 zEnterprise Application Assist Processor(zAAP)에서 오프로드하기 위해 적합합니다.

CICS Java 파이프라인을 사용하는 JSON 요청에 대한 서비스 요청자로서 CICS

외부 JSON 클라이언트에 서비스를 제공하기 위한 CICS의 경우, 이는 요청을 수신해야 하고 파이프라인을 통해 이를 대상 애플리케이션 프로그램으로 전달해야 합니다. 애플리케이션으로부터의 응답은 동일한 파이프라인을 통해 JSON 클라이언트로 리턴됩니다. JSON 변환은 JAVA 서버 내에서 Java를 사용하여 수행됩니다.

그림 24에서는 CICS가 Java 파이프라인을 사용하는 서비스 제공자일 때 외부 JSON 클라이언트로부터 요청을 처리하기 위해 필요한 아키텍처 및 자원의 예제 구성을 표시합니다. JSON 요청에 대한 파이프라인 처리는 CICS가 Java 파이프라인에서 SOAP 요청을 처리하는 방식과 유사합니다. 자세한 정보는구성에서의 Java 기반 SOAP 파이프라인의 내용을 참조하십시오.

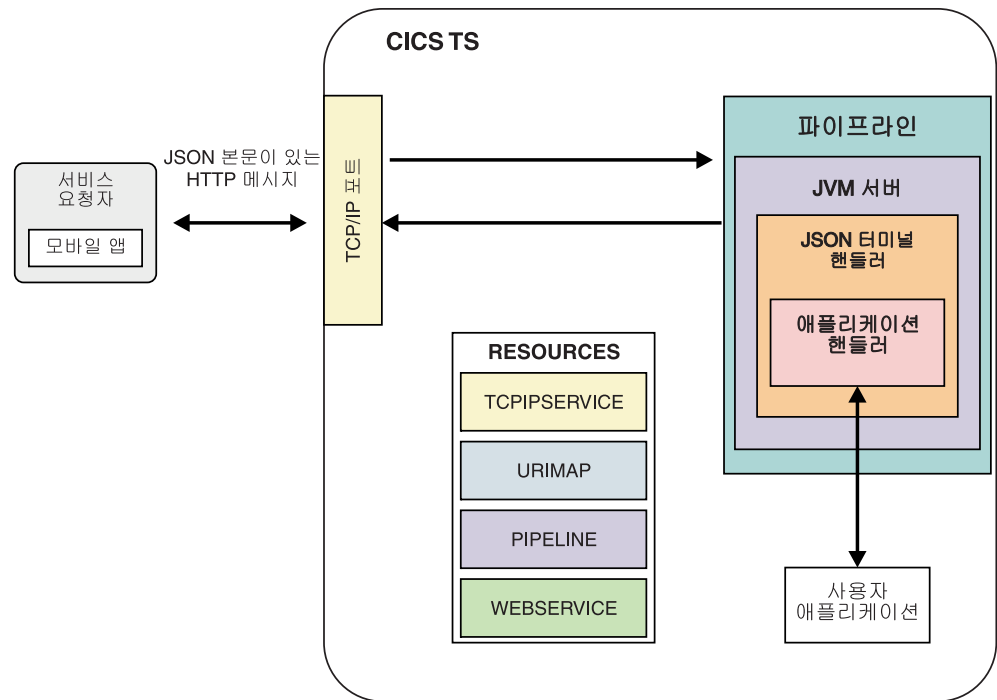


그림 24. CICSJava 파이프라인을 사용하는 JSON 서비스 제공자에 대한 아키텍처 및 자원

JSON 요청 처리

요청을 처리하기 위해 CICS는 다음 조작을 완료합니다.

1. 서비스 요청자로부터 요청을 수신하십시오.

TCPIPService 자원은 요청 수신을 위한 포트를 지정합니다. 이 포트는 CICS 제공 소켓 리스너 태스크(CSOL)에 의해 모니터링됩니다.

2. 요청을 검사하고 대상 애플리케이션 프로그램과 관련된 콘텐츠를 추출하십시오.

요청 메시지가 적절한 포트에서 수신될 때, URIMAP 자원 정의는 해당 USAGE 속성을 PIPELINE로 설정하도록 하고 해당 PATH 속성을 요청에서 찾은 URI로 설정하도록 하는 URIMAP 정의에 대해 스캐닝됩니다. 적절한 URIMAP 정의가 발견되면, URIMAP 정의의 PIPELINE 및 WEBSERVICE 속성에서 PIPELINE 및 WEBSERVICE 정의가 사용됩니다. URIMAP 정의의 TRANSACTION 속성은 파이프라인을 처리하기 위해 첨부되어야 하는 트랜잭션의 이름을 판별합니다. 기본적으로, CPIH 트랜잭션이 사용됩니다. URIMAP 정의는 사용할 PIPELINE 및 WEBSERVICE 자원을 또한 식별합니다.

이러한 PIPELINE 및 WEBSERVICE 자원은 CICS가 수행하는 처리를 제어합니다. 특히, WEBSERVICE 자원에 의해 지정되는 WSBIND 파일이 JSON과 언어 구조 사이에 데이터 변환에 사용됩니다. JSON 웹 서비스에 대한 WSBIND 파일은 유틸리티 DFHLS2JS 및 DFHJS2LS를 사용하여 생성됩니다.

참고: 스키마에 대한 JSON 데이터의 런타임 유효성 검증이 지원되지 않습니다. JSON 페이로드와 함께 사용되는 WEBSERVICE 자원의 VALIDATION 속성 값은 무시됩니다.

적용하는 제한사항에 대한 정보는 JSON 웹 서비스 제한의 내용을 참조하십시오.

3. 파이프라인 처리가 시작되고 정의되는 핸들러를 통해 요청이 플로우합니다. SOAP 웹 서비스에 대해 CICS가 현재 제공하는 핸들러가 JSON 웹 서비스와 관련된다는 것이 예상되지 않습니다.

4. 파이프라인의 종료로 JSON 터미널 핸들러가 호출됩니다. 이 터미널 핸들러는 Axis2 파이프라인과 인터페이스하는 Java 프로그램입니다. 터미널 핸들러는 Axis2 구성의 필수 설정을 수행하고 HTTP 요청 본문으로 Axis2 엔진을 시작합니다. Axis2 파이프라인 내에서 JSON 본문(있는 경우)이 구문 분석되고 콘텐츠를 표시하는 Java 오브젝트 모델이 구성됩니다. CICS가 애플리케이션 핸들러를 호출합니다. 애플리케이션 핸들러의 기본 역할은 요청의 Java 오브젝트 모델 표시를 애플리케이션 데이터에 매핑하는 것입니다. 이 매핑은 WSBIND 파일에서 언어 구조의 설명을 사용하여 수행됩니다.

5. 요청으로부터 추출된 데이터를 전달하여 애플리케이션 프로그램을 호출하십시오.

그런 다음, 애플리케이션 핸들러가 애플리케이션 프로그램에 링크됩니다. 프로그램이 이 입력을 처리하고 애플리케이션 핸들러에 응답을 리턴합니다.

6. 애플리케이션 프로그램이 리턴하는 데이터를 사용하여 응답을 구성하고 서비스 요청자로 응답을 전송하십시오.

애플리케이션 핸들러 및 메시지 핸들러는 서비스 제공자 애플리케이션으로부터 수신한 응답 메시지를 원래 요청의 형식의 메시지로 변환합니다. 이 메시지는 서비스 요청자로 다시 전송됩니다.

파이프라인 내의 일부 처리는 zEnterprise Application Assist Processor(zAAP)으로 오프로드하기에 적합합니다.

SOAP 서비스 제공자를 위한 CICS 인프라 작성

SOAP 서비스 제공자를 위한 CICS 인프라를 작성하려면 파이프라인 구성 파일을 작성하고 다수의 CICS 자원을 작성해야 합니다.

시작하기 전에

Java 파이프라인을 사용하려면 JVMSERVER 자원이 JVM 프로파일에서 JAVA_PIPELINE=YES 옵션이 지정된 상태로 존재하는지 확인하십시오.

JVM 서버는 여러 Java 파이프라인의 SOAP 처리를 처리할 수 있습니다.

이 태스크 정보

CICS 또는 CICSplex SM 기능을 사용하여 로컬 CICS 리전에서 PIPELINE 자원을 정의할 수 있습니다. 또는 CICS Explorer를 사용하여 로컬 CICS 리전 또는 CICS 번들에서 PIPELINE 자원을 정의할 수 있습니다. CICS Explorer를 사용하여 CICS 번들에서 PIPELINE 자원을 정의할 때 파이프라인 구성 파일을 작성하고 파일을 개별적으로 관리하지 않아도 되도록 작성한 파일을 CICS 번들로 패키징합니다. PROGRAM 자원 및 WEBSERVICE 자원은 CICS 번들에서 정의할 수도 있습니다. CICS 번들에서 WEBSERVICE 자원을 정의할 때 웹 서비스 바인딩 파일 및 WSDL 문서 또는 WSDL 아카이브 파일을 가져와서 번들에 포함시킬 수 있습니다. 또한 웹 서비스 지원을 위한 URIMAP 정의를 작성하고 이러한 정의를 번들로 패키징할 수 있습니다. CICS Explorer를 사용하여 CICS 번들에서 자원을 작성하고 편집하는 작업에 대한 자세한 도움말은 CICS Explorer 제품 문서의 번들에 대한 작업의 내용을 참조하십시오.

프로시저

1. 전송 인프라를 정의하십시오.
 - a. WebSphere MQ 전송을 사용하는 경우 처리될 때까지 입력 메시지를 저장하는 한 개 이상의 로컬 큐 및 입력 메시지를 처리하는 CICS 트랜잭션을 지정하는 한 개의 트리거 프로세스를 정의해야 합니다.
 - 1) 세부사항은 구성에서 WebSphere MQ 전송을 사용하도록 CICS 구성의 내용을 참조하십시오.
 - b. HTTP 전송을 사용하는 경우 인바운드 요청을 수신하는 포트를 정의하는 TCPIP SERVICE 자원을 정의해야 합니다.

- 1) 세부사항은 구성에서 웹 서비스를 위한 CICS 자원의 내용을 참조하십시오.
2. 옵션: 필요한 각 전송 구성에 대해 이 단계를 반복하십시오.
3. 인바운드 웹 서비스 요청 및 그에 대한 응답을 처리하려면 파이프라인 구성 파일에 포함시킬 메시지 핸들러 및 헤더 처리 프로그램을 정의하십시오. CICS는 다음과 같은 핸들러 및 헤더 처리 프로그램을 제공합니다.
 - a. SOAP 1.1 또는 1.2 메시지 처리를 위한 SOAP 메시지 핸들러. 서비스 제공자 파이프라인에서는 한 레벨의 SOAP만 지원할 수 있습니다.
 - b. MTOM/XOP 스펙을 준수하는 MIME 다중 파트/관련 메시지 처리를 위한 MTOM 핸들러
 - c. 보안 웹 서비스 메시지 처리를 위한 보안에서 웹 서비스 보안 지원
 - d. 원자적 트랜잭션 메시지 처리를 위한 구성에서 웹 서비스 트랜잭션 지원
4. 옵션: 파이프라인에서 자체 처리를 수행하려는 경우 메시지 핸들러 또는 헤더 처리 프로그램을 작성해야 합니다. 세부사항은 구성에서 메시지 핸들러의 내용을 참조하십시오. 사용자 정의 메시지 핸들러 프로그램을 작성하려고 결정한 경우 성능을 최적화하기 위해 스레드세이프(threadsafe)로 작성해야 합니다.
5. 메시지 핸들러, 헤더 처리 프로그램 및 애플리케이션 핸들러를 포함하는 XML 파이프라인 구성 파일을 작성하십시오.
 - a. CICS는 두 개의 기본 제공자 모드 파이프라인 구성 파일 샘플인 `basicsoap11provider.xml` 및 `basicsoap11javaprovider.xml`을 제공합니다.
 - b. 이러한 샘플을 편집하거나 추가 메시지 핸들러를 적절하게 추가할 수 있습니다. 샘플은 `/usr/lpp/cicsts/cicsts53/samples/pipelines` 라이브러리(여기서, `/usr/lpp/cicsts/cicsts53`는 z/OS UNIX에서 CICS 파일의 기본 설치 디렉토리임)에서 제공됩니다.
 - c. 파이프라인 구성 파일에서 사용 가능한 옵션에 대한 자세한 정보는 구성에서 파이프라인 구성 파일의 내용을 참조하십시오.
6. 파이프라인 구성 파일을 z/OS UNIX의 적절한 디렉토리에 복사하십시오.
7. CICS 리전에서 파일을 읽을 수 있도록 파이프라인 구성 파일 권한을 변경하십시오.
8. 필요한 각 파이프라인 구성에 대해 5단계 - 7단계를 반복하십시오.
9. PIPELINE 자원을 작성하십시오.
 - a. PIPELINE 자원은 파이프라인 구성 파일의 위치를 정의합니다. 또한 웹 서비스 바인딩 파일 및 선택적으로 WSDL을 포함하는 z/OS UNIX 디렉토리인 펍업 디렉토리도 지정합니다.
 - b. 각 파이프라인 구성에 대해 이 단계를 반복하십시오.

- a. PIPELINE 자원을 작성할 때 CICS는 지정된 픽업 디렉토리에서 파일을 읽고 WEBSERVICE 자원 및 URIMAP 자원을 동적으로 작성합니다.
- 10. 자동으로 설치된 PROGRAM 정의를 사용하지 않는 경우 파이프라인에서 실행되는 각 프로그램에 대해 PROGRAM 자원을 작성하십시오. 여기에는 일반적으로 트랜잭션 CPIH에서 실행되는 대상 애플리케이션 프로그램이 포함됩니다. 이 트랜잭션은 속성 TASKDATALOC(ANY)로 정의됩니다. 따라서 프로그램을 링크 편집할 때에는 AMODE(31) 옵션을 지정해야 합니다.

결과

이제 CICS 시스템에 각 서비스 제공자에게 필요한 인프라가 포함되어 있습니다.

다음에 수행할 작업

추가 전송 인프라를 정의하거나 추가 파이프라인을 작성하기 위해 필요한 경우에는 구성을 확장할 수 있습니다.

SOAP 서비스 요청자를 위한 CICS 인프라 작성

SOAP 서비스 요청자를 위한 CICS 인프라를 작성하려면 파이프라인 구성 파일을 작성하고 다수의 CICS 자원을 작성해야 합니다.

시작하기 전에

Java 파이프라인을 사용하려면 JVMSERVER 자원이 JVM 프로파일에서 JAVA_PIPELINE=YES 옵션이 지정된 상태로 존재하는지 확인하십시오. 참조 -> 시스템 정의에서 JVMSERVER 자원의 내용을 참조하십시오.

JVM 서버는 여러 Java 파이프라인의 SOAP 처리를 처리할 수 있습니다.

이 태스크 정보

CICS 또는 CICSplex SM 기능을 사용하여 로컬 CICS 리전에서 PIPELINE 자원을 정의할 수 있습니다. 또는 CICS Explorer를 사용하여 로컬 CICS 리전 또는 CICS 번들에서 PIPELINE 자원을 정의할 수 있습니다. CICS Explorer를 사용하여 CICS 번들에서 PIPELINE 자원을 정의할 때 파이프라인 구성 파일을 작성하고 파일을 개별적으로 관리하지 않아도 되도록 작성한 파일을 CICS 번들로 패키징합니다. PROGRAM, WEBSERVICE 및 URIMAP 자원도 CICS 번들에서 정의할 수 있습니다. CICS 번들에서 WEBSERVICE 자원을 정의할 때 웹 서비스 바인딩 파일 및 WSDL 문서 또는 WSDL 아카이브 파일을 가져와서 번들로 패키징하고 서비스 제공자에 대해 PROGRAM 정의를 포함시키도록 선택할 수 있습니다. 또한 웹 서비스 지원을 위한 URIMAP 정의를 작성하고 이러한 정의를 번들로 패키징할 수 있습니다. CICS Explorer를 사용하여 CICS 번들에서 자원을 작성하고 편집하는 작업에 대한 자세한 도움말은

CICS Explorer 제품 문서의 번들에 대한 작업의 내용을 참조하십시오.

프로시저

1. 인바운드 웹 서비스 요청 및 그에 대한 응답을 처리하려면 파이프라인 구성 파일에 포함시킬 메시지 핸들러 및 헤더 처리 프로그램을 정의하십시오. CICS는 다음과 같은 핸들러 및 헤더 처리 프로그램을 제공합니다.
 - a. SOAP 1.1 또는 1.2 메시지 처리를 위한 SOAP 메시지 핸들러 서비스 요청자 파이프라인에서는 한 레벨의 SOAP만 지원할 수 있습니다.
 - b. TOM/XOP 스펙을 준수하는 MIME 다중 파트/관련 메시지 처리를 위한 MTOM 핸들러
 - c. 보안 웹 서비스 메시지 처리를 위한 보안 핸들러
 - d. 원자적 트랜잭션 메시지 처리를 위한 WS-AT 헤더 처리 프로그램
2. 옵션: 파이프라인에서 자체 처리를 수행하려는 경우 메시지 핸들러 또는 헤더 처리 프로그램을 작성해야 합니다. 세부사항은 329 페이지의 『메시지 핸들러』의 내용을 참조하십시오. 사용자 정의 메시지 핸들러 프로그램을 작성하려고 결정한 경우 성능을 최적화하기 위해 스레드세이프(threadsafe)로 작성해야 합니다.
3. 메시지 핸들러 및 헤더 처리 프로그램을 포함하는 XML 파이프라인 구성 파일을 작성하십시오. CICS는 사용자가 적절하게 복사하고 편집할 수 있는 두 개의 기본 요청자 모드 파이프라인 구성 파일 샘플인 `basicsoap11provider.xml` 및 `basicsoap11javaprovider.xml`을 제공합니다. 이러한 샘플은 `/usr/lpp/cicsts/cicsts53/samples/pipelines` 라이브러리(여기서, `/usr/lpp/cicsts/cicsts53`는 z/OS UNIX에서 CICS 파일을 위한 기본 설치 디렉토리임)에서 제공됩니다. 파이프라인 구성 파일에서 사용 가능한 옵션에 대한 자세한 정보는 269 페이지의 『파이프라인 구성 파일』의 내용을 참조하십시오.
4. 파이프라인 구성 파일을 z/OS UNIX의 적절한 디렉토리에 복사하십시오.
5. CICS 리전에서 파일을 읽을 수 있도록 파이프라인 구성 파일 권한을 변경하십시오.
6. 필요한 각 파이프라인 구성에 대해 3단계 - 5단계를 반복하십시오.
7. PIPELINE 자원을 작성하십시오. 참조 -> 시스템 정의에서 PIPELINE 자원의 내용을 참조하십시오. PIPELINE 자원은 파이프라인 구성 파일의 위치를 정의합니다. 또한 웹 서비스 바인딩 파일 및 선택적으로 WSDL을 포함하는 z/OS UNIX 디렉토리인 픽업 디렉토리도 지정합니다. CICS가 웹 서비스 제공자로부터 응답을 대기하는 시간을 판별하는 제한시간을 초 단위로 지정할 수도 있습니다. 각 파이프라인 구성 파일에 대해 이 단계를 반복하십시오. PIPELINE 자원을 작성할 때 CICS는 픽업 디렉토리에서 지정된 파일을 읽고 WEBSERVICE 자원을 동적으로 작성합니다(참조 -> 시스템 정의에서 WEBSERVICE 자원 참조).
8. 자동 설치 PROGRAM 정의를 사용하지 않는 경우 파이프라인에서 실행되는 각 프로그램에 대해 PROGRAM 자원을 작성하십시오. 참조 -> 시스템 정의에서 PROGRAM 자원의 내용을 참조하십시오. 이러한 프로그램에는 일반적으로 트랜잭

션 CPIH에서 실행되는 서비스 요청자 애플리케이션 프로그램이 포함됩니다. 이 트랜잭션은 속성 TASKDATALOC(ANY)로 정의됩니다. 따라서 프로그램을 링크 편집할 때에는 AMODE(31) 옵션을 지정해야 합니다.

9. 옵션: 애플리케이션 개발에서 HTTP 클라이언트로 CICS에 대한 URIMAP 자원 작성의 지시사항에 따라 서비스 요청자가 요청을 작성하는 데 사용하는 각 URI에 대한 클라이언트 요청의 URIMAP 자원(참조 -> 시스템 정의에서 URIMAP 자원 참조)을 작성하십시오. URIMAP 자원을 사용하는 대신 프로그램에서 **INVOKE SERVICE** 명령에 대한 URI를 직접 지정할 수 있습니다. 그러나 URIMAP 자원의 사용은 서비스 제공자의 URI가 변경되는 경우 사용자 애플리케이션을 다시 컴파일하지 않아도 됨을 의미합니다. 후속 요청을 위해 애플리케이션 또는 같은 서비스를 호출하는 다른 애플리케이션이 클라이언트 연결을 재사용할 수 있도록 URIMAP 자원을 통해 CICS가 클라이언트 연결 사용 후 이 연결을 열린 상태로 유지하는 연결 풀 구현을 선택할 수도 있습니다.

결과

이제 CICS 시스템에 각 서비스 요청자에게 필요한 인프라가 포함되어 있습니다.

다음에 수행할 작업

추가 파이프라인을 작성하기 위해 필요한 경우 구성을 확장할 수 있습니다.

JSON 서비스 제공자를 위한 CICS 인프라 작성

JSON 서비스 제공자를 위한 CICS 인프라를 작성하려면 파이프라인 구성 파일을 작성하고 다수의 CICS 자원을 작성해야 합니다.

시작하기 전에

CICS를 JSON 요청자의 서비스 제공자로 사용하거나 링크 가능한 인터페이스를 사용하여 JSON을 변환하려면 **JAVA_PIPELINE=YES** 옵션이 지정된 JVM 프로파일을 사용하여 JVMSERVER 자원을 정의하고 설치하십시오. DFHAXIS라고 하는 예제 JVMSERVER 자원 정의는 DFH\$AXIS 그룹에서 제공됩니다.

참고: 여기에 설명된 인프라는 사용자가 JSON 서비스 제공자에게 연결하기 위해 CICS용 z/OS Connect를 사용하고 않는 것으로 가정하고 그 결과로서 JVM 서버 내 Java 구문 분석을 사용하여 JSON 메시지를 구문 분석합니다. 비Java JSON 구문 분석을 사용하려는 경우 CICS용 z/OS Connect를 사용하여 JSON 웹 서비스에 연결해야 합니다. CICS용 z/OS Connect 설정에 대한 자세한 정보는 의 내용을 참조하십시오.

프로시저

1. 전송 인프라를 정의하십시오.

- 인바운드 요청을 수신하는 포트를 정의하는 TCPIPService 자원을 정의하십시오. 세부사항은 웹 서비스를 위한 CICS 자원을 참조하십시오.
2. 인바운드 웹 서비스 요청 및 그에 대한 응답을 처리하려면 파이프라인 구성 파일에 포함시킬 메시지 핸들러를 정의하십시오.

파이프라인에서 자체 처리를 수행하려는 경우 메시지 핸들러를 작성해야 합니다. 세부사항은 메시지 핸들러의 내용을 참조하십시오. 사용자 정의 메시지 핸들러 프로그램을 작성하려고 결정한 경우 성능을 최적화하기 위해 스레드세이프(threadsafe)로 작성해야 합니다.

3. 메시지 핸들러, 헤더 처리 프로그램 및 애플리케이션 핸들러를 포함하는 XML 파이프라인 구성 파일을 작성하십시오. CICS는 기본 제공자 모드 파이프라인 구성 파일 샘플인 jsonjavaprovider.xml을 제공합니다. 이 샘플을 편집하여 추가 메시지 핸들러를 적절하게 추가할 수 있습니다. 이 샘플은 /usr/lpp/cicsts/cicsts53/samples/pipelines 디렉토리(여기서, /usr/lpp/cicsts/cicsts53는 z/OS UNIX에서 CICS 파일의 기본 설치 디렉토리임)에서 제공됩니다. 파이프라인 구성 파일에서 사용 가능한 옵션에 대한 자세한 정보는 287 페이지의 『서비스 제공자 및 서비스 요청자 파이프라인에서 사용되는 요소』의 내용을 참조하십시오.
4. 파이프라인 구성 파일을 z/OS UNIX의 적절한 디렉토리에 복사하십시오.
5. CICS 리전에서 파일을 읽을 수 있도록 파이프라인 구성 파일 권한을 변경하십시오.
6. PIPELINE 자원을 작성하십시오. PIPELINE 자원은 파이프라인 구성 파일의 위치를 정의합니다. 또한 웹 서비스 바인딩 파일을 포함하는 z/OS UNIX 디렉토리인 픽업 디렉토리도 지정합니다. 각 파이프라인 구성에 대해 이 단계를 반복하십시오. PIPELINE 자원을 설치하거나 PIPELINE SCAN을 수행할 때, CICS는 지정된 픽업 디렉토리에서 .wsbind 파일을 읽고 적절한 WEBSERVICE 및 URIMAP 자원을 동적으로 작성합니다.
7. 자동으로 설치된 PROGRAM 정의를 사용하지 않는 경우 파이프라인에서 실행되는 각 프로그램에 대해 PROGRAM 자원을 작성하십시오. 여기에는 일반적으로 트랜잭션 CPIH에서 실행되는 대상 애플리케이션 프로그램이 포함됩니다. 이 트랜잭션은 속성 TASKDATALOC(ANY)로 정의됩니다. 따라서 프로그램을 링크 편집할 때에는 AMODE(31) 옵션을 지정해야 합니다.

결과

각 서비스 제공자에게 필요한 인프라 작성을 완료했으며 이제 CICS 시스템에 이러한 자원을 설치할 수 있습니다.

다음에 수행할 작업

자원을 설치하십시오. 추가 전송 인프라를 정의하거나 추가 파이프라인을 작성하기 위해 필요한 경우에는 구성을 확장할 수 있습니다.

CICS용 z/OS 연결 구성

CICS용 z/OS 연결은 CICS 트랜잭션 서버의 일부로 분배됩니다. 사용자는 CICS용 z/OS 연결을 수동으로 구성할 수 있고, 이는 일반적으로 개별 서비스에 대한 추가 구성으로 한 번 수행됩니다.

시작하기 전에

CICS에서 구성되는 WebSphere Liberty Profile JVM 서버를 이미 가지고 있는지 여부를 고려하는 것이 중요합니다. 동일한 WebSphere Liberty Profile 환경에서 z/OS 연결 및 기타 관련 없는 서비스를 호스팅하는 것이 가능한 동안, z/OS 연결의 단독 사용을 위한 개별 JVM 서버를 구성하기 위해 좋은 관행입니다. 단일 CICS 리전에서 구성되는 단일 WebSphere Liberty Profile JVM 서버만 있는 것이 좋습니다. 자체 CICS 리전 또는 CICS 리전의 그룹에서 z/OS 연결을 호스팅할 수 있고 분배 프로그램 링크 메커니즘을 사용하여 CICS 리전을 소유하는 애플리케이션에서 CICS 프로그램을 호출할 수 있습니다.

프로시저

1. WebSphere Liberty Profile을 지원하기 위해 구성된 JVMSERVER를 작성하십시오. WebSphere Liberty Profile JVMSERVER 작성에 대한 자세한 정보는 의 내용을 참조하십시오.
2. WebSphere Liberty Profile에는 보안 요구사항을 충족시키기 위한 구성이 필요합니다. CICS 용 z/OS 연결에는 SSL의 사용이 필요합니다. 기본적으로 클라이언트 공인 SSL 인증서의 사용을 예상합니다. HTTP 기본 인증을 사용할 수 있도록 `server.xml` 파일에 다음 구성 옵션을 추가할 수 있습니다.

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

WebSphere Liberty Profile 보안에 대한 자세한 정보는 보안에서 Liberty JVM 서버에 대한 보안 구성의 내용을 참조하십시오.

3. WebSphere Liberty Profile에 대한 `server.xml` 파일을 찾고 `<featureManager>` 목록을 갱신하여 다음 예제에 표시된 대로 `<feature>cicsts:zosConnect-1.0</feature>` 기능을 포함시키십시오.

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

4. CICS용 z/OS 연결 수신기 프로그램을 정의하도록 `server.xml` 파일에 다음 명령문을 추가하십시오.

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

5. JVMSERVER를 설치하십시오. 생성된 messages.log 파일의 콘텐츠를 검토하십시오. 이 로그에는 CICS용 z/OS 연결에 의해 발행되는 메시지를 포함하여 WebSphere Liberty Server에 의해 생성되는 메시지가 포함됩니다. 예를 들어, 다음과 같습니다.

SRVE0169I: Loading Web Module: z/OS Connect.

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

오류 또는 경고 메시지에 대해 로그를 검사하십시오.

6. XML 파이프라인 구성 파일을 작성하십시오. 샘플 파이프라인 구성 파일 jsonzosconnectprovider.xml은 /usr/lpp/cicsts/cicsts53/samples/pipelines/디렉토리에서 제공됩니다. 여기서, /usr/lpp/cicsts/cicsts53는 z/OS UNIX에 CICS 파일을 위한 기본 설치 디렉토리입니다. Liberty JVM 서버(기본) 내에 Java를 사용하여 JSON을 구문 분석하거나 비Java JSON 구문 분석기를 사용하려는지 여부를 결정해야 합니다. 비Java JSON 구문 분석기를 사용하려는 경우, <provider_pipeline_json> 요소에 java_parser="no" 속성을 포함시켜야 합니다.

- Liberty JVM 서버 내에 Java를 사용하여 JSON을 구문 분석하려는 경우 샘플 파이프라인 구성 파일을 사용할 수 있지만 <jvmserver> 요소의 DFHWLP를 1단계의 JVMSERVER 이름으로 대체할 수 있습니다.
- 비Java 구문 분석기를 사용하여 JSON을 구문 분석하려면, 다음 예제에서처럼 <provider_pipeline_json> 요소에 java_parser="no" 속성을 추가하기 위한 샘플 구성 파일을 수정하십시오.

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

DFHWLP를 1단계의 JVMSERVER 이름으로 대체하십시오.

7. 파이프라인 구성 파일을 zFS의 적합한 디렉토리로 복사하고 파일 권한을 사용하여 CICS 리전이 파일을 읽을 수 있는지 확인하십시오.
8. PIPELINE 자원을 작성하십시오. PIPELINE 자원은 파이프라인 구성 파일의 위치를 정의합니다. 이는 선택적 웹 서비스 바인딩 디렉토리(픽업 디렉토리라고도 알려짐)도 지정합니다. 이는 웹 서비스 바인딩 .wsbind 파일을 포함하는 zFS 디렉토리입니다. PIPELINE 자원을 설치하거나 PIPELINE SCAN을 수행할 때, CICS는 지정된 픽업 디렉토리에서 .wsbind 파일을 읽고 적절한 WEBSERVICE 및 URIMAP 자원을 동적으로 작성합니다.

결과

CICS용 z/OS 연결 인스턴스가 구성됩니다. 이 URL을 웹 브라우저(https://hostname:portnumber/zosConnect/apim/services)에 입력하여 z/OS 연결에 대한

기본 구성을 테스트할 수 있습니다. 여기서, *hostname*은 CICS용 z/OS 연결을 호스팅하는 CICS 리전이 실행 중인 시스템의 호스트 이름 또는 IP 주소이고 *portnumber*는 *server.xml* 파일의 `<httpEndpoint>` 요소에 지정되는 **httpsPort**입니다. 웹 브라우저에는 설치된 서비스의 목록이 표시됩니다. 아직 설치된 서비스가 없으므로 목록이 비어 있습니다.

다음에 수행할 작업

JSON 웹 서비스를 CICS용 z/OS 연결에 설치할 준비가 되었습니다. 자세한 정보는 의 내용을 참조하십시오.

관련 개념:

CICS용 z/OS 연결로 시작하기

IBM z/OS 연결은 오늘의 모바일 계산 환경에 더 잘 참여하도록 z/OS에서 실행되는 시스템을 사용하는 소프트웨어입니다. CICS용 z/OS 연결은 CICS 프로그램이 JSON(JavaScript Object Notation) 인터페이스와 함께 호출될 수 있도록 합니다.

CICS용 z/OS 연결의 기능

CICS의 통합 z/OS 연결 기능에는 기타 환경의 z/OS 연결에 대한 차이가 있습니다. CICS용 z/OS 연결은 CICS에 대한 로컬 액세스를 위해 최적화되고 표준 CICS 서비스를 사용합니다.

CICS용 z/OS 연결에 대한 보안

z/OS 연결은 WebSphere Liberty Profile 애플리케이션이고 기타 WebSphere Liberty Profile 애플리케이션과 동일한 구성 및 고려사항을 가집니다. CICS용 z/OS 연결에는 일부 추가 요구사항이 있습니다.

관련 정보:

CICS JSON 웹 서비스에 대한 CICS용 z/OS 연결 구성

처음으로 CICS용 z/OS 연결을 구성하면 CICS JSON 웹 서비스에 대해 구성할 수 있습니다. JSON 웹 서비스는 다른 CICS PIPELINE 환경에 배치되는 방법에 유사한 방식으로 CICS용 z/OS 연결에 배치됩니다.

시작하기 전에

이 태스크를 시작하기 전에 263 페이지의 『CICS용 z/OS 연결 구성』을 완료해야 합니다. 배치하려는 각 서비스에 대한 JSON WSBIND 파일이 또한 필요합니다. 이는 JSON 지원, **DFHLS2JS** 및 **DFHJS2LS**를 사용하여 생성될 수 있습니다.

이 태스크 정보

JSON 웹 서비스는 다른 CICS PIPELINE 환경에 배치되는 방법에 유사한 방식으로 CICS용 z/OS 연결에 배치됩니다.

프로시저

- 사용자의 애플리케이션이 단순하면, PIPELINE 스캔 명령을 사용하여 CICS용 z/OS 연결에 대한 웹 서비스를 자동으로 구성할 수 있습니다. CICS용 z/OS 연결에 대해 구성되는 PIPELINE에 JSON WSBind 파일을 배치하고, **PIPELINE SCAN** 명령을 실행하십시오. 작성될 때 URI가 WSBind 파일과 연관되는 경우, 해당 URI는 CICS용 z/OS 연결로 동적으로 등록되고 추가 조치가 필요하지 않습니다. 사용자 **WEBSERVICE**가 동적으로 CICS용 z/OS 연결에 등록되면 이는 **CICSService_**로 접두부가 표시되는 **WEBSERVICE** 자원의 이름을 사용합니다. 예를 들어, TESTWEBS라는 **WEBSERVICE**가 CICS용 z/OS 연결에 대해 스캐닝되면 **CICSService_TESTWEBS**로 알려집니다. 사용자 애플리케이션이 더 복잡하면 추가 구성 단계가 필요합니다. 이는 다음 글머리 기호 지점 아래에 나열됩니다.
- 사용자 애플리케이션이 더 복잡하면 이를 배치하기 위해 다음 단계를 수행하십시오.
 1. WSBind 파일을 PIPELINE의 WSDIR에 두고 **PIPELINES sCAN** 명령을 실행하거나 WSBind 파일을 적절한 **PIPELINE** 자원과 연관시키는 **WEBSERVICE** 자원을 설치하십시오.
 2. **URIMAP** 자원이 **WEBSERVICE**에 대해 설치되는지 확인하십시오. **PIPELINE SCAN**을 사용하여 **WEBSERVICE**가 설치되고 서비스가 설치되는 URI를 표시하는 경우, **URIMAP**가 자동으로 작성됩니다. 그렇지 않으면, **URIMAP**이 수동으로 정의될 수 있습니다. **URIMAP**은 **USAGE(JVMSEVER)**를 지정하고 **PIPELINE** 및 **WEBSERVICE** 속성이 설정되도록 해야 합니다.
 3. Liberty JVM 서버 구성 파일(server.xml)을 찾으십시오. 서비스의 URI를 CICS용 z/OS 연결 수신기와 연관하기 위해 이 파일에 항목이 필요합니다. 후미 * 와일드카드의 사용은 많은 URI를 엔드포인트와 연관시키기 위해 사용될 수 있습니다. 적절한 이름 지정 규칙이 적용되는 경우 * 와일드카드의 사용이 이 단계를 불필요하게 할 수 있습니다. 예를 들어, 다음과 같습니다.

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

이 예제에서, "/json"으로 시작되는 모든 URI는 **CICSService1** 이름 아래에서 CICS용 z/OS 연결 수신기 프로그램과 연관됩니다. URI 초기 /json을 가진 후속 서비스에는 추가 구성이 필요하지 않습니다. 추가 항목을 추가할 수 있습니다. 각 항목에는 고유한 **serviceName** 및 **invokeURI**가 있지만, **serviceRef**에 대한 동일한 값을 사용하십시오. 사용될 **server.xml** 구성 파일에 대해 이 변경을 위해 Liberty JVM 서버를 재시작할 필요가 없습니다.

결과

URL(`https://hostname:portnumber/zosConnect/apim/services`)을 웹 브라우저에 입력하여 CICS용 z/OS 연결에 대한 구성을 테스트할 수 있습니다. *hostname*은 CICS용 z/OS 연결을 호스팅하는 CICS 리전의 이름 또는 IP 주소입니다. *portnumber*은 `server.xml` 구성 파일의 `<httpEndpoint>` 섹션에서 구성되는 **httpsPort**입니다. 웹 브라우저는 설치된 서비스 목록을 표시합니다.

서비스는 이전에 설명된 것과 동일한 *hostname* 및 *portnumber*를 사용하여 JSON 클라이언트로부터 호출될 준비가 됩니다. 나머지 URI는 CICS URIMAP 자원에 정의된 것처럼 형성됩니다. 요청이 Liberty JVM 서버에 도착할 때, `server.xml` 구성 파일에서 정보를 사용하는 CICS용 z/OS 연결 수신기와 연관됩니다. 새 CICS 태스크가 이 작업을 수행하기 시작하고 URIMAP 자원으로부터 정보를 사용하는 특정 **WEBSERVICE** 자원과 연관됩니다. 데이터 변환 프로세스가 Liberty JVM 서버에서 발생하고 `WSBind` 파일에서 이름 지정된 것처럼 대상 CICS 프로그램이 첨부됩니다.

관련 개념:

CICS용 z/OS 연결로 시작하기

IBM z/OS 연결은 오늘의 모바일 계산 환경에 더 잘 참여하도록 z/OS에서 실행되는 시스템을 사용하는 소프트웨어입니다. CICS용 z/OS 연결은 CICS 프로그램이 JSON(JavaScript Object Notation) 인터페이스와 함께 호출될 수 있도록 합니다.

CICS용 z/OS 연결의 기능

CICS의 통합 z/OS 연결 기능에는 기타 환경의 z/OS 연결에 대한 차이가 있습니다. CICS용 z/OS 연결은 CICS에 대한 로컬 액세스를 위해 최적화되고 표준 CICS 서비스를 사용합니다.

CICS용 z/OS 연결에 대한 보안

z/OS 연결은 WebSphere Liberty Profile 애플리케이션이고 기타 WebSphere Liberty Profile 애플리케이션과 동일한 구성 및 고려사항을 가집니다. CICS용 z/OS 연결에는 일부 추가 요구사항이 있습니다.

관련 태스크:

263 페이지의 『CICS용 z/OS 연결 구성』

CICS용 z/OS 연결은 CICS 트랜잭션 서버의 일부로 분배됩니다. 사용자는 CICS용 z/OS 연결을 수동으로 구성할 수 있고, 이는 일반적으로 개별 서비스에 대한 추가 구성으로 한 번 수행됩니다.

Java 파이프라인에서 CICS용 z/OS 연결로 JSON 웹 서비스 이동

Java 파이프라인에서 CICS용 z/OS 연결로 JSON 웹 서비스를 이동하려면 수행되어야 하는 단계가 있습니다.

시작하기 전에

Java 파이프라인에 배치되는 기존 JSON 웹 서비스가 필요합니다. 적절한 예제는 IBM Redbooks: 모바일 애플리케이션용 IBM CICS JSON 웹 서비스 구현 Redbooks 서적에서 이용될 수 있습니다.

이 태스크 정보

Java 파이프라인(CICS Transaction Server Feature Pack for Mobile Extensions V1.0과 함께 사용된 것처럼)에서 CICS용 z/OS 연결로 JSON 웹 서비스를 재배포하는 것은 단순한 과정입니다. CICS의 z/OS 연결 환경은 Java 파이프라인 기술과 호환 가능합니다. 개발되고 Java 파이프라인에 배치되는 아티팩트는 CICS용 z/OS 연결용으로 재배포될 수 있습니다. IBM Redbooks: 모바일 애플리케이션용 IBM CICS JSON 웹 서비스 구현 Redbooks 서적에는 JSON 예제가 있습니다. 이는 CICS용 z/OS 연결과 함께 사용될 수 있습니다.

Java 파이프라인 및 CICS용 z/OS 연결에 사용되는 WSBInd 파일이 동일한 도구(DFHLS2JS 및 DFHJS2LS)를 사용하여 생성되고 서로 완전히 호환 가능합니다.

CICS용 z/OS 연결 환경은 클라이언트 애플리케이션과 CICS 사이에서 SSL을 사용할 필요가 있습니다. 기존 Java 파이프라인 환경이 SSL을 사용하지 않으면, CICS용 z/OS 연결에 대한 변환에 추가 단계가 포함됩니다.

프로시저

1. 로부터 지시사항을 사용하여 CICS용 필수 z/OS 연결 인프라를 작성하십시오. 이 구성의 파트의 경우, Liberty JVM 서버가 연결 수신을 위해 청취하는 SSL TCP/IP 포트 번호를 선택합니다. 고려할 두 가지 옵션이 있습니다.
 - a. Java 파이프라인에 대해 TCPIP SERVICE가 사용하는 포트 번호로부터 다른 포트 번호를 선택하십시오. 이 옵션에는 양쪽 환경 모두가 다른 TCP/IP 포트에 동시에 설치될 수 있는지 확인한다는 이점이 있습니다. 이는 클라이언트 프로그램이 새 JSON 서비스를 대상으로 갱신할 필요가 있다는 것을 의미합니다. 이전 환경이 SSL을 사용하지 않으면, CICS용 z/OS 연결에 대한 변환에 URI에 대한 변경사항이 필요합니다. 따라서, 이 옵션이 더 적합합니다.
 - b. Java 파이프라인에 대해 TCPIP SERVICE를 사용하는 것과 동일한 포트 번호를 선택하십시오. 클라이언트 프로그램이 사용하는 URI의 포트 번호는 변경될 필요가 없지만 두 개의 환경은 동시에 설치될 수 없습니다. URI는 기타 이유(예: SSL을 사용할 수 있을 때 HTTP에서 HTTPS로 전환)로 변경될 수 있습니다.
2. CICS용 z/OS 연결에 WSBInd 파일을 배치하십시오. 기존 WSBInd 파일이 전체적으로 CICS용 z/OS 연결과 호환 가능합니다. 예 설명된 대로 CICS용 z/OS 연

결에 새 JSON 웹 서비스를 배치하려면 다음 단계를 따르십시오. CICS는 동일한 이름을 가진 두 WEBSERVICE 자원이 모두 설치되는 것을 허용하지 않습니다. 따라서, 두 가지 옵션이 있습니다.

- a. 새로운 것이 이전에 사용된 것과 동일한 이름으로 설치될 수 있도록 원래 WEBSERVICE를 버리십시오.
 - b. 충돌을 피하기 위해 새 WEBSERVICE의 이름을 바꾸십시오.
3. PIPELINE 핸들러 프로그램을 사용하여 Java 파이프라인의 처리를 사용자 정의하려면 해당 사용자 정의가 여전히 필요한지 여부를 고려하십시오. 해당 사용자 정의가 필요한 경우, 동등한 기능을 가진 z/OS 연결 인터셉터 프로그램을 작성하고 이를 글로벌 인터셉터로 배치하십시오. IBM HTTP Server 8.5.5 제품 문서의 z/OS Connect 인터셉터 정의 주제에 z/OS 연결 인터셉터에 대한 자세한 정보가 있습니다.

결과

새 CICS용 z/OS 연결 JSON 웹 서비스를 테스트할 준비가 되었습니다. Java 파이프라인에 사용되는 것과 동일한 포트 번호를 사용하여 서비스를 배치하는 경우, 보안 구성이 변경되지 않는다면(예: SSL 사용 중), 클라이언트는 변경되지 않아야 합니다. 서비스에 대한 포트 번호 또는 URI를 변경한 경우, 클라이언트가 변경되어야 합니다.

관련 정보:

파이프라인 구성 파일

웹 서비스 요청을 처리하기 위해 사용되는 파이프라인의 구성은 *파이프라인 구성 파일*로 알려진 XML 문서에 지정됩니다.

파이프라인 구성 파일이 z/OS UNIX 시스템 서비스 파일 시스템에 저장되고 해당 이름이 PIPELINE 자원 정의의 CONFIGFILE 속성에 지정됩니다. 적합한 XML 편집기 또는 텍스트 편집기를 사용하여 파이프라인 구성 파일에 대해 작업하십시오. 파이프라인 구성 파일에 대한 XML 스키마는 /usr/lpp/cicsts/cicsts53/schemas/pipeline/디렉토리에 있습니다. 여기서, /usr/lpp/cicsts/cicsts53은 z/OS UNIX에 있는 CICS 파일에 대한 기본 설치 디렉토리입니다. 구성 파일에 대해 작업할 때, 문자 세트 인코딩이 UTF-8인지 확인하십시오. EBCDIC에서 인코딩되는 기존 구성 파일을 가져오는 경우, 이는 자동으로 UTF-8로 변환됩니다.

CICS가 웹 서비스 요청을 처리할 때, 이는 요청을 처리하기 위해 하나 이상의 메시지 핸들러의 파이프라인을 사용합니다. 파이프라인은 웹 서비스 보안 및 웹 서비스 트랜잭션에 대한 지원과 같은 애플리케이션의 다른 카테고리에 적용되는 실행 환경의 측면을 제공하기 위해 구성됩니다. 일반적으로, 수많은 서비스 제공자 또는 서비스 요청자 애플

리케이션을 가지고 있는 CICS 리전에는 여러 다른 파이프라인 구성이 필요합니다. 그러나, 다른 애플리케이션에 유사한 요구사항이 있는 곳에서 동일한 파이프라인 구성을 공유할 수 있습니다.

참고: CICS 탐색기를 사용하여 번들의 일부로 새 PIPELINE 구성 파일을 작성할 때, 번들의 루트에서 동일한 이름을 가진 구성 파일이 되지 않아야 합니다.

두 가지 종류의 파이프라인 구성이 있습니다. 하나는 서비스 제공자 파이프라인의 구성에 대해 설명하고 하나는 서비스 요청자 파이프라인에 대해 설명합니다. 각각은 자체 스키마에 의해 정의되고 각각에는 다른 루트 요소가 있습니다.

파이프라인	스키마	루트 요소
서비스 제공자	Provider.xsd	<provider_pipeline>
서비스 요청자	Requester.xsd	<requester_pipeline>

사용되는 다수의 XML 요소가 파이프라인 구성의 두 가지 유형 모두에 공통일지라도 다른 것은 하나 또는 그 이외의 것에서만 사용됩니다. 따라서 제공자와 요청자 모두에 대해 동일한 구성 파일을 사용할 수 없습니다.

제한사항: 네임스페이스 규정 요소 이름은 파이프라인 구성 파일에서 지원되지 않습니다.

<provider_pipeline> 및 <requester_pipeline> 요소에는 다음 즉각적인 하위 요소가 있습니다.

- 모든 요청에 대해 호출되는 메시지 핸들러를 지정하는 <service> 요소. 이 요소는 <provider_pipeline> 요소에서 사용될 때 필수적이고 <requester_pipeline> 요소에서는 선택적입니다..
- 메시지 전송에 사용되는 자원을 기반으로 런타임 시 선택되는 메시지 핸들러를 지정하는 선택적 <transport> 요소.
- <provider_pipeline>의 경우에만 채널 첨부 애플리케이션 핸들러를 지정하기 위해 사용되는 선택적 <apphandler> 요소.
- <provider_pipeline>의 경우에만 Axis2 애플리케이션 핸들러를 지정하기 위해 사용되는 선택적 <apphandler_class> 요소.
- 파이프라인의 메시지 핸들러에 사용할 수 있는 매개변수를 포함하는 선택적 <service_parameter_list> 요소.

특정 요소에는 이와 연관되는 속성이 있을 수 있습니다. 각 속성 값은 유효한 XML 문서를 생성하기 위해 따옴표로 묶여 있어야 합니다.

PIPELINE 자원은 파이프라인 구성 파일과 연관됩니다. 속성에는 CONFIGFILE이 포함되고, 이는 z/OS UNIX에서 파이프라인 구성 파일의 이름을 지정합니다. PIPELINE 정의를 설치할 때, CICS는 파일에서 파이프라인을 구성하기 위해 요청하는 정보를 읽습니다.

CICS는 자체 구성 파일을 개발하기 위한 기본으로 사용할 수 있는 샘플 구성 파일을 제공합니다. 이는 /usr/lpp/cicsts/cicsts53/samples/pipelines 라이브러리에서 제공됩니다.

basicsoap11provider.xml

Java를 지원하지 않는 파이프라인에 대한 SOAP 1.1 프로토콜을 사용하는 서비스 제공자 파이프라인 정의. 파이프라인은 <cics_soap_1.1_handler> 메시지 핸들러를 사용하고 CICS 애플리케이션이 CICS 웹 서비스 보조를 사용하여 배치될 때 사용됩니다.

basicsoap11requester.xml

Java를 지원하지 않는 파이프라인에 대한 SOAP 1.1 프로토콜을 사용하는 서비스 요청자 파이프라인 정의. 파이프라인은 <cics_soap_1.1_handler> 메시지 핸들러를 사용하고 CICS 애플리케이션이 CICS 웹 서비스 보조를 사용하여 배치될 때 사용됩니다.

basicsoap11javaprovider.xml

Java를 지원하는 파이프라인에 대한 SOAP 1.1 프로토콜을 사용하는 서비스 제공자 파이프라인 정의. 파이프라인은 <cics_soap_1.1_handler_java> 메시지 핸들러를 사용하고 애플리케이션이 CICS 웹 서비스 보조를 사용하여 배치될 때 사용됩니다. 이 구성은 <jvmserver> 요소를 포함합니다. 이 메시지 핸들러는 구성이 사용될 수 있기 전에 적절한 JVM 서버를 지정하도록 편집되어야 합니다.

basicsoap11javarequester.xml

Java를 지원하는 파이프라인에 대한 SOAP 1.1 프로토콜을 사용하는 서비스 요청자 파이프라인 정의. 파이프라인은 <cics_soap_1.1_handler_java> 메시지 핸들러를 사용하고 애플리케이션이 CICS 웹 서비스 보조를 사용하여 배치될 때 사용됩니다. 이 구성은 <jvmserver> 요소를 포함합니다. 이 메시지 핸들러는 구성이 사용될 수 있기 전에 적절한 JVM 서버를 지정하도록 편집되어야 합니다.

jsonjavaprovider.xml

Java를 지원하는 파이프라인에 대한 JSON 메시지 형식을 사용하는 서비스 제공자 파이프라인 정의. 파이프라인은 <cics_json_handler_java> 메시지 핸들러를 사용하고 CICS 애플리케이션이 CICS JSON 보조를 사용하여 배치될 때 사용됩니다. 이 구성은 <jvmserver> 요소를 포함합니다. 이 메시지 핸들러는 구성이 사용될 수 있기 전에 적절한 JVM 서버를 지정하도록 편집되어야 합니다.

jsonzosconnectprovider.xml

CICS에 대한 z/OS 연결에 구성되는 PIPELINE에 배치되는 JSON 웹 서비스에 대한 파이프라인 정의. 파이프라인은 <provider_pipeline_json> 메시지

핸들러를 사용합니다. 이 구성은 <jvmserver> 요소를 포함합니다. 이 메시지 핸들러는 구성이 사용될 수 있기 전에 적절한 JVM 서버를 지정하도록 편집되어야 합니다.

kerberosprovider.xml

basicsoap11provider.xml.

samlprovider.xml

basicsoap11provider.xml에 SAML 지원에 대한 구성 정보를 추가하는 서비스 제공자 파이프라인 정의.

samlrequester.xml

basicsoap11requester.xml에 SAML 지원에 대한 구성 정보를 추가하는 서비스 요청자 파이프라인 정의.

propagatesamlprovider.xml

CICS 트랜잭션을 통해 basicsoap11provider.xml에 SAML 정보의 전파에 SAML 지원에 대한 구성 정보를 추가하는 서비스 제공자 파이프라인 정의.

propagatesamlrequester.xml

CICS 트랜잭션을 통해 basicsoap11requester.xml에 SAML 정보의 전파를 가진 SAML 지원에 대한 구성 정보를 추가하는 서비스 요청자 파이프라인 정의.

wsatprovider.xml

basicsoap11provider.xml에 웹 서비스 트랜잭션에 대한 구성 정보를 추가하는 파이프라인 정의.

wsatrequester.xml

basicsoap11requester.xml에 웹 서비스 트랜잭션에 대한 구성 정보를 추가하는 파이프라인 정의.

예제 제공자 파이프라인 구성 파일(JSON 애플리케이션 핸들러)

이는 <cics_json_handler_java> 요소를 사용하는 서비스 제공자 파이프라인에 대한 구성 파일의 단순한 예제입니다.

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <terminal_handler>
      <cics_json_handler_java>
        <jvmserver>DFHAXIS</jvmserver>
        <repository>/usr/lpp/cicsts/cicsts52/lib/pipeline/repository</repository>
      </cics_json_handler_java>
    </terminal_handler>
  </service>
  <apphandler_class>com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler</apphandler_class>
</provider_pipeline>
```


이 파이프라인은 하나의 메시지 핸들러만을 포함합니다. 핸들러는 DFHJSON 프로그램에 링크됩니다.

- <provider_pipeline> 요소는 서비스 제공자 파이프라인에 대한 파이프라인 구성 파일의 루트 요소입니다.
- <service> 요소는 모든 요청에 대해 호출되는 메시지 핸들러를 지정합니다. 예제에 하나의 메시지 핸들러만 있습니다.
- <terminal_handler> 요소에는 파이프라인의 터미널 메시지 핸들러의 정의가 포함됩니다.
- <cics_json_handler_java> 요소는 파이프라인이 Java 기반 파이프라인이고 파이프라인의 서비스 핸들러가 JSON 메시지를 지원하는 메시지 핸들러라는 것을 표시합니다.
- <apphandler> 요소는 파이프라인의 터미널 핸들러가 기본적으로 링크되는 애플리케이션 핸들러의 이름을 지정합니다. 이 경우, 프로그램은 DFHJSON이고, 이는 CICS JSON 보조로 배치되는 애플리케이션에 대한 CICS 제공 프로그램입니다.

예제 제공자 파이프라인 구성 파일(채널 첨부 애플리케이션 핸들러)

이는 <cics_json_handler_java> 요소를 사용하는 서비스 제공자 파이프라인에 대한 구성 파일의 단순한 예제입니다.

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

이 파이프라인은 하나의 메시지 핸들러만을 포함합니다. 핸들러는 DFHPITP 프로그램에 링크됩니다.

- <provider_pipeline> 요소는 서비스 제공자 파이프라인에 대한 파이프라인 구성 파일의 루트 요소입니다.
- <service> 요소는 모든 요청에 대해 호출되는 메시지 핸들러를 지정합니다. 예제에 하나의 메시지 핸들러만 있습니다.
- <terminal_handler> 요소에는 파이프라인의 터미널 메시지 핸들러의 정의가 포함됩니다.
- <cics_soap_1.1_handler> 요소는 파이프라인이 Java 기반 파이프라인이 아니고 파이프라인의 터미널 핸들러가 SOAP 1.1 메시지를 지원하는 메시지 핸들러라는 것을 표시합니다.

- <apphandler> 요소는 파이프라인의 터미널 핸들러가 기본적으로 링크되는 애플리케이션 핸들러의 이름을 지정합니다. 이 경우, 프로그램은 DFHPITP이고, 이는 CICS 웹 서비스 보조로 배치되는 애플리케이션에 대한 CICS 제공 프로그램입니다.

예제 제공자 파이프라인 구성 파일(Axis2 애플리케이션 핸들러)

이는 <cics_soap_1.1_handler_java> 요소를 사용하는 서비스 제공자 파이프라인에 대한 구성 파일의 단순한 예제입니다.

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

이 파이프라인은 하나의 메시지 핸들러만을 포함합니다. 핸들러는 DFHPITP 프로그램에 링크됩니다.

- <provider_pipeline> 요소는 서비스 제공자 파이프라인에 대한 파이프라인 구성 파일의 루트 요소입니다.
- <service> 요소는 모든 요청에 대해 호출되는 메시지 핸들러를 지정합니다. 예제에 하나의 메시지 핸들러만 있습니다.
- <terminal_handler> 요소에는 파이프라인의 터미널 메시지 핸들러의 정의가 포함됩니다.
- <cics_soap_1.1_handler_java> 요소는 파이프라인이 Java 기반 파이프라인이고 파이프라인의 서비스 핸들러가 SOAP 1.1 메시지를 지원하는 메시지 핸들러라는 것을 표시합니다.
- <apphandler_class> 요소는 제공된 Axis2 애플리케이션 핸들러를 지정합니다.

예제 요청자 파이프라인 구성 파일

이는 Axis2 MTOM/XOP 지원으로 <cics_soap_1.2_handler_java> 요소를 사용하는 서비스 요청자 파이프라인에 대한 구성 파일의 단순한 예제입니다.

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<requester_pipeline
  xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <service_handler_list>
      <cics_soap_1.2_handler_java>
        <jvmserver>JVMSESV1</jvmserver>
        <mtom>
      </cics_soap_1.2_handler_java>
    </service_handler_list>
  </service>
</requester_pipeline>
```

이 파이프라인은 하나의 메시지 핸들러만을 포함합니다.

- <requester_pipeline> 요소는 서비스 요청자 파이프라인에 대한 파이프라인 구성 파일의 루트 요소입니다.
- <service> 요소는 모든 요청에 대해 호출되는 메시지 핸들러를 지정합니다. 예제에 하나의 메시지 핸들러만 있습니다.
- <service_handler_list>는 모든 요청에 대해 호출되는 메시지 핸들러의 목록을 지정합니다.
- <cics_soap_1.2_handler_java> 요소는 파이프라인이 Java를 지원하고 파이프라인의 서비스 핸들러가 SOAP 1.2 메시지를 지원하는 메시지 핸들러라는 것을 표시합니다.
- <jvmserver> 요소는 사용될 JVM 서버를 지정합니다.
- <mtom/> 요소는 아웃바운드 XOP 문서가 MTOM 메시지에 패키징되고 전송되는 것을 지정합니다. 기본적으로, 인바운드 MTOM 메시지가 허용되고 Java 기반 파이프라인을 위해 패키지를 풉니다.

CICS JSON 웹 서비스에 대한 z/OS 연결을 위한 예제 제공자 파이프라인 구성 파일

이는 <provider_pipeline_json> 요소를 사용하는 서비스 제공자 파이프라인에 대한 구성 파일의 단순한 예제입니다. java_parser="NO" 속성이 제공되므로 비Java JSON 구문 분석기를 사용합니다.

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

<provider_pipeline_json> 요소는 핸들러 프로그램이 정의될 수 없다는 점에서 <provider_pipeline> 요소와 다릅니다.

- <provider_pipeline_json> 요소는 CICS JSON 웹 서비스 제공자 파이프라인에 대한 z/OS 연결을 위한 파이프라인 구성 파일의 루트 요소입니다.
- java_parser="NO" 속성은 비Java JSON 구문 분석기가 사용된다는 것을 지정합니다.
- <jvmserver> 요소는 사용될 JVM 서버를 지정합니다.

참고: CICS에 대한 z/OS 연결 이외의 것을 사용하여 <provider_pipeline_json> 파이프라인을 시작하려는 시도는 결과적으로 오류가 됩니다.

전송 관련 핸들러

각 파이프라인에 대한 구성 파일에서 두 세트 이상의 메시지 핸들러를 지정할 수 있습니다. 런타임 시, CICS는 메시지 전송에 사용되는 자원을 기반으로 호출되는 메시지 핸들러를 선택합니다.

서비스 제공자 및 서비스 요청자에서는 특정 전송(HTTP 또는 WebSphere MQ)이 사용 중일 때에만 일부 메시지 핸들러가 호출되어야 한다는 것을 지정할 수 있습니다. 예를 들어, 사용자의 직원에게 사용할 수 있는 웹 서비스를 고려하십시오. 회사 위치에서 일하는 직원은 보안 내부 네트워크에서 WebSphere MQ 전송을 사용하여 서비스에 액세스합니다. 그러나, 비즈니스 파트너 위치에서 일하는 직원은 인터넷을 통해 HTTP 전송을 사용하여 서비스에 액세스합니다. 이 상황에서, 정보의 민감한 특성으로 인해 HTTP 전송이 사용될 때 메시지의 일부를 암호화하기 위해 메시지 핸들러를 사용하려고 할 수 있습니다.

서비스 제공자에서 인바운드 메시지는 이름 지정된 자원(HTTP 전송의 경우 TCPIPService, MQ 전송의 경우 QUEUE)과 연관됩니다. 특정 자원이 인바운드 요청에 사용될 때에만 일부 메시지 핸들러가 호출되어야 한다는 것을 지정할 수 있습니다.

이를 가능하게 하려면 메시지 핸들러가 파이프라인 구성 파일의 두 개의 다른 부분에 지정됩니다.

서비스 섹션

파이프라인이 실행될 때마다 호출되는 메시지 핸들러를 지정합니다.

전송 섹션

사용 중인 전송 자원에 따라 호출되거나 호출되지 않는 메시지 핸들러를 지정합니다.

알아두기: 런타임 시, 메시지 핸들러는 파이프라인의 실행을 줄이도록 선택할 수 있습니다. 따라서, 특별한 메시지 핸들러가 파이프라인 구성 파일에 있는 것을 기반으로 호출되어야 한다고 CICS가 결정해도 의사결정은 초기 메시지 핸들러에 의해 무효화될 수 있습니다.

전송 섹션(전송 관련 핸들러) 내에 지정되는 메시지 핸들러는 여러 목록에서 관리됩니다. 런타임 시, CICS는 전송 자원이 사용 중인 것을 기반으로 실행을 위한 해당 목록 중 하나에서만 핸들러를 선택합니다. 둘 이상의 목록이 사용되는 전송 자원과 일치하는 경우, CICS는 가장 선택적인 목록을 사용합니다. 서비스 제공자와 서비스 요청자 파이프라인 모두에서 사용되는 목록은 다음과 같습니다.

<default_transport_handler_list>

이는 전송 관련 핸들러의 가장 작은 선택적 목록입니다. 이 목록에 지정된 핸들러는 사용되는 전송 자원이 다음 목록 중 어떤 것과도 일치하지 않을 때 호출됩니다.

<default_http_transport_handler_list>

서비스 요청자 파이프라인에서 이 목록의 핸들러는 HTTP 전송이 사용 중일 때 호출됩니다.

서비스 제공자 파이프라인에서 이 목록의 핸들러는 HTTP 전송이 사용 중일 때 호출되고 <named_transport_entry>는 TCP/IP 연결에 대한 TCIPSERVICE의 이름을 지정하지 않습니다.

<default_mq_transport_handler_list>

서비스 요청자 파이프라인에서 이 목록의 핸들러는 WebSphere MQ 전송이 사용 중일 때 호출됩니다.

서비스 제공자 파이프라인에서 이 목록의 핸들러는 WebSphere MQ 전송이 사용 중일 때 호출되고 <named_transport_entry>는 인바운드 메시지가 수신되는 메시지 큐의 이름을 지정하지 않습니다.

메시지 핸들러의 다음 목록은 서비스 제공자 파이프라인에 대한 구성 파일에서만 사용됩니다.

<named_transport_entry>

핸들러의 목록 뿐만 아니라, <named_transport_entry>는 자원의 이름 및 전송 유형을 지정합니다.

- HTTP 전송의 경우, 이 목록의 핸들러는 자원 이름이 인바운드 TCP/IP 연결에 대한 TCIPSERVICE의 이름과 일치할 때 호출됩니다.
- WebSphere MQ 전송의 경우, 이 목록의 핸들러는 자원 이름이 인바운드 메시지를 수신하는 메시지 큐의 이름과 일치할 때 호출됩니다.

예제

이는 서비스 제공자 파이프라인에 대한 파이프라인 구성 파일의 <transport> 요소 예제입니다.

```
<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>
```

이 정의의 효과는 이것입니다.

- <default_mq_transport_handler_list>는 MQ 전송을 사용하는 메시지가 핸들러 HANDLER3에 의해 처리되는지 확인합니다.
- <named_transport_entry>는 TCPIP SERVICE(WS00)와 연관된 TCP/IP 연결을 사용하는 메시지가 핸들러 HANDLER4에 의해 처리되는지 확인합니다.
- <default_transport_handler_list>는 모든 잔존 메시지(즉, HTTP 전송을 사용하지만, TCPIP SERVICE(WS00)를 사용하지 않음)가 핸들러 HANDLER1 및 HANDLER2에 의해 처리됩니다.

알아두기: 전송 섹션에 지정된 것 뿐만 아니라 파이프라인 정의의 서비스 섹션에 지정되는 핸들러가 호출됩니다.

서비스 제공자에 대한 파이프라인 정의

메시지 핸들러는 XML 문서에서 정의되며 z/OS UNIX에서 저장됩니다. 문서를 포함하는 파일의 이름은 파이프라인 정의의 CFGFILE 속성으로 지정됩니다.

파이프라인 구성 문서의 루트 요소는 <provider_pipeline> 요소입니다. 문서의 상위 레벨 구조는 279 페이지의 그림 25에 표시되어 있습니다.

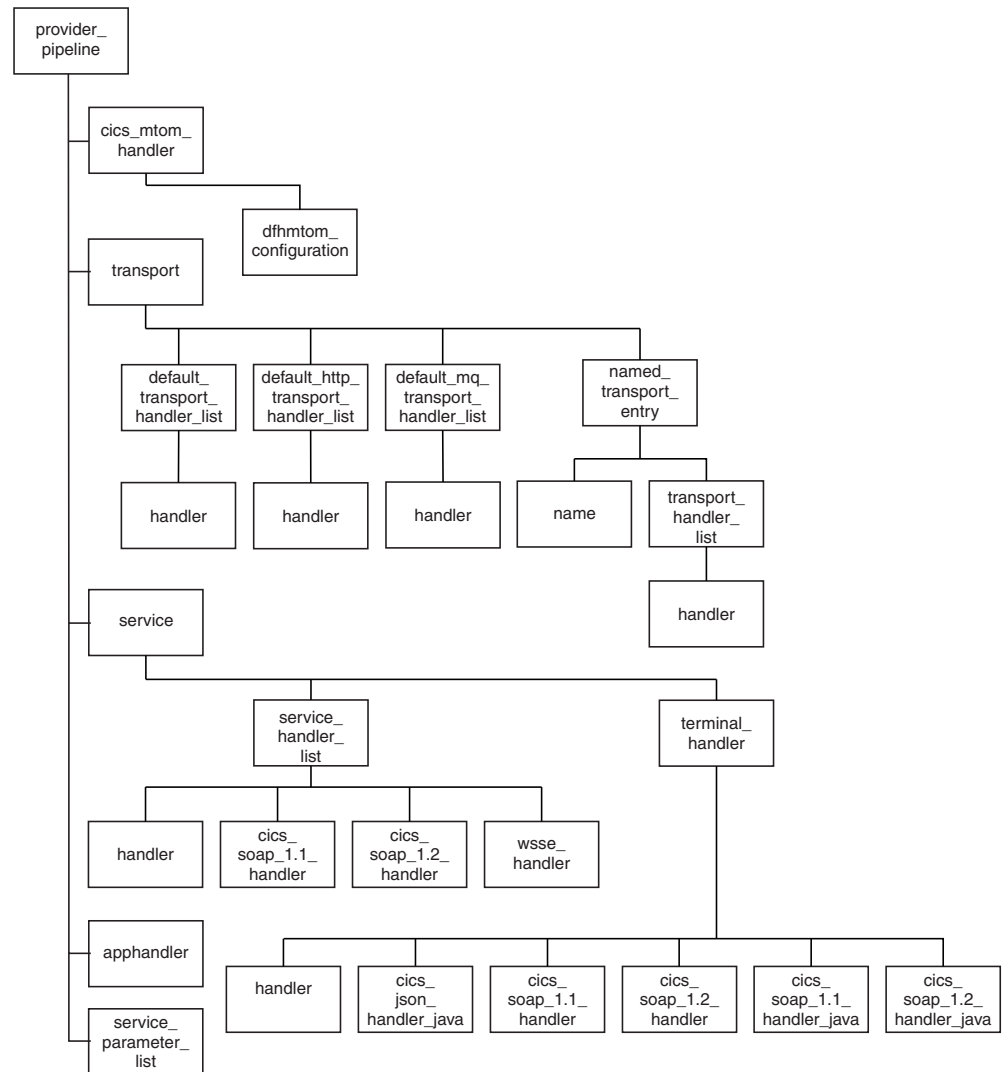


그림 25. 서비스 제공자에 대한 파이프라인 정의의 구조

참고: 그림을 단순화하기 위해 <handler>, <cics_json_handler_java>, <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 및 <cics_soap_1.2_handler_java> 요소의 하위 요소는 표시되지 않습니다.

서비스 요청자에 대한 파이프라인 정의

메시지 핸들러는 XML 문서에서 정의되며 z/OS UNIX에서 저장됩니다. 문서를 포함하는 파일의 이름은 파이프라인 정의의 CFGFILE 속성으로 지정됩니다.

파이프라인 구성 문서의 루트 요소는 <requester_pipeline> 요소입니다. 문서의 상위 레벨 구조는 그림 26에 표시되어 있습니다.

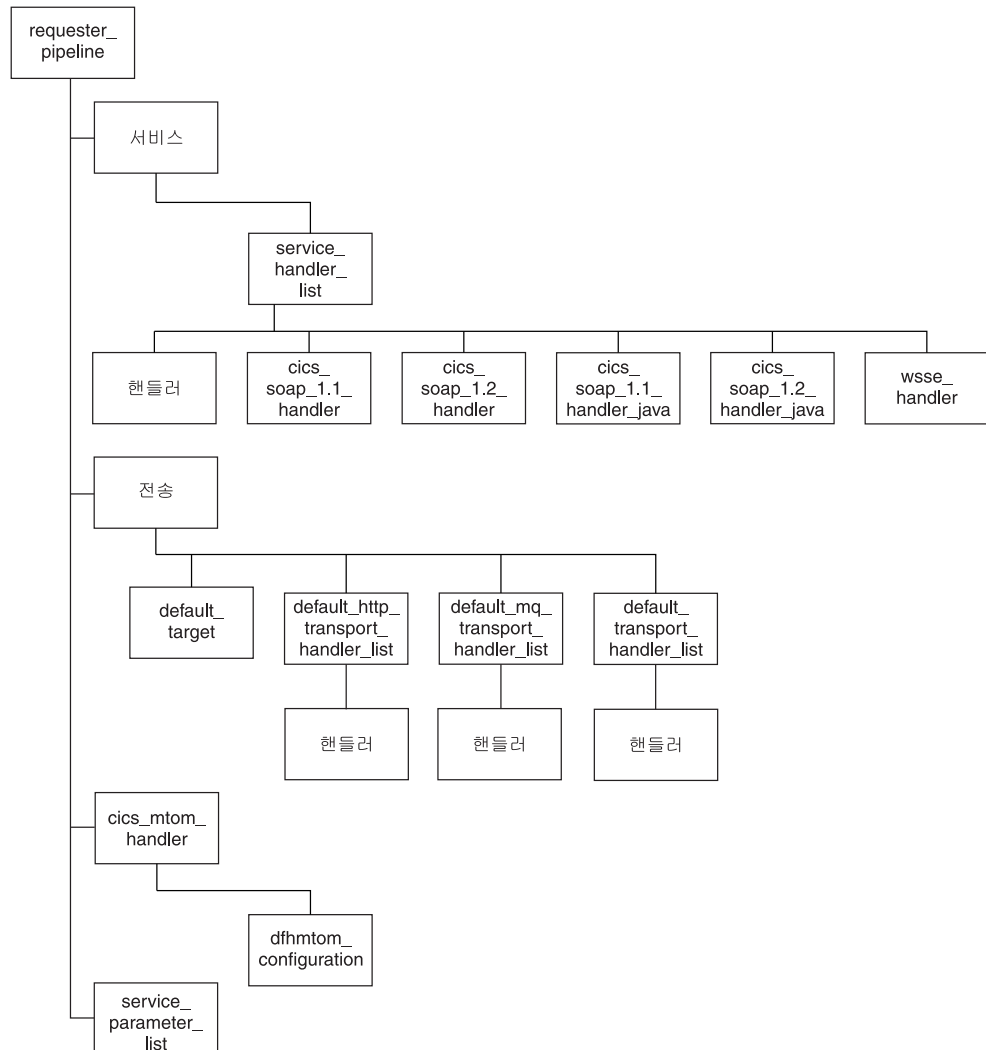


그림 26. 서비스 요청자에 대한 파이프라인 정의의 구조

참고: 그림을 단순화하기 위해 <handler>, <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 및 <cics_soap_1.2_handler_java> 요소의 하위 요소는 표시되지 않습니다.

서비스 제공자에서만 사용되는 요소

파이프라인 구성 파일에서 사용되는 일부 XML 요소는 서비스 제공자 파이프라인에만 적용됩니다.

<apphandler> 요소

파이프라인의 터미널 핸들러가 기본적으로 링크되는 애플리케이션 핸들러의 이름을 지정합니다.

<apphandler> 요소는 터미널 핸들러가 제공된 SOAP 메시지 핸들러 중 하나인 경우에 사용됩니다. 이 상황은 <terminal_handler> 요소에 <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소가 포함되는 경우에 발생합니다. 그러나 <terminal_handler> 요소에 <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소가 포함되는 경우 <apphandler> 요소 대신 <apphandler_class> 요소를 지정하여 제공된 Axis2 애플리케이션 핸들러를 사용할 수 있습니다. 자세한 정보는 <apphandler_class> 요소를 참조하십시오. 그러나 <apphandler_class> 및 <apphandler> 요소는 같은 파이프라인 구성 파일에서 지정하지 않습니다.

CICS 웹 서비스 지원을 사용하여 웹 서비스 애플리케이션을 배치하는 경우 <apphandler> 요소에 다음 애플리케이션 핸들러 중 하나를 지정해야 합니다.

- Java를 사용하여 애플리케이션 핸들러를 처리하지 않으려는 경우 제공된 애플리케이션 핸들러 DFHPITP
- DFHPITP를 사용하는 자체 애플리케이션 핸들러
- 작성하는 PROGRAM 자원의 이름.

애플리케이션 핸들러에 대한 자세한 정보는 327 페이지의 『애플리케이션 핸들러』의 내용을 참조하십시오.

사용 위치:

- 서비스 제공자

다음에 포함됨:

- <provider_pipeline> 요소

예제

```
<apphandler>DFHPITP</apphandler>
```

<apphandler_class> 요소

파이프라인의 터미널 핸들러가 Axis2 애플리케이션 핸들러에 링크되도록 지정합니다.

<apphandler_class> 요소는 <terminal_handler> 요소에 <cics_json_handler_java>, <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소가 포함될 때 Axis2 애플리케이션 핸들러를 지정하는 데 사용됩니다. 제공된 Axis2 애플리케이션 핸들러를 사용하려면 <apphandler_class> 요소에서

com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler를 지정하십시오. CICS SOAP 핸들러를 사용할 때 자체 Axis2 애플리케이션 핸들러 클래스를 지정할 수도 있습니다.

또는 채널 접속 애플리케이션 핸들러를 사용하려는 경우 파이프라인 구성 파일에서 <apphandler> 요소를 지정할 수 있습니다. 자세한 정보는 <apphandler> 요소를 참조하십시오. 그러나 <apphandler_class> 및 <apphandler> 요소는 같은 파이프라인 구성 파일에서 지정하지 않습니다.

참고: <apphandler> 요소는 <cics_json_handler_java> 요소와 함께 사용하지 않습니다.

<terminal_handler> 요소에 <cics_soap_1.1_handler> 또는 <cics_soap_1.2_handler> 요소가 포함되는 경우 <apphandler_class> 요소는 사용하지 않습니다.

애플리케이션 핸들러에 대한 자세한 정보는 327 페이지의 『애플리케이션 핸들러』의 내용을 참조하십시오.

사용 위치:

- 서비스 제공자

다음에 포함됨:

- <provider_pipeline> 요소

예제

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

<named_transport_entry> 요소

이름 지정된 전송 자원을 서비스 제공자가 사용 중인 경우에 호출되는 핸들러 목록을 포함합니다.

- WebSphere MQ 전송의 경우 이름 지정된 자원은 요청을 수신하는 로컬 입력 큐입니다.
- HTTP 전송의 경우 자원은 요청을 수신한 포트를 정의하는 TCPIPSERVICE입니다.

사용 위치:

- 서비스 제공자

다음에 포함됨:

<transport>

속성:

이름	설명
유형	이름 지정된 자원이 연관되는 전송 메커니즘입니다. wmq 이름 지정된 자원이 큐입니다. http 이름 지정된 자원이 TCIPSERVICE입니다.

포함 대상:

1. 자원의 이름을 포함하는 <name> 요소
2. 선택적 <transport_handler_list> 요소. 각 <transport_handler_list>에는 한 개 이상의 <handler> 요소가 포함됩니다.

<transport_handler_list> 요소를 코드화하지 않으면 이름 지정된 전송이 사용될 때 호출되는 메시지 핸들러만 <service> 요소에서 지정되는 메시지 핸들러입니다.

예제

```
<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>
```

이 예제에서 지정된 메시지 핸들러(HANDLER1 및 HANDLER2)가 이름이 PORT80인 TCIPSERVICE에서 수신된 메시지에 대해 호출됩니다.

<provider_pipeline> 요소

웹 서비스 제공자의 CICS 파이프라인 구성을 설명하는 XML 문서의 루트 요소를 지정합니다.

사용 위치:

- 서비스 제공자

포함 대상:

1. 선택적 <cics_mtom_handler> 요소
2. 선택적 <transport> 요소
3. <service> 요소
4. 선택적 <apphandler> 요소
5. 선택적 <apphandler_class> 요소

6. 컨테이너 DFH-HANDLERPLIST에서 파이프라인의 모든 메시지 핸들러에 사용할 수 있도록 설정된 XML 요소를 포함하는 선택적 <service_parameter_list> 요소

예제

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

<provider_pipeline_json> 요소

z/OS Connect JSON 웹 서비스 제공자의 CICS 파이프라인 구성을 설명하는 XML 문서의 루트 요소를 지정합니다.

이는 핸들러 프로그램을 정의할 수 없다는 점에서 <provider_pipeline> 요소와 다릅니다. 이 스타일의 파이프라인은 z/OS Connect에서 사용되는 **WEBSERVICE** 자원의 컨테이너로 사용됩니다. z/OS Connect 외의 프로그램을 사용하여 <provider_pipeline_json> 파이프라인을 시작하려고 하면 오류가 발생합니다. 결과로 발생한 **PIPELINE** 자원은 **USAGE(PIPELINE) URIMAP** 자원의 대상으로 사용할 수 없습니다. 이 자원은 **USAGE(JVMSEVER) URIMAP** 자원과 함께만 사용할 수 있습니다.

사용 위치:

- 서비스 제공자

속성:

java_parser={yes|no}

인바운드 메시지를 처리하는 데 사용되는 JSON 구문 분석기의 유형을 선택합니다.

가능한 값은 다음과 같습니다.

yes JVM 서버 내에서 Java를 사용하여 JSON 구문 분석을 수행합니다. 이는 기본값입니다.

no JSON 메시지의 비Java 구문 분석을 수행합니다.

참고: java_parser 속성은 선택적입니다. 이 값을 제공하지 않으면 기본 동작은 JVM 서버 내에서 Java를 사용하여 JSON 메시지를 구문 분석하는 것입니다. 이는 java_parser="yes"를 지정하는 것과 같습니다.

포함 대상:

- z/OS Connect가 구성되는 JVMSEVER 자원의 이름을 포함하는 <jvmserver> 요소

Java 구문 분석을 사용하는 예제

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

비Java 구문 분석을 사용하는 예제

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

`<terminal_handler>` 요소

서비스 제공자 파이프라인의 터미널 메시지 핸들러에 대한 정의를 포함합니다.

사용 위치:

- 서비스 제공자

다음에 포함됨:

- `<service>` 요소

포함 대상:

다음 요소 중 하나를 포함합니다.

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

파이프라인이 SOAP 1.1 및 SOAP 1.2 메시지를 둘 다 처리할 것으로 예상하는 경우 `<cics_soap_1.2_handler>` 또는 `<cics_soap_1.2_handler_java>` 요소를 사용해야 합니다.

알아두기: 서비스 제공자에서는 `<terminal_handler>` 요소 외에도 `<service_handler_list>` 요소의 `<cics_soap_1.1_handler>` 및 `<cics_soap_1.2_handler>`를 지정할 수 있습니다. 그러나 서비스 제공자에서는 `<terminal_handler>` 요소의 `<cics_soap_1.1_handler_java>` 및 `<cics_soap_1.2_handler_java>`만 지정할 수 있습니다.

예제

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

<transport_handler_list> 요소

이름 지정된 자원이 사용될 때 호출되는 메시지 핸들러 목록을 포함합니다.

- MQ 전송의 경우 이름 지정된 자원은 로컬 입력 큐의 이름입니다.
- HTTP 전송의 경우 자원은 요청을 수신한 포트를 정의하는 TCPIPSERVICE입니다.

사용 위치:

- 서비스 제공자

다음에 포함됨:

- <named_transport_entry> 요소

포함 대상:

- 한 개 이상의 <handler> 요소

예제

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

서비스 요청자에서 사용되는 요소

파이프라인 구성 파일에서 사용되는 일부 XML 요소는 서비스 요청자 파이프라인에만 적용됩니다.

<requester_pipeline> 요소

서비스 요청자에서 파이프라인의 구성을 설명하는 XML 문서의 루트 요소입니다.

사용 위치:

- 서비스 요청자

포함 대상:

1. 선택적 <service> 요소
2. 선택적 <transport> 요소

3. 선택적 <cics_mtom_handler> 요소
4. 컨테이너 DFH-HANDLERPLIST에서 메시지 핸들러에 사용할 수 있도록 설정된 XML 요소를 포함하는 선택적 <service_parameter_list> 요소

예제

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

서비스 제공자 및 서비스 요청자 파이프라인에서 사용되는 요소

파이프라인 구성 파일에서 사용되는 일부 XML 요소는 서비스 제공자 및 서비스 요청자 파이프라인 둘 다에 적용됩니다.

<addressing> 요소

Java 기반 SOAP 처리에서 WS-Addressing에 대한 지원을 지정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

<cics_soap_1.1_handler_java> 요소

<cics_soap_1.2_handler_java> 요소

포함 대상:

<namespace> 요소. 서비스 제공자에서 이 요소는 선택사항입니다. 이 요소는 CICS에서 지원되는 두 개의 WS-Addressing 스키마 중 하나를 포함합니다. 인바운드 메시지의 경우 Axis2는 두 스펙을 모두 지원합니다. 아웃바운드 메시지의 경우 이 요소에서 지정된 네임스페이스가 사용됩니다. 이 요소를 지정하지 않거나 두 개의 요소가 있는 경우 CICS는 아웃바운드 메시지에 대해 인바운드 메시지와 같은 스펙을 사용합니다. 서비스 요청자에서 이 요소는 필수이며 아웃바운드 메시지에 대해 한 개의 네임스페이스만 지정할 수 있습니다.

이 예제는 두 WS-Addressing 스펙이 모두 지원되는 서비스 제공자 파이프라인에 대한 구성을 보여줍니다. CICS는 아웃바운드 메시지에 대해 인바운드 메시지와 같은 스펙을 사용합니다. 비어 있는 <addressing> 요소를 지정하여 같은 결과를 얻을 수 있습니다.


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

<cics_json_handler_java> 요소

Java 기반 JSON 파이프라인에서 JSON 메시지에 대한 핸들러 프로그램의 속성을 지정합니다.

사용 위치:

- 서비스 제공자

다음에 포함됨:

```
<service_handler_list> 요소
<terminal_handler> 요소
```

포함 대상:

1. <jvmserver> 요소
2. 선택적 <repository> 요소

예제

다음 예제는 Java 기반 JSON 핸들러에 대한 XML 및 해당하는 중첩된 요소를 보여줍니다.

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <terminal_handler>
      <cics_json_handler_java>
        <jvmserver>DFHAXIS</jvmserver>
        <repository>/usr/lpp/cicsts/cicsts53/lib/pipeline/repository</repository>
      </cics_json_handler_java>
    </terminal_handler>
  </service>
  <apphandler_class>com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler</apphandler_class>
</provider_pipeline>
```

<cics_soap_1.1_handler> 요소

비Java 파이프라인에서 SOAP 1.1 메시지에 대한 핸들러 프로그램의 속성을 지정합니다.

사용 위치:

- 서비스 요청자
- 서비스 제공자

다음에 포함됨:

```
<service_handler_list> 요소
<terminal_handler> 요소
```

포함 대상:

0, 1 또는 2개 이상의 <headerprogram> 요소. 각 <headerprogram>에는 다음이 포함됩니다.

1. 헤더 처리 프로그램의 이름을 포함하는 <program_name> 요소
2. 다음 <localname> 요소와 함께 사용되어 헤더 처리 프로그램이 처리해야 하는 SOAP 메시지의 헤더 블록을 판별하는 <namespace> 요소. <namespace> 요소에는 헤더 블록 네임스페이스의 URI(Uniform Resource Identifier)가 포함됩니다.
3. 앞의 <namespace> 요소와 함께 사용되어 헤더 처리 프로그램이 처리해야 하는 SOAP 메시지의 헤더 블록을 판별하는 <localname> 요소. <localname>에는 헤더 블록의 요소 이름이 포함됩니다.

예를 들어, 이 헤더 블록을 고려하십시오.

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

- 네임스페이스 이름은 http://mynamespace입니다.
- 요소 이름은 myheaderblock입니다.

헤더 프로그램을 이 헤더 블록과 일치시키려면 <namespace> 및 <localname> 요소를 다음과 같이 코드화하십시오.

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

<localname> 요소에서 별표(*)를 코드화하여 이름이 지정된 문자열로 시작되는 네임스페이스의 모든 헤더 블록이 처리되어야 함을 표시할 수 있습니다. 예를 들어, 다음과 같습니다.

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

<localname> 요소에 별표를 사용하면 메시지의 헤더가 둘 이상의 <headerprogram> 요소와 일치할 수 있습니다. 예를 들어, 이 헤더 블록

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

은 다음과 같은 모든 <headerprogram> 요소와 일치합니다.

```
<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>

```

이 경우 실행되는 헤더 프로그램은 헤더 블록의 요소 이름이 가장 정확하게 명시된 `<headerprogram>` 요소에서 지정되는 헤더 프로그램입니다. 예제에서 이러한 헤더 프로그램은 HDRPROG3입니다.

SOAP 메시지에 둘 이상의 헤더가 있는 경우 헤더 처리 프로그램은 헤더 일치가 발생할 때마다 한 번씩 호출되지만 헤더가 처리되는 순서는 정의되지 않습니다.

같은 `<namespace>` 및 `<localname>`을 포함하지만 서로 다른 헤더 프로그램을 지정하는 둘 이상의 `<headerprogram>` 요소를 코드화하는 경우 헤더 프로그램 중 하나만 실행되지만 그 중 실행될 프로그램은 정의되지 않습니다.

4. XML 부울 값(true 또는 false)을 포함하는 `<mandatory>` 요소. 또는 값을 각각 1 또는 0으로 코드화할 수 있습니다.

true

서비스 제공자 파이프라인에서 서비스 요청을 처리하는 동안 및 서비스 요청자 파이프라인에서 서비스 응답을 처리하는 동안 SOAP 메시지의 헤더 중 `<namespace>` 및 `<localname>` 요소와 일치하는 헤더가 없는 경우에도 헤더 처리 프로그램이 최소 한 번 이상 호출됩니다.

- 일치하는 헤더가 없는 경우에도 헤더 처리 프로그램은 한 번 호출됩니다.
- 일치하는 헤더가 있는 경우에 헤더 처리 프로그램은 헤더가 일치할 때마다 한 번씩 호출됩니다.

서비스 요청자 파이프라인에서 서비스 요청을 처리하는 동안 및 서비스 제공자 파이프라인에서 서비스 응답을 처리하는 동안 CICS가 작성하는 SOAP 메시지에 처음부터 헤더가 없는 경우라도 헤더 처리 프로그램은 최소 한 번 이상 호출됩니다. 메시지에 헤더를 추가하려면 `<mandatory>true</mandatory>` 또는 `<mandatory>1</mandatory>`를 지정하여 최소 한 개 이상의 헤더 처리 프로그램이 호출되도록 해야 합니다.

false

SOAP 메시지의 헤더가 한 개 이상 `<namespace>` 및 `<localname>` 요소와 일치하는 경우에만 헤더 처리 프로그램이 호출됩니다.

- 일치하는 헤더가 없는 경우 헤더 처리 프로그램은 호출되지 않습니다.
- 일치하는 헤더가 있는 경우에 헤더 처리 프로그램은 헤더가 일치할 때마다 한 번씩 호출됩니다.

예제

```
<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_1.1_handler_java> 요소

Java 기반 파이프라인에서 SOAP 1.1 메시지에 대한 핸들러 프로그램의 속성을 지정합니다.

사용 위치:

- 서비스 요청자
- 서비스 제공자

다음에 포함됨:

```
<service_handler_list> 요소
<terminal_handler> 요소
```

포함 대상:

1. <jvmserver> 요소
2. 선택적 <repository> 요소
3. 선택적 <addressing> 요소. Axis2 에서 WS-Addressing을 사용으로 설정하는 경우 DFHWSADH 헤더 처리 프로그램을 사용하지 마십시오.
4. 0, 1 또는 2개 이상의 <headerprogram> 요소. 각 <headerprogram> 요소에는 다음이 포함됩니다.
 - a. 헤더 처리 프로그램의 이름을 포함하는 <program_name> 요소. Java에서 Axis2 핸들러를 작성하여 SOAP 헤더를 처리할 수 있습니다.
 - b. 다음 <localname> 요소와 함께 사용되어 헤더 처리 프로그램이 처리해야 하는 SOAP 메시지의 헤더 블록을 판별하는 <namespace> 요소. <namespace> 요소에는 헤더 블록 네임스페이스의 URI(Uniform Resource Identifier)가 포함됩니다.
 - c. 앞의 <namespace> 요소와 함께 사용되어 헤더 처리 프로그램이 처리해야 하는 SOAP 메시지의 헤더 블록을 판별하는 <localname> 요소. <localname>에는 헤더 블록의 요소 이름이 포함됩니다.

예를 들어, 이 헤더 블록을 고려하십시오.

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

네임스페이스 이름은 http://mynamespace이고 요소 이름은 myheaderblock입니다.

헤더 프로그램을 이 헤더 블록과 일치시키려면 <namespace> 및 <localname> 요소를 다음과 같이 코드화하십시오.

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

<localname> 요소에서 별표(*)를 코드화하여 이름이 지정된 문자열로 시작되는 네임스페이스의 모든 헤더 블록이 처리되어야 함을 표시할 수 있습니다. 예를 들어, 다음과 같습니다.

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

<localname> 요소에 별표를 사용하면 메시지의 헤더가 둘 이상의 <headerprogram> 요소와 일치할 수 있습니다. 예를 들어, 이 헤더 블록

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

은 다음과 같은 모든 <headerprogram> 요소와 일치합니다.

```
<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>
```

이 경우 실행되는 헤더 프로그램은 헤더 블록의 요소 이름이 가장 정확하게 명시된 <headerprogram> 요소에서 지정되는 헤더 프로그램입니다. 예제에서 이러한 헤더 프로그램은 HDRPROG3입니다.

SOAP 메시지에 둘 이상의 헤더가 있는 경우 헤더 처리 프로그램은 헤더 일치가 발생할 때마다 한 번씩 호출되지만 헤더가 처리되는 순서는 정의되지 않습니다.

같은 <namespace> 및 <localname> 요소를 포함하지만 서로 다른 헤더 프로그램을 지정하는 둘 이상의 <headerprogram> 요소를 코드화하는 경우 헤더 프로그램 중 하나만 실행되지만 그 중 실행될 프로그램은 정의되지 않습니다.

- d. XML 부울 값(true 또는 false)을 포함하는 <mandatory> 요소. 또는 값을 각각 1 또는 0으로 코드화할 수 있습니다.

true

서비스 제공자 파이프라인에서 서비스 요청을 처리하는 동안 및 서비스 요청자 파이프라인에서 서비스 응답을 처리하는 동안 SOAP 메시지의 헤더 중 <namespace> 및 <localname> 요소와 일치하는 헤더가 없는 경우에도 헤더 처리 프로그램이 최소 한 번 이상 호출됩니다.

- 일치하는 헤더가 없는 경우에도 헤더 처리 프로그램은 한 번 호출됩니다.
- 일치하는 헤더가 있는 경우에 헤더 처리 프로그램은 헤더가 일치할 때마다 한 번씩 호출됩니다.

서비스 요청자 파이프라인에서 서비스 요청을 처리하는 동안 및 서비스 제공자 파이프라인에서 서비스 응답을 처리하는 동안 CICS가 작성하는 SOAP 메시지에 처음부터 헤더가 없는 경우라도 헤더 처리 프로그램은 최소 한 번 이상 호출됩니다. 메시지에 헤더를 추가하려면 <mandatory>true</mandatory> 또는 <mandatory>1</mandatory>를 지정하여 최소 한 개 이상의 헤더 처리 프로그램이 호출되도록 해야 합니다.

false

SOAP 메시지의 헤더가 한 개 이상 <namespace> 및 <localname> 요소와 일치하는 경우에만 헤더 처리 프로그램이 호출됩니다.

- 일치하는 헤더가 없는 경우 헤더 처리 프로그램은 호출되지 않습니다.
- 일치하는 헤더가 있는 경우에 헤더 처리 프로그램은 헤더가 일치할 때마다 한 번씩 호출됩니다.

예제

다음 예제는 Java 기반 SOAP 핸들러에 대한 XML 및 해당하는 중첩된 요소를 보여줍니다.

```
<cics_soap_1.1_handler_java>
  <jvmserver>JVMSERV1</jvmserver>
  <headerprogram>
    <program_name>HDRPROG4</program_name>
    <namespace>http://mynamespace</namespace>
    <localname>myheaderblock</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.1_handler_java>
```

<cics_soap_1.2_handler> 요소

비Java 파이프라인에서 SOAP 1.2 메시지에 대한 핸들러 프로그램의 속성을 지정합니다.

사용 위치:

- 서비스 요청자
- 서비스 제공자

다음에 포함됨:

<service_handler_list> 요소

<terminal_handler> 요소

포함 대상:

0, 1 또는 2개 이상의 <headerprogram> 요소. 각 <headerprogram>에는 다음이 포함됩니다.

1. 헤더 처리 프로그램의 이름을 포함하는 <program_name> 요소
2. 다음 <localname> 요소와 함께 사용되어 헤더 처리 프로그램이 처리해야 하는 SOAP 메시지의 헤더 블록을 판별하는 <namespace> 요소. <namespace> 요소에는 헤더 블록 네임스페이스의 URI(Uniform Resource Identifier)가 포함됩니다.
3. 앞의 <namespace> 요소와 함께 사용되어 헤더 처리 프로그램이 처리해야 하는 SOAP 메시지의 헤더 블록을 판별하는 <localname> 요소. <localname>에는 헤더 블록의 요소 이름이 포함됩니다.

예를 들어, 이 헤더 블록을 고려하십시오.

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

- 네임스페이스 이름은 http://mynamespace입니다.
- 요소 이름은 myheaderblock입니다.

헤더 프로그램을 이 헤더 블록과 일치시키려면 <namespace> 및 <localname> 요소를 다음과 같이 코드화하십시오.

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

<localname> 요소에서 별표(*)를 코드화하여 이름이 지정된 문자열로 시작되는 네임스페이스의 모든 헤더 블록이 처리되어야 함을 표시할 수 있습니다. 예를 들어, 다음과 같습니다.

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

<localname> 요소에 별표를 사용하면 메시지의 헤더가 둘 이상의 <headerprogram> 요소와 일치할 수 있습니다. 예를 들어, 이 헤더 블록


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

은 다음과 같은 모든 <headerprogram> 요소와 일치합니다.

```
<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>
```

이 경우 실행되는 헤더 프로그램은 헤더 블록의 요소 이름이 가장 정확하게 명시된 <headerprogram> 요소에서 지정되는 헤더 프로그램입니다. 예제에서 이러한 헤더 프로그램은 HDRPROG3입니다.

SOAP 메시지에 둘 이상의 헤더가 있는 경우 헤더 처리 프로그램은 헤더 일치가 발생할 때마다 한 번씩 호출되지만 헤더가 처리되는 순서는 정의되지 않습니다.

같은 <namespace> 및 <localname>을 포함하지만 서로 다른 헤더 프로그램을 지정하는 둘 이상의 <headerprogram> 요소를 코드화하는 경우 헤더 프로그램 중 하나만 실행되지만 그 중 실행될 프로그램은 정의되지 않습니다.

4. XML 부울 값(true 또는 false)을 포함하는 <mandatory> 요소, 또는 값을 각각 1 또는 0으로 코드화할 수 있습니다.

true

서비스 제공자 파이프라인에서 서비스 요청을 처리하는 동안 및 서비스 요청자 파이프라인에서 서비스 응답을 처리하는 동안 SOAP 메시지의 헤더 중 <namespace> 및 <localname> 요소와 일치하는 헤더가 없는 경우에도 헤더 처리 프로그램이 최소 한 번 이상 호출됩니다.

- 일치하는 헤더가 없는 경우에도 헤더 처리 프로그램은 한 번 호출됩니다.
- 일치하는 헤더가 있는 경우에 헤더 처리 프로그램은 헤더가 일치할 때마다 한 번씩 호출됩니다.

서비스 요청자 파이프라인에서 서비스 요청을 처리하는 동안 및 서비스 제공자 파이프라인에서 서비스 응답을 처리하는 동안 CICS가 작성하는 SOAP 메시지에 처음부터 헤더가 없는 경우라도 헤더 처리 프로그램은 최소 한 번 이상 호

출됩니다. 메시지에 헤더를 추가하려면 `<mandatory>true</mandatory>` 또는 `<mandatory>1</mandatory>`를 지정하여 최소 한 개 이상의 헤더 처리 프로그램이 호출되도록 해야 합니다.

false

SOAP 메시지의 헤더가 한 개 이상 `<namespace>` 및 `<localname>` 요소와 일치하는 경우에만 헤더 처리 프로그램이 호출됩니다.

- 일치하는 헤더가 없는 경우 헤더 처리 프로그램은 호출되지 않습니다.
- 일치하는 헤더가 있는 경우에 헤더 처리 프로그램은 헤더가 일치할 때마다 한 번씩 호출됩니다.

예제

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

<cics_soap_1.2_handler_java> 요소

Java 기반 SOAP 파이프라인에서 SOAP 1.2 메시지에 대한 핸들러 프로그램의 속성을 지정합니다.

사용 위치:

- 서비스 요청자
- 서비스 제공자

다음에 포함됨:

```
<service_handler_list> 요소
<terminal_handler> 요소
```

포함 대상:

1. `<jvmserver>` 요소
2. 선택적 `<repository>` 요소
3. 선택적 `<addressing>` 요소. Axis2 에서 WS-Addressing에 대한 지원을 사용으로 설정하는 경우 헤더 처리 프로그램을 사용하지 마십시오. Java에서 Axis2 핸들러를 작성하여 SOAP 헤더를 처리할 수 있습니다.
4. 0, 1 또는 2개 이상의 `<headerprogram>` 요소. 각 `<headerprogram>` 요소에는 다음이 포함됩니다.
 - a. 헤더 처리 프로그램의 이름을 포함하는 `<program_name>` 요소

- b. 다음 <localname> 요소와 함께 사용되어 헤더 처리 프로그램이 처리해야 하는 SOAP 메시지의 헤더 블록을 판별하는 <namespace> 요소. <namespace> 요소에는 헤더 블록 네임스페이스의 URI(Uniform Resource Identifier)가 포함됩니다.
- c. 앞의 <namespace> 요소와 함께 사용되어 헤더 처리 프로그램이 처리해야 하는 SOAP 메시지의 헤더 블록을 판별하는 <localname> 요소. <localname>에는 헤더 블록의 요소 이름이 포함됩니다.

예를 들어, 이 헤더 블록을 고려하십시오.

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

네임스페이스 이름은 http://mynamespace이고 요소 이름은 myheaderblock입니다.

헤더 프로그램을 이 헤더 블록과 일치시키려면 <namespace> 및 <localname> 요소를 다음과 같이 코드화하십시오.

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

<localname> 요소에서 별표(*)를 코드화하여 이름이 지정된 문자열로 시작되는 네임스페이스의 모든 헤더 블록이 처리되어야 함을 표시할 수 있습니다. 예를 들어, 다음과 같습니다.

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

<localname> 요소에 별표를 사용하면 메시지의 헤더가 둘 이상의 <headerprogram> 요소와 일치할 수 있습니다. 예를 들어, 이 헤더 블록

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

은 다음과 같은 모든 <headerprogram> 요소와 일치합니다.

```
<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>
```

이 경우 실행되는 헤더 프로그램은 헤더 블록의 요소 이름이 가장 정확하게 명시된 <headerprogram> 요소에서 지정되는 헤더 프로그램입니다. 예제에서 이러한 헤더 프로그램은 HDRPROG3입니다.

SOAP 메시지에 둘 이상의 헤더가 있는 경우 헤더 처리 프로그램은 헤더 일치가 발생할 때마다 한 번씩 호출되지만 헤더가 처리되는 순서는 정의되지 않습니다.

같은 <namespace> 및 <localname> 요소를 포함하지만 서로 다른 헤더 프로그램을 지정하는 둘 이상의 <headerprogram> 요소를 코드화하는 경우 헤더 프로그램 중 하나만 실행되지만 그 중 실행될 프로그램은 정의되지 않습니다.

- d. XML 부울 값(true 또는 false)을 포함하는 <mandatory> 요소. 또는 값을 각각 1 또는 0으로 코드화할 수 있습니다.

true

서비스 제공자 파이프라인에서 서비스 요청을 처리하는 동안 및 서비스 요청자 파이프라인에서 서비스 응답을 처리하는 동안 SOAP 메시지의 헤더 중 <namespace> 및 <localname> 요소와 일치하는 헤더가 없는 경우에도 헤더 처리 프로그램이 최소 한 번 이상 호출됩니다.

- 일치하는 헤더가 없는 경우에도 헤더 처리 프로그램은 한 번 호출됩니다.
- 일치하는 헤더가 있는 경우에 헤더 처리 프로그램은 헤더가 일치할 때마다 한 번씩 호출됩니다.

서비스 요청자 파이프라인에서 서비스 요청을 처리하는 동안 및 서비스 제공자 파이프라인에서 서비스 응답을 처리하는 동안 CICS가 작성하는 SOAP 메시지에 처음부터 헤더가 없는 경우라도 헤더 처리 프로그램은 최소 한 번 이상 호출됩니다. 메시지에 헤더를 추가하려면 <mandatory>true</mandatory> 또는 <mandatory>1</mandatory>를 지정하여 최소 한 개 이상의 헤더 처리 프로그램이 호출되도록 해야 합니다.

false

SOAP 메시지의 헤더가 한 개 이상 <namespace> 및 <localname> 요소와 일치하는 경우에만 헤더 처리 프로그램이 호출됩니다.

- 일치하는 헤더가 없는 경우 헤더 처리 프로그램은 호출되지 않습니다.
- 일치하는 헤더가 있는 경우에 헤더 처리 프로그램은 헤더가 일치할 때마다 한 번씩 호출됩니다.

예제

다음 예제는 Java 기반 SOAP 핸들러에 대한 XML 및 해당하는 중첩된 요소를 보여줍니다.

```

<cics_soap_1.2_handler_java>
  <jvmserver>JVMSEV1</jvmserver>
  <headerprogram>
    <program_name>HDRPROG4</program_name>
    <namespace>http://mynamespace</namespace>
    <localname>myheaderblock</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler_java>

```

<default_http_transport_handler_list> 요소

HTTP 전송을 사용 중인 경우 기본적으로 호출되는 메시지 핸들러를 지정합니다.

서비스 공급자에서 이 목록에 지정된 메시지 핸들러는 <named_transport_entry> 요소에서 정의된 핸들러 목록이 덜 구체적인 경우에만 호출됩니다.

사용 위치:

- 서비스 공급자
- 서비스 요청자

다음에 포함됨:

- <transport> 요소

포함 대상:

- 한 개 이상의 <handler> 요소

예제

```

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

```

<default_mq_transport_handler_list> 요소

WebSphere MQ 전송을 사용 중인 경우 기본적으로 호출되는 메시지 핸들러를 지정합니다.

서비스 공급자에서 이 목록에 지정된 메시지 핸들러는 <named_transport_entry> 요소에서 정의된 핸들러 목록이 덜 구체적인 경우에만 호출됩니다.

사용 위치:

- 서비스 공급자
- 서비스 요청자

다음에 포함됨:

- <transport> 요소

포함 대상:

- 한 개 이상의 <handler> 요소

예제

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

<default_transport_handler_list> 요소

전송을 사용 중인 경우 기본적으로 호출되는 메시지 핸들러를 지정합니다.

서비스 제공자에서 이 목록에 지정된 메시지 핸들러는 다음 요소 중 한 요소에서 정의된 핸들러 목록이 덜 구체적인 경우에만 호출됩니다.

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

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

- <transport> 요소

포함 대상:

- 한 개 이상의 <handler> 요소

예제

```
<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> 요소

메시지 핸들러 프로그램의 속성을 지정합니다.

일부 CICS 제공 핸들러 프로그램은 <handler> 요소를 사용하지 않습니다. 예를 들어, CICS 제공 SOAP 메시지 핸들러 프로그램은 <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 및 <cics_soap_1.2_handler_java> 요소를 사용하여 정의됩니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

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

포함 대상:

1. 핸들러 프로그램의 이름을 포함하는 <program> 요소
2. 컨테이너 DFH-HANDLERPLIST에서 메시지 핸들러에 사용할 수 있도록 설정된 XML 요소를 포함하는 <handler_parameter_list> 요소

예제

```
<?xml version="1.0"?>
<provider_pipeline>
  xmlns="http://www.ibm.com/software/htp/cics/pipeline">
  <service>
    <service_handler_list>
      <handler>
        <program>MYPROG</program>
        <handler_parameter_list><output print="yes"/></handler_parameter_list>
      </handler>
    </service_handler_list>
    <terminal_handler>
      <cics_soap_1.1_handler>
        ...
      </cics_soap_1.1_handler>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```


이 예제에서 핸들러 프로그램은 MYPROG입니다. 이 핸들러 매개변수 목록은 단일 <output> 요소로 구성되고 이 매개변수 목록의 콘텐츠는 MYPROG에 알려집니다.

<jvmserver> 요소

JVMSERVER 자원의 이름을 지정합니다.

이 요소는 요청을 처리하는 JVMSERVER 자원의 이름을 식별합니다. 값이 제공되지 않은 경우에는 오류 메시지가 생성되고 파이프라인이 사용 안함 상태로 설치됩니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

- 구성에서 <cics_json_handler_java> 요소 요소
- 구성에서 <cics_soap_1.1_handler_java> 요소 요소
- 구성에서 <cics_soap_1.2_handler_java> 요소 요소
- 요소

예제

```
<jvmserver>JVMSERVER_NAME</jvmserver>
```

<repository> 요소

Axis2 저장소의 디렉토리 이름을 지정합니다.

이 선택적 요소는 Axis2 저장소의 디렉토리 이름을 식별합니다. 이 옵션을 사용하는 경우 핸들러 XML 앞에 구성에서 <jvmserver> 요소를 지정해야 합니다. 이 요소가 제공되지 않으면 샘플 저장소가 사용됩니다. CICS Transaction Server를 설치할 때 샘플 Axis2 저장소가 /usr/lpp/cicsts/cicsts53/lib/pipeline/repository 디렉토리에 설치됩니다. 여기서, /usr/lpp/cicsts/cicsts53은 z/OS UNIX에서 CICS 파일의 기본 설치 디렉토리입니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

- 구성에서 <cics_json_handler_java> 요소
- 구성에서 <cics_soap_1.1_handler_java> 요소
- 구성에서 <cics_soap_1.2_handler_java> 요소

예제

```
<cics_soap_1.1_handler_java>  
  <jvmserver>JVMSEV1</jvmserver>  
  <repository>/lib/pipeline/repository</repository>  
</cics_soap_1.1_handler_java>
```

<service> 요소

모든 요청에 대해 호출되는 메시지 핸들러를 지정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

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

포함 대상:

1. <service_handler_list> 요소
2. <terminal_handler> 요소(서비스 제공자에서만 해당됨)

예제

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

<service_handler_list> 요소

모든 요청에 대해 호출되는 메시지 핸들러의 목록을 지정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

- <service> 요소

포함 대상:

다음 요소 중 한 개 이상을 포함합니다.

```
  <cics_soap_1.1_handler>
```

```

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

```

<service_handler_list> 요소에서 핸들러 요소를 지정하는 순서에 따라 런타임 시 각 핸들러가 호출되는 순서를 판별합니다. 예를 들어, 파이프라인이 WS-Security를 지원하는 경우 암호화된 SOAP 메시지는 <wsse_handler> 요소가 호출될 때까지 암호화된 상태로 유지됩니다. 따라서 암호화되지 않은 메시지를 처리하는 다른 핸들러 프로그램 앞에 <wsse_handler> 요소를 지정해야 합니다.

서비스 제공자의 <service_handler_list> 요소는 <cics_soap_1.1_handler_java> 및 <cics_soap_1.2_handler_java> 요소를 포함할 수 없습니다. 이러한 요소는 Java 기반 파이프라인의 <terminal_handler> 요소에서 지정되어야 하기 때문입니다. 서비스 요청자는 <cics_soap_1.1_handler_java> 및 <cics_soap_1.2_handler_java>를 포함할 수 있지만 이러한 요소가 사용되는 경우 이들 요소는 <service_handler_list> 요소에 나열되는 첫 번째 요소여야 합니다.

파이프라인이 SOAP 1.1 및 SOAP 1.2 메시지를 둘 다 처리할 것으로 예상하는 경우 <cics_soap_1.2_handler> 또는 <cics_soap_1.2_handler_java> 요소를 사용해야 합니다.

서비스 요청자 파이프라인에서 SOAP 1.1 또는 SOAP 1.2 핸들러를 사용할 수 있지만 이 경우 SOAP 1.2 핸들러는 SOAP 1.1 메시지를 지원하지 않습니다. 서비스 요청자 애플리케이션이 DFHREQUEST 컨테이너에서 완전한 SOAP 엔벨로프를 전송 중인 경우 파이프라인에서 SOAP 1.1 또는 SOAP 1.2 핸들러를 지정하지 마십시오. 이렇게 하면 아웃바운드 메시지에서 SOAP 메시지 헤더가 중복되는 것을 막을 수 있습니다.

서비스 제공자에서는 <service_handler_list> 요소 외에도 <terminal_handler> 요소의 일반 핸들러 및 SOAP 핸들러도 지정할 수 있습니다. SOAP 헤더 처리에 대한 자세한 정보는 339 페이지의 『헤더 처리 프로그램』의 내용을 참조하십시오.

예제

```

<service_handler_list>
  <wsse_handler>
  ...
  </wsse_handler>
  <cics_soap_1.1_handler_java>
  ...
  </cics_soap_1.1_handler_java>

```

```

<handler>
...
</handler>
</service_handler_list>

```

<service_parameter_list> 요소

컨테이너 DFH-HANDLERPLIST에서 파이프라인의 모든 메시지 핸들러에 사용할 수 있도록 설정된 XML 요소를 지정합니다. 이는 선택적 요소입니다.

사용 위치:

- 서비스 요청자
- 서비스 제공자

포함 대상:

- WS-AT를 사용하는 경우: <registration_service_endpoint> 요소
- 서비스 요청자에서 WS-AT를 사용하는 경우: 선택적 <new_tx_context_required/> 요소
- 선택적 사용자 정의 태그

예제

```

<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>

```

관련 참조:

286 페이지의 『<requester_pipeline> 요소』

서비스 요청자에서 파이프라인의 구성을 설명하는 XML 문서의 루트 요소입니다.

283 페이지의 『<provider_pipeline> 요소』

웹 서비스 제공자의 CICS 파이프라인 구성을 설명하는 XML 문서의 루트 요소를 지정합니다.

<transport> 요소

특정 전송을 사용 중인 경우에만 호출되는 핸들러를 지정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

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

포함 대상:

서비스 제공자에서:

1. 선택적 <default_transport_handler_list> 요소
2. 선택적 <default_http_transport_handler_list> 요소
3. 선택적 <default_mq_transport_handler_list> 요소
4. 0, 1 또는 2개 이상의 <named_transport_entry> 요소

서비스 요청자에서:

1. 선택적 <default_target> 요소. <default_target>은 서비스 요청자 애플리케이션이 URI를 제공하지 않을 때 CICS가 대상 웹 서비스를 찾는 데 사용하는 URI를 포함합니다. 그러나 대부분의 경우에 대상의 URI는 서비스 요청자 애플리케이션에 의해 제공되며 <default_target>에 지정하는 값은 무시됩니다. 예를 들어, CICS 웹 서비스 지원을 사용하여 배치되는 서비스 제공자 애플리케이션은 일반적으로 웹 서비스 설명에서 URI를 가져옵니다.
2. 선택적 <default_http_transport_handler_list> 요소
3. 선택적 <default_mq_transport_handler_list> 요소
4. 선택적 <default_transport_handler_list> 요소

예제

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

MTOM/XOP을 위한 파이프라인 구성

CICS SOAP 파이프라인은 MTOM(Message Transmission Optimization Mechanism) 및 XOP(XML-binary Optimized Packaging) 스펙을 지원할 수 있습니다. 이러한 스펙은 base64 인코딩 오버헤드를 발생시키지 않고 SOAP을 사용하여 2진 데이터를 전송하고 수신하기 위한 메커니즘을 정의합니다. MTOM 지원을 사용 설정하려면 그에 따라 파이프라인을 구성해야 합니다.

<mtom> 요소

Java 기반 파이프라인에 대해 MTOM/XOP 지원을 사용합니다. 이 요소가 파이프라인 구성 파일에서 정의되는 경우 모든 인바운드 및 아웃바운드 메시지에 대해 MTOM 지

원이 사용 설정됩니다. 그러나 이 요소가 파이프라인 구성 파일에 지정되지 않으면 인바운드 메시지에만 MTOM 지원을 사용할 수 있습니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

```
<cics_soap_1.1_handler_java>  
<cics_soap_1.2_handler_java>
```

제공자 및 요청자 파이프라인 구성 파일 둘 다에서 <mtom> 요소는 선택적 <addressing> 요소 뒤와 선택적 <headerprogram> 요소 앞에 정의되어야 합니다.

예제

제공자 또는 요청자 모드 파이프라인의 경우 다음을 지정할 수 있습니다.

```
<cics_soap_1.2_handler_java>  
  <jvmserver>JVMSERV1</jvmserver>  
  <addressing></addressing>  
  <mtom></mtom>  
  <headerprogram>  
    <program_name>HDRPROG4</program_name>  
    <namespace>http://mynamespace</namespace>  
    <localname>myheaderblock</localname>  
    <mandatory>true</mandatory>  
  </headerprogram>  
</cics_soap_1.2_handler_java>
```

<cics_mtom_handler> 요소

SOAP 파이프라인에서 제공된 MTOM 핸들러 프로그램을 사용합니다. 이 프로그램은 XOP 문서 및 2진 첨부 파일을 포함하는 MTOM MIME 다중 파트/관련 메시지에 대한 지원을 제공합니다. MTOM 지원은 파이프라인에서 수신되는 모든 인바운드 메시지에 대해 사용으로 설정되지만 아웃바운드 메시지에 대한 MTOM 지원은 추가 옵션에 따라 조건부로 사용 설정됩니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

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

제공자 파이프라인 구성 파일에서 <transport> 앞에 <cics_mtom_handler> 요소를 정의해야 합니다. 런타임 시 MTOM 핸들러 프로그램은 전송 핸들러 등의 다른 핸들러가 처리하기 전에 인바운드 MTOM 메시지의 패키지를 풀어야 합니다. 그러면 이 핸들러 프로그램은 응답 메시지에 대한 마지막 핸들러로 호출되어 웹 서비스 요청자에 전송할 MTOM 메시지를 패키징합니다.

요청자 파이프라인 구성 파일에서 <transport> 요소 뒤에 <cics_mtom_handler> 요소를 정의해야 합니다. 런타임 시 아웃바운드 요청 메시지는 다른 모든 핸들러가 처리하기 전까지 MTOM 형식으로 변환되지 않습니다. 그러면 이 핸들러 프로그램은 인바운드 응답 메시지에 대한 첫 번째 핸들러로 호출되어 다른 핸들러가 처리하기 전에 MTOM 메시지의 패키지를 풀고 요청 프로그램으로 리턴합니다.

참고: 이 핸들러 프로그램은 Java 기반 파이프라인에서 사용하지 않습니다. Java 기반 파이프라인에서 <mtom> 요소를 지정하십시오.

포함 대상:

<dfhmtom_configuration> 요소

기본 옵션은 <dfhmtom_configuration> 요소에서 지정된 구성 옵션을 사용하여 변경할 수 있습니다. 기본 옵션을 변경하지 않으려는 경우 비어 있는 요소를 사용할 수 있습니다.

예제

제공자 모드 파이프라인에 대해 다음을 지정할 수 있습니다.

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

<dfhmtom_configuration> 요소

Java를 지원하지 않는 파이프라인에 대해 제공된 MTOM 핸들러 프로그램의 구성 정보를 지정합니다. 이 프로그램은 XOP 문서와 2진 첨부 파일을 포함하는 MIME 메시지에 대한 지원을 제공합니다. MTOM에 대한 구성을 지정하지 않는 경우 CICS는 기본값을 가정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

<cics_mtom_handler>

속성:

이름	설명
버전	구성 정보의 버전을 표시하는 정수입니다. 유일한 올바른 값은 1입니다.

포함 대상:

- 선택적 <mtom_options> 요소
- 선택적 <xop_options> 요소
- 선택적 <mime_options> 요소

예제

```
<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> 요소

Java를 지원하는 파이프라인의 아웃바운드 SOAP 메시지에 MTOM을 사용하는 시기를 지정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

<dfhmtom_configuration>

속성:

속성	설명
send_mtom	아웃바운드 SOAP 메시지를 MIME 메시지로 변환하는 데 MTOM을 사용해야 하는지 여부를 지정합니다. 아니오 아웃바운드 SOAP 메시지에 MTOM이 사용되지 않습니다. 같음 서비스 제공자 모드에서 요청자가 MTOM을 사용할 때마다 SOAP 응답 메시지에 MTOM이 사용됩니다. 서비스 제공자 파이프라인에서는 이 값이 기본값입니다. 서비스 요청자 모드에서 이 값을 지정하는 것은 send_mtom="yes"를 지정하는 경우와 같습니다. 예 모든 아웃바운드 SOAP 메시지에 MTOM이 사용됩니다. 서비스 요청자 파이프라인에서는 이 값이 기본값입니다.
send_when_no_xop	메시지에 2진 첨부 파일이 없는 경우에도 MTOM 메시지를 전송해야 하는지 여부를 지정합니다. 아니오 2진 첨부 파일이 메시지와 함께 전송되는 경우에만 MTOM이 사용됩니다. 예 MTOM이 모든 아웃바운드 SOAP 메시지에 대해 사용되며 메시지에 전송할 2진 첨부 파일이 없는 경우에도 사용됩니다. 이 값은 기본값이고 발신인이 MTOM/XOP를 지원하는 수신 프로그램에 대한 표시기로 주로 사용됩니다. 이 속성은 send_mtom 속성 값과 결합할 수 있지만 send_mtom="no"를 지정하는 경우에는 효과가 없습니다.

예제

```

<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>

```

이 제공자 파이프라인 예제에서 SOAP 메시지는 2진 첨부 파일이 메시지와 함께 전송되어야 하고 서비스 요청자가 MTOM 메시지를 전송한 경우에만 MTOM 메시지로 변환됩니다.

<xop_options> 요소

XOP 처리가 직접 모드에서 실행되는지 또는 Java를 지원하지 않는 파이프라인의 호환성 모드에서 실행되는지 지정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

<dfhmtom_configuration>

속성:

속성	설명
apphandler_supports_xop	제공자 모드에서 애플리케이션 핸들러가 직접 모드로 XOP 문서를 핸들링할 수 있는지 여부를 지정합니다. 아니오 애플리케이션 핸들러가 XOP 문서를 직접 핸들링할 수 없습니다. <apphandler> 요소가 DFHPITP를 지정하지 않는 경우에 이 값이 기본값입니다. 호환성 모드는 MTOM 형식으로 수신 또는 전송되는 인바운드 및 아웃바운드 메시지를 처리하기 위해 파이프라인에서 사용됩니다. 예 애플리케이션 핸들러가 XOP 문서를 처리할 수 있습니다. <apphandler> 요소가 DFHPITP를 지정하는 경우에 이 값이 기본값입니다. 직접 모드는 MTOM 형식으로 수신 또는 전송되는 인바운드 또는 아웃바운드 메시지를 처리하기 위해 파이프라인에서 사용됩니다. 이는 런타임 시 제한사항의 영향을 받습니다. 예를 들어, 파이프라인 구성 파일에서 WS-Security 관련 요소를 지정한 경우 MTOM 핸들러는 XOP 문서 처리에 직접 모드가 아닌 호환성 모드를 사용하도록 결정합니다. 요청자 모드에서는 서비스 요청자 애플리케이션이 직접 모드로 XOP 문서를 작성하고 처리하는 데 CICS 웹 서비스 지원을 사용하는지 여부를 지정합니다. 아니오 서비스 요청자 애플리케이션이 CICS 웹 서비스 지원을 사용하지 않습니다. 요청자 애플리케이션이 DFHPITP에 링크되어 파이프라인을 구동하고 따라서 직접 모드로 XOP 문서를 작성 및 핸들링할 수 없는 경우에 이 값을 지정하십시오. 예 서비스 요청자 애플리케이션이 CICS 웹 서비스 지원을 사용하지 않습니다. 요청자 애플리케이션이 EXEC CICS INVOKE WEBSERVICE 명령을 사용하는 경우에는 이 값을 지정하십시오.

예제

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

이 제공자 파이프라인 예제에서 인바운드 MTOM 메시지 및 아웃바운드 응답 메시지는 호환성 모드를 사용하여 파이프라인에서 처리됩니다.

<mime_options> 요소

Java를 지원하지 않는 파이프라인의 MIME 콘텐츠 ID 값을 생성할 때 사용해야 하는 도메인 이름을 지정합니다. MIME 콘텐츠 ID 값은 2진 첨부 파일을 식별하는 데 사용됩니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

<dfhmtom_configuration>

속성:

속성	설명
content_id_domain	사용할 구문은 <i>domain.name</i> 입니다. 인터넷 표준을 준수하려면 이름은 유효한 인터넷 호스트 이름이어야 하고 파이프라인이 설치되는 CICS 시스템에서 고유해야 합니다. 이는 CICS에서 확인되지 않는다는 점에 유의하십시오. 이 요소가 생략되면 CICS는 <i>cicsts</i> 값을 사용합니다.

예제

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

이 예제에서 2진 첨부 파일에 대한 참조는 *cid:unique_value@example.org*를 사용하여 작성됩니다.

WS-Security를 위한 파이프라인 구성

웹 서비스 요청자 및 제공자 애플리케이션이 WS-Security 프로토콜에 참여하게 하려면 그에 따라 메시지 핸들러 DFHWSSE를 포함시키고 이 핸들러에 대한 구성 정보를 제공하여 파이프라인을 구성해야 합니다.

예

WS-Security를 사용하는 제공자 파이프라인 구성 파일은 다음과 같은 양식을 가질 수 있습니다.

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

<wsse_handler> 요소

WS-Security에 대한 지원을 제공하는 CICS 제공 메시지 핸들러에서 사용되는 매개변수를 지정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

```
<service_handler_list>
```

포함 대상:

- <dfhwsse_configuration> 요소

제공자 파이프라인 구성 파일에서 WS-Security에 대한 CICS 제공 메시지 핸들러는 암호화된 메시지를 복호화해야 될 수도 있습니다. <wsse_handler> 요소는 암호화되지 않은 메시지 콘텐츠를 처리해야 하는 다른 핸들러 프로그램 앞에 정의되어야 합니다.

요청자 파이프라인 구성 파일에서 WS-Security에 대한 CICS 제공 메시지 핸들러는 메시지를 암호화해야 할 수도 있습니다. 이 메시지 핸들러는 CICS SOAP 핸들러 프로그램을 포함하여 암호화되지 않은 메시지 콘텐츠를 처리해야 하는 다른 핸들러 프로그램 뒤에 정의되어야 합니다.

<dfhwsse_configuration> 요소

웹 서비스 보안을 위한 지원을 제공하는 보안 핸들러 DFHWSSE1에 대한 구성 정보를 지정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

<wsse_handler>

속성:

이름	설명
버전	구성 정보의 버전을 표시하는 정수입니다. 유일한 올바른 값은 1입니다.

포함 대상:

1. 다음 요소 중 하나입니다.

- 선택적 <authentication> 요소
 - 서비스 요청자 파이프라인에서 <authentication> 요소는 아웃바운드 SOAP 메시지의 보안 헤더에서 사용해야 하는 인증 유형을 지정합니다.
 - 서비스 제공자 파이프라인에서 이 요소는 CICS가 작업을 처리할 사용자 ID를 판별하기 위해 인바운드 SOAP 메시지에서 보안 토큰을 사용하는지 여부를 지정합니다.
- 선택적 <sts_authentication> 요소.

이 요소의 action 속성은 보안 토큰 서비스에 전송할 요청 유형을 지정합니다. 요청이 ID 토큰을 발행하는 것이면 CICS는 중첩된 요소의 값을 사용하여 지정된 유형의 ID 토큰을 요청합니다.

2. <sts_authentication> 요소를 지정하는 경우 <sts_endpoint> 요소도 지정해야 합니다.

이 요소가 존재하는 경우 CICS는 <endpoint> 요소의 URI를 사용하여 보안 토큰 서비스에 요청을 전송합니다.

3. 비어 있는 선택적 <expect_signed_body/> 요소.

<expect_signed_body/> 요소는 인바운드 메시지의 <body>이 서명되어야 함을 표시합니다. 인바운드 메시지의 본문이 올바르게 서명되지 않은 경우 CICS는 보안 결함을 이유로 이 메시지를 거부합니다.

4. 비어 있는 선택적 <expect_encrypted_body/> 요소.

<expect_encrypted_body/> 요소는 인바운드 메시지의 <body>가 암호화되어야 함을 표시합니다. 인바운드 메시지의 본문이 올바르게 암호화되지 않은 경우 CICS는 보안 결함을 이유로 이 메시지를 거부합니다.

5. 선택적 <sign_body> 요소.

이 요소가 존재하는 경우 CICS는 <sign_body> 요소에 포함된 <algorithm> 요소에 지정된 알고리즘을 사용하여 아웃바운드 메시지의 <body>에 서명합니다.

6. 선택적 <encrypt_body> 요소.

이 요소가 존재하는 경우 CICS는 <encrypt_body> 요소에 포함된 <algorithm> 요소에 지정된 알고리즘을 사용하여 아웃바운드 메시지의 <body>를 암호화합니다.

7. 선택적 <reject_signature/> 요소(제공자 파이프라인에서만 해당됨).

이 요소가 존재하는 경우 CICS는 메시지 본문의 일부 또는 전체에 서명하는 인증서를 헤더에 포함하고 있는 모든 메시지를 거부합니다. SOAP 결함이 웹 서비스 요청자에게 발행됩니다.

8. 선택적 <reject_encryption/> 요소(제공자 파이프라인에서만 해당됨).

이 요소가 존재하는 경우 CICS는 부분적으로 또는 완전히 암호화되는 모든 메시지를 거부합니다. SOAP 결함이 웹 서비스 요청자에게 발행됩니다.

예제

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

```

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

```

<authentication> 요소

인바운드 및 아웃바운드 SOAP 메시지의 헤더에서 보안 토큰의 사용을 지정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

```
<dfhwsse_configuration>
```

속성:

속성	설명
신뢰 및 모드	함께 사용할 때 신뢰 및 모드 속성은 다음을 지정합니다. • 어설션된 ID가 사용되는지 여부 • SOAP 메시지에서 사용된 보안 토큰의 조합. 어설션된 ID를 사용하면 신뢰 사용자가 해당 ID와 연관된 신임 정보를 가지지 않고도 작업이 다른 ID <i>asserted identity</i>로 실행되어야 함을 어설션할 수 있습니다. 어설션된 ID가 사용되는 경우 메시지에 신뢰 토큰 및 ID 토큰이 포함됩니다. 신뢰 토큰은 발신인이 ID를 어설션하기 위한 올바른 권한을 가지는지 확인하는 데 사용할 수 있습니다. ID 토큰은 어설션된 ID를 유지합니다. 즉, 이 사용자 ID에서 요청이 실행됩니다. 어설션된 ID를 사용하려면 이 어설션을 수행하기 위해 서비스 제공자가 요청자를 신뢰해야 합니다. CICS에서 신뢰 관계는 보안 관리자 대리 정의로 설정되며 요청하는 ID에는 어설션된 ID 대신 작업을 시작할 수 있도록 올바른 권한이 있어야 합니다. 이러한 속성 및 해당 의미의 허용 가능한 조합은 표 4 and 317 페이지의 표 5에서 설명하고 있습니다.

표 4. 서비스 요청자 파이프라인의 모드 및 신뢰 속성

신뢰	모드	의미
없음	없음	메시지에 신임 정보가 추가되지 않음
없음	기본	속성 값의 올바른지 않은 조합
없음	서명	어설션된 ID가 사용되지 않음. CICS가 단일 X.509 보안 토큰을 사용하며 이 토큰은 메시지에 추가되어 메시지 본문에 서명하는 데 사용됩니다. 인증서는 <certificate_label> 요소로 식별하고 알고리즘은 <algorithm> 요소에서 지정합니다.

표 4. 서비스 요청자 파이프라인의 모드 및 신뢰 속성 (계속)

신뢰	모드	의미
블라인드	없음	속성 값의 올바르지 않은 조합
블라인드	기본	어설션된 ID가 사용되지 않음. CICS가 메시지에 ID 토큰은 추가하지만 신뢰 토큰은 제공하지 않습니다. ID 토큰은 비밀번호가 없는 사용자 이름입니다. ID 토큰에 배치된 사용자 ID는 DFHWS-USERID 컨테이너의 콘텐츠입니다(기본적으로 실행 중인 태스크의 사용자 ID가 포함됨).
블라인드	서명	속성 값의 올바르지 않은 조합
기본	없음	속성 값의 올바르지 않은 조합
기본	기본	속성 값의 올바르지 않은 조합
기본	서명	속성 값의 올바르지 않은 조합
서명	없음	속성 값의 올바르지 않은 조합
서명	기본	어설션된 ID가 사용됨. CICS가 다음 토큰을 메시지에 추가합니다. - 신뢰 토큰은 X.509 보안 토큰입니다. - ID 토큰은 비밀번호가 없는 사용자 이름입니다. ID 토큰 및 메시지 본문에 서명하는 데 사용되는 인증서는 <certificate_label>에 의해 지정됩니다. ID 토큰에 배치된 사용자 ID는 DFHWS-USERID 컨테이너의 콘텐츠입니다(기본적으로 실행 중인 태스크의 사용자 ID가 포함됨).
서명	서명	속성 값의 올바르지 않은 조합

표 5. 서비스 제공자 파이프라인의 모드 및 신뢰 속성

신뢰	모드	의미
없음	없음	인바운드 메시지는 신임 정보가 포함되지 않아도 되고 CICS는 메시지에서 찾은 신임 정보를 추출하거나 확인하려고 시도하지 않습니다. 그러나 CICS는 서명된 요소가 올바르게 서명되었는지 확인합니다.
없음	기본	인바운드 메시지는 비밀번호가 있는 사용자 이름 보안 토큰이 포함되어야 합니다. CICS는 사용자 이름을 DFHWS-USERID 컨테이너에 넣습니다.
없음	기본 ICRX	속성 값의 올바르지 않은 조합
없음	기본 kerberos	속성 값의 올바르지 않은 조합
없음	서명	인바운드 메시지는 메시지 본문에 서명하는 데 사용된 X.509 보안 토큰이 포함되어야 합니다.
블라인드	없음	속성 값의 올바르지 않은 조합
블라인드	기본	인바운드 메시지는 ID 토큰에 사용자 ID 및 선택적으로 비밀번호가 포함되는 ID 토큰이 포함되어야 합니다. CICS는 사용자 ID를 DFHWS-USERID 컨테이너에 넣습니다. 비밀번호가 포함되지 않은 경우 CICS는 비밀번호를 확인하지 않고 사용자 ID를 사용합니다. 비밀번호가 포함된 경우 보안 핸들러 DFHWSSE1이 비밀번호를 확인합니다.

표 5. 서비스 제공자 파이프라인의 모드 및 신뢰 속성 (계속)

신뢰	모드	의미
블라인드	기본 ICRX	인바운드 메시지는 ICRX ID 토큰이 포함되어야 합니다. CICS는 ID를 분석하여 사용자 ID를 DFHWS-USERID 컨테이너에 넣고 ICRX를 컨테이너 DFHWS-ICRX에 넣습니다. 필요한 경우 인증에서는 클라이언트 인증 SSL 또는 다른 보안 프로토콜을 사용합니다.
블라인드	기본 kerberos	속성 값의 올바른지 않은 조합
블라인드	서명	인바운드 메시지는 ID 토큰이 포함되어야 하며 이 메시지 에서 ID 토큰은 SOAP 메시지 헤더에서 첫 번째 X.509 인증서입니다. 이 인증서는 메시지에 서명하지 않아도 됩니다. 보안 핸들러는 일치하는 사용자 ID를 추출하고 추출한 사용자 ID를 DFHWS-USERID 컨테이너에 배치합니다.
기본	없음	속성 값의 올바른지 않은 조합
기본	기본	인바운드 메시지는 어설션된 ID를 사용해야 함 - 신뢰 토큰은 비밀번호가 있는 사용자 이름 토큰입니다. - ID 토큰은 비밀번호가 없는 두 번째 사용자 이름 토큰입니다. CICS는 이 사용자 이름을 DFHWS-USERID 컨테이너에 넣습니다.
기본	기본 ICRX	인바운드 메시지는 어설션된 ID를 사용해야 함 - 신뢰 토큰은 비밀번호가 있는 사용자 이름 토큰입니다. CICS는 사용자 ID 및 비밀번호 조합이 올바른지 확인하고 올바른 경우에는 CICS가 어설션된 ICRX 기반 ID를 사용자 ID로 분석합니다. 그런 다음 CICS는 인증된 ID에서 어설션된 ID로 대리 보안 확인을 수행합니다. - ID 토큰은 ICRX이고 이 토큰은 특정 클라이언트 사용자를 식별합니다. CICS는 DFHWS-USERID 컨테이너에 사용자 이름을 넣고 DFHWS-ICRX 컨테이너에 ICRX 를 넣습니다.
기본	기본 kerberos	인바운드 메시지는 어설션된 ID를 사용해야 함. 한 개의 토큰이 필수이고 다음 형식 유형 중 하나인 Kerberos 버전 5 토큰: http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ1510 http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ1510 http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ4120 http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ4120 이 토큰은 Base-64로 인코딩되어야 합니다. CICS는 z/OS용 네트워크 인증 서비스를 사용하여 토큰을 유효성 검증하고 이 토큰과 연관된 사용자 ID를 DFHWS-USERID 컨테이너에 넣습니다.

표 5. 서비스 제공자 파이프라인의 모드 및 신뢰 속성 (계속)

신뢰	모드	의미
기본	서명	인바운드 메시지는 어설션된 ID를 사용해야 함 - 신뢰 토큰은 비밀번호가 있는 사용자 이름 토큰입니다. - ID 토큰은 X.509 인증서입니다. CICS는 인증서와 연관된 사용자 ID를 DFHWS-USERID 컨테이너에 넣습니다.
서명	없음	속성 값의 올바른지 않은 조합
서명	기본	인바운드 메시지는 어설션된 ID를 사용해야 함 - 신뢰 토큰은 X.509 인증서입니다. - ID 토큰은 비밀번호가 없는 사용자 이름 토큰입니다. CICS는 사용자 이름을 DFHWS-USERID 컨테이너에 넣습니다. ID 토큰 및 본문은 X.509 인증서로 서명해야 합니다.
서명	기본 ICRX	인바운드 메시지는 어설션된 ID를 사용해야 함. - 신뢰 토큰은 X.509 인증서가 포함된 서명된 ICRX입니다. CICS는 X.509 인증서를 사용자 ID로 분석하고 XML 서명이 유효한지 확인합니다. CICS는 어설션된 ICRX 기반 ID를 사용자 ID로 분석합니다. 그런 다음 CICS는 인증된 X.509 ID에서 어설션된 ICRX ID로의 대리 보안 확인을 수행합니다. - ID 토큰은 비밀번호가 없는 사용자 이름 토큰입니다. CICS는 DFHWS-USERID 컨테이너에 사용자 이름을 넣고 DFHWS-ICRX 컨테이너에 ICRX 를 넣습니다.
서명	기본 kerberos	속성 값의 올바른지 않은 조합
서명	서명	인바운드 메시지는 어설션된 ID를 사용해야 함 - 신뢰 토큰은 X.509 인증서입니다. - ID 토큰은 두 번째 X.509 인증서입니다. CICS는 인증서와 연관된 사용자 ID를 DFHWS-USERID 컨테이너에 넣습니다. ID 토큰 및 본문은 첫 번째 X.509 인증서(신뢰 토큰)로 서명해야 합니다.

참고:

1. 파이프라인 설치 시 신뢰 및 모드 속성 값의 조합이 확인됩니다. 속성이 잘못 코드화된 경우 설치에 실패합니다.
2. CICS는 여기에 설명한 프로세스 중에 비밀번호 검증을 사용하여 사용자 ID를 검증합니다. 기본적으로 비밀번호 검증으로 인해 사용자 ID가 사인온에 사용되었음을 RACF[®]가 기록하지는 않습니다. 감사를 목적으로 이 정보가 필요하거나 시스템이 사용하지 않은 사용자 ID를 취소하도록 설정된 경우 CICS 리전에 대해 시스템 초기화 매개변수 **SECVFYFREQ=USRDELAY**를 지정하십시오. 이 시스템 초기화 매개변수를 설정한 경우 CICS는 CICS 리전에 로그인하는 데 사용되는 각 사용자 ID에 대

해 최소 하루에 한 번 이상 전체 검증 요청을 강제 실행합니다. 전체 검증 요청은 RACF가 사용자 ID의 마지막 액세스 날짜 및 시간을 기록하고 사용자 통계를 작성하도록 합니다.

포함 대상:

1. 선택적으로 비어 있는 <suppress/> 요소.

서비스 제공자 파이프라인에서 이 요소가 지정되는 경우 핸들러는 작업을 실행하는 사용자 ID를 판별하기 위해 메시지의 보안 토큰을 사용하려고 시도하지 않습니다.

서비스 요청자 파이프라인에서 이 요소가 지정되는 경우 핸들러는 아웃바운드 SOAP 메시지에 인증에 필요한 보안 토큰을 추가하려고 시도하지 않습니다.

2. 요청자 파이프라인에서 SOAP 메시지의 본문에 서명하는 데 사용되는 알고리즘의 URI를 지정하는 선택적 <algorithm> 요소. 신뢰 및 모드 속성 값의 조합이 메시지가 서명됨을 나타내는 경우 이 요소를 지정해야 합니다. 이 요소에서 SHA1 알고리즘을 포함하는 RSA만 지정할 수 있습니다. 이 URI는 <http://www.w3.org/2000/09/xmldsig#rsa-sha1>입니다.
3. RACF에 설치된 X.509 디지털 인증서와 연관된 레이블을 지정하는 선택적 <certificate_label> 요소. 서비스 요청자 파이프라인에서 이 요소를 지정하고 <suppress> 요소가 지정되지 않은 경우에 인증서는 SOAP 메시지의 보안 헤더에 추가됩니다. <certificate_label> 요소를 지정하지 않는 경우 CICS는 RACF 키 링의 기본 인증서를 사용합니다.

이 요소는 서비스 제공자 파이프라인에서 무시됩니다.

예제

```
<authentication trust="signature" mode="basic">
  <suppress/>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>AUTHCERT03</certificate_label>
</authentication>
```

<sts_authentication> 요소

인증에 STS(Security Token Service)를 사용하도록 지정하고 전송되는 요청 유형을 판별합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

<dfhwsse_configuration>

속성:

이름	설명
조치	서비스 제공자 파이프라인에서 메시지를 수신할 때 CICS가 STS로 전송하는 요청 유형을 지정합니다. 올바른 값은 다음과 같습니다. issue STS가 SOAP 메시지에 대한 ID 토큰을 발행합니다. 이 값은 제공자 파이프라인의 SAML에 대해 올바르지 않습니다. validate STS가 제공된 ID 토큰을 유효성 검증하고 이 토큰이 유효한지 여부를 보안 핸들러에게 리턴합니다. 이 속성을 지정하지 않으면 CICS는 조치가 ID 토큰 요청인 것으로 가정합니다. 서비스 요청자 파이프라인에서 이 속성을 지정할 수 없습니다. CICS는 항상 STS가 토큰을 발행하도록 요청하기 때문입니다.
추출	이 속성은 SAML을 사용하는 경우에만 유효합니다. SAML 토큰의 요소가 추출되니까? 올바른 값은 다음과 같습니다. 아니오 SAML 토큰의 요소가 컨테이너로 추출되지 않습니다. 예 SAML 토큰의 기본 요소가 CICS에 의해 작성되는 컨테이너에서 추출되고 배치됩니다.
token_signature	이 속성은 SAML을 사용하는 경우에만 유효합니다. 토큰 서명을 제공해야 합니까? 올바른 값은 다음과 같습니다. 무시 제공되는 모든 서명이 무시됩니다. 필수 올바른 서명을 제공해야 합니다. 이는 기본값입니다.

이름	설명
tran_channel	이 속성은 SAML을 사용하는 경우에만 유효합니다. 서비스 제공자 파이프라인에서 이 속성은 파이프라인에서 수신되는 메시지에 포함된 SAML 어설션이 트랜잭션 채널 DFHTRANSACTION의 컨테이너에서 대상 애플리케이션 프로그램에 사용 가능해지는지 여부를 지정합니다. 올바른 값은 다음과 같습니다. 예 SAML 어설션이 DFHTRANSACTION 채널의 컨테이너로 복사되어 프로그램에 사용 가능해집니다. 컨테이너 이름 및 유형에 대한 자세한 정보는 SAML 링크 가능 인터페이스, DFHSAML의 내용을 참조하십시오. 아니오 SAML 어설션이 DFHTRANSACTION 채널을 통해서가 아닌 파이프라인에 의해 프로그램에 전달되는 채널의 컨테이너에서 프로그램에 사용 가능해집니다. 이는 기본값입니다. 서비스 제공자에 대해 이 속성을 지정하지 않으면 어설션은 SOAP 파이프라인에서 프로그램으로 전달되는 채널의 컨테이너에서만 사용 가능해집니다. 서비스 요청자 파이프라인에서 이 속성은 트랜잭션 채널 DFHTRANSACTION의 DFHSAML-OUTTOKEN 컨테이너에 포함된 SAML 토큰이 요청에 대해 사용되는지 여부를 지정합니다. 올바른 값은 다음과 같습니다. 예 DFHTRANSACTION 채널 DFHSAML-OUTTOKEN 컨테이너의 콘텐츠가 요청에 대한 SAML 토큰으로 사용 됩니다. 아니오 파이프라인으로 전달되는 채널에서 DFHSAML-OUTTOKEN 컨테이너의 콘텐츠가 요청의 SAML 토큰으로 사용됩니다. 이는 기본값입니다. 서비스 요청자에 대해 이 속성을 지정하지 않으면 SOAP 파이프라인으로 전달되는 채널의 DFHSAML-OUTTOKEN 컨테이너에서 SAML 토큰을 가져옵니다.

포함 대상:

1. <auth_token_type> 요소. 이 요소는 서비스 요청자 파이프라인에서 <sts_authentication> 요소를 지정하는 경우에는 필수이고 서비스 제공자 파이프라인에서는 선택적입니다. 자세한 정보는 <auth_token_type>의 내용을 참조하십시오.
 - 서비스 요청자 파이프라인에서 <auth_token_type> 요소는 CICS가 DFHWS-USERID 컨테이너에 포함된 사용자 ID를 전송할 때 STS가 발행하는 토큰의 유형입니다. STS에서 CICS가 수신하는 토큰은 아웃바운드 메시지의 헤더에 배치됩니다.

- 서비스 제공자 파이프라인에서 <auth_token_type> 요소는 CICS가 메시지 헤더에서 가져와서 교환 또는 유효성 검증을 위해 STS로 전송하는 ID 토큰을 판별하는 데 사용됩니다. CICS는 메시지 헤더에서 지정된 유형의 ID 토큰을 먼저 사용합니다. 이 요소를 지정하지 않으면 CICS는 메시지 헤더에서 발견하는 첫 번째 ID 토큰을 사용합니다. CICS는 다음을 ID 토큰으로 간주하지 않습니다.
 - wsu:Timestamp
 - xenc:ReferenceList
 - xenc:EncryptedKey
 - ds:Signature
- 2. 비어 있는 선택적 <suppress/> 요소(서비스 제공자 파이프라인에서만 해당됨). 이 요소가 지정되면 핸들러가 작업을 실행하는 사용자 ID를 판별하기 위해 메시지의 보안 토큰을 사용하려고 시도하지 않습니다. <suppress/> 요소는 STS에 의해 리턴되는 ID 토큰을 포함합니다.

예제

다음 예제는 서비스 제공자 파이프라인을 표시하며 여기에서 보안 핸들러는 STS로부터 토큰을 요청합니다.

```
<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> 요소

필요한 ID 토큰의 유형을 지정합니다.

이 요소는 서비스 요청자 파이프라인에서 <sts_authentication> 요소를 지정하는 경우 필수이고 서비스 제공자에서는 선택사항입니다.

- 서비스 요청자 파이프라인에서 <auth_token_type> 요소는 CICS가 DFHWS-USERID 컨테이너에 포함된 사용자 ID를 전송할 때 STS가 발행하는 토큰의 유형입니다. STS에서 CICS가 수신하는 토큰은 아웃바운드 메시지의 헤더에 배치됩니다.
- 서비스 제공자 파이프라인에서 <auth_token_type> 요소는 CICS가 메시지 헤더에서 가져와서 교환 또는 유효성 검증을 위해 STS로 전송하는 ID 토큰을 판별하는 데 사용됩니다. CICS는 메시지 헤더에서 지정된 유형의 ID 토큰을 먼저 사용합니다. 이 요소를 지정하지 않으면 CICS는 메시지 헤더에서 발견하는 첫 번째 ID 토큰을 사용합니다. CICS는 다음을 ID 토큰으로 간주하지 않습니다.
 - wsu:Timestamp

- xenc:ReferenceList
- xenc:EncryptedKey
- ds:Signature

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

<sts_authentication>

포함 대상:

1. <namespace> 요소. 이 요소에는 유효성 검증하거나 교환할 토큰 유형의 네임스페이스가 포함됩니다.

SAML을 사용하는 경우 SAML의 버전에 따라 이 요소의 콘텐츠를 urn:oasis:names:tc:SAML:1.0:assertion 또는 urn:oasis:names:tc:SAML:2.0:assertion으로 설정하십시오.

2. <element> 요소. 이 요소에는 유효성 검증하거나 교환할 토큰 유형의 로컬 이름이 포함됩니다.

SAML에서는 로컬 이름 Assertion을 사용하십시오.

이러한 요소의 값은 토큰의 Qname을 구성합니다.

예제

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

<sts_endpoint> 요소

STS(Security Token Service)의 위치를 지정합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

<dfhwsse_configuration>

포함 대상:

- <endpoint> 요소. 이 요소에는 네트워크에서 STS(Security Token Service)의 위치를 가리키는 URI가 포함됩니다. HTTP를 사용하지 않고 SSL 또는 TLS를 사용하여 STS 보안에 대한 연결을 유지하도록 권장됩니다.

SAML 지원을 사용하려면 엔드포인트를 `cics://PROGRAM/DFHSAML`로 설정하십시오.

JMS 형식의 URI를 사용하여 WebSphere MQ 엔드포인트를 지정할 수도 있습니다.

- 선택적 <jvmserver> 요소. 이 요소는 SAML 토큰 서비스를 실행하도록 구성되는 JVM 서버를 식별합니다. 이 요소가 포함되지 않은 경우 기본 샘플 자원 JVM 서버 DFHXSTS가 가정됩니다. 이 요소는 SAML을 사용하는 경우에만 유효하며 다른 상황에서 사용하는 경우에는 오류가 발생합니다.

예제

이 예제에서는 지정된 URI에서 STS에 대한 보안 연결을 사용하도록 엔드포인트가 구성됩니다.

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

이 예제에서는 CICS SAML 지원을 사용하도록 엔드포인트가 구성됩니다.

```
<sts_endpoint>
  <endpoint>cics://PROGRAM/DFHSAML</endpoint>
</sts_endpoint>
```

<sign_body> 요소

DFHWSSE가 아웃바운드 SOAP 메시지의 본문에 서명하도록 지시하고 메시지가 서명되는 방식에 대한 정보를 제공합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

```
<dfhwsse_configuration>
```

포함 대상:

1. SOAP 메시지의 본문에 서명하는 데 사용되는 알고리즘을 식별하는 URI를 포함하는 <algorithm> 요소.

다음 알고리즘을 지정할 수 있습니다.

알고리즘	URI
DSA with SHA1(Digital Signature Algorithm with Secure Hash Algorithm 1) 인바운드 SOAP 메시지에 대해서만 지원됩니다.	http://www.w3.org/2000/09/xmldsig#dsa-sha1
RSA with SHA1(Rivest-Shamir-Adleman algorithm with Secure Hash Algorithm 1)	http://www.w3.org/2000/09/xmldsig#rsa-sha1

2. RACF에 설치된 디지털 인증서와 연관된 레이블을 지정하는 <certificate_label> 요소. 디지털 인증서는 메시지에 서명하는 데 사용되는 키를 제공합니다.

예제

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

<encrypt_body> 요소

DFHWSSE가 아웃바운드 SOAP 메시지의 본문을 암호화하도록 지시하고 메시지가 암호화되는 방식에 대한 정보를 제공합니다.

사용 위치:

- 서비스 제공자
- 서비스 요청자

다음에 포함됨:

```
<dfhwsse_configuration>
```

포함 대상:

1. SOAP 메시지의 본문을 암호화하는 데 사용되는 알고리즘을 식별하는 URI를 포함하는 <algorithm> 요소.

다음 알고리즘을 지정할 수 있습니다.

알고리즘	URI
3-DES(Data Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmenc#tripledes-cbc
키 길이가 128비트인 AES(Advanced Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmenc#aes128-cbc
키 길이가 192비트인 AES(Advanced Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmenc#aes192-cbc

알고리즘	URI
키 길이가 256비트인 AES(Advanced Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmenc#aes256-cbc

2. RACF의 디지털 인증서와 연관된 레이블을 지정하는 <certificate_label> 요소.
디지털 인증서는 메시지를 암호화하는 데 사용되는 키를 제공합니다.

예제

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

애플리케이션 핸들러

애플리케이션 핸들러는 SOAP 서비스 제공자 파이프라인의 터미널 핸들러가 런타임 시 링크되는 CICS 프로그램입니다.

애플리케이션 핸들러는 터미널 핸들러가 제공된 SOAP 메시지 핸들러 중 하나인 제공자 모드 파이프라인에서 사용됩니다. 이 상황은 <terminal_handler> 요소에 <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소가 포함될 때 발생합니다.

애플리케이션 핸들러는 SOAP 요청의 본문을 처리하고 리턴된 데이터를 사용하여 응답을 생성할 책임이 있습니다. 애플리케이션 핸들러는 이 처리를 완료하기 위해 기타 프로그램을 호출할 수 있습니다. 일반적으로, 애플리케이션 핸들러는 하나 이상의 비즈니스 애플리케이션 주위에 일반 목적 프리젠테이션 계층으로 작동합니다. 이는 해당 애플리케이션을 첨부하여 애플리케이션이 사용할 수 있는 양식으로 XML을 맵핑하고 리턴된 데이터를 사용하여 응답을 생성할 책임이 있습니다.

애플리케이션 핸들러는 두 가지 방식으로 CICS에 의해 첨부될 수 있습니다. 일반 메커니즘은 채널 및 제어 컨테이너를 포함합니다. 다른 메소드는 Axis2용 Java 바인딩을 포함합니다.

채널 첨부 애플리케이션 핸들러는 <provider_pipeline> 요소의 <apphandler> 요소에 지정됩니다. 런타임 시, DFHWS-APPHANDLER 컨테이너는 <apphandler>의 콘텐츠로 채워집니다. 그러나, DFHWS-APPHANDLER 컨테이너는 기타 메시지 핸들러에 의해 동적으로 갱신될 수 있습니다. 따라서, 런타임 시 링크되는 프로그램은 <apphandler> 요소에 지정되는 프로그램과 다를 수 있습니다. 다음 애플리케이션 핸들러는 DFHWS-APPHANDLER 컨테이너 또는 <apphandler> 요소에 지정될 수 있습니다.

- 제공된 채널 첨부 SOAP 애플리케이션 핸들러, DFHPITP. 채널 첨부 애플리케이션 핸들러에 대한 자세한 정보는 『채널 첨부 애플리케이션 핸들러』의 내용을 참조하십시오.
- 사용자 자체 채널 첨부 애플리케이션 핸들러. 이 애플리케이션 핸들러는 Java 이외의 언어로 작성될 수 있습니다. 채널 첨부 애플리케이션 핸들러에 사용될 수 있는 제어 컨테이너에 대한 자세한 정보는 346 페이지의 『제어 컨테이너』의 내용을 참조하십시오.
- ApplicationHandler Java 인터페이스를 구현하고 Axis2 MessageContext를 사용하여 파이프라인에 첨부되는 Java 기반 파이프라인에 대한 Java 애플리케이션 핸들러. ApplicationHandler Java 인터페이스에 대한 자세한 정보는 참조 -> 애플리케이션 개발 참조서의 JCICS 클래스 참조의 내용을 참조하십시오.

Axis2용 Java 바인딩을 사용하는 애플리케이션 핸들러를 사용하려면

<provider_pipeline> 요소의 <apphandler_class> 요소를 지정해야 합니다. Axis2 애플리케이션 핸들러는 JVM 서버가 실행할 웹 서비스 파이프라인 및 애플리케이션 핸들러에 대해 존재해야 하고 웹 서비스 파이프라인의 터미널 핸들러가 <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 메시지 핸들러여야 합니다. 제공 Axis2 애플리케이션 핸들러를 사용하려면 <apphandler_class> 요소에 com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler를 지정해야 합니다. 그러나, Axis2 애플리케이션 핸들러 클래스에서 지정할 수 있습니다. 런타임 시, DFHWS-APPHANCLAS 컨테이너는 <apphandler_class>의 콘텐츠로 채워집니다.

CICS 웹 서비스 지원을 사용하여 배치되는 웹 서비스 애플리케이션의 경우, <apphandler> 요소에서 DFHPITP를 사용하는 애플리케이션 핸들러 또는 DFHPITP를 지정해야 합니다. 또는 <apphandler_class> 요소에서 com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler를 지정해야 합니다. CICS 웹 서비스 지원에 대한 자세한 정보는 62 페이지의 『CICS 웹 서비스 지원자』의 내용을 참조하십시오.

웹 서비스 배치의 Axis2 스타일을 사용하여 CICS에서 제공자 모드 웹 서비스로서 Axis2 애플리케이션을 배치할 수도 있습니다. 자세한 정보는 172 페이지의 『Java 제공자 모드 웹 서비스를 Axis2 JVM 서버에 배치』의 내용을 참조하십시오.

채널 첨부 애플리케이션 핸들러

채널 첨부 애플리케이션 핸들러는 채널 및 제어 컨테이너를 사용하여 CICS에 첨부되는 애플리케이션 핸들러입니다.

애플리케이션 핸들러가 사용하는 채널은 DFHAHC-V1 채널입니다. 이 채널은 터미널 핸들러와 제공자 모드 웹 서비스 애플리케이션 사이에 다음 컨테이너를 전달합니다.

DFHWS-XMLNS

네임스페이스 접두부를 네임스페이스에 맵핑하는 이름값 쌍의 목록을 포함합니다.

- 입력 시, 목록에는 SOAP 엔벨로프의 범위에 있는 네임스페이스가 포함됩니다.
- 출력 시, 목록에는 엔벨로프 태그에 있을 것으로 가정되는 네임스페이스 데이터가 포함됩니다.

DFHWS-BODY

SOAP 엔벨로프의 본문 섹션을 포함합니다. 일반적으로, 애플리케이션은 콘텐츠를 수정합니다. 애플리케이션이 콘텐츠를 수정하지 않으면, 애플리케이션 핸들러 프로그램은 터미널 핸들러로 돌아가기 전에 컨테이너에 동일한 콘텐츠를 위치시킨다고 해도 이 컨테이너의 콘텐츠를 갱신해야 합니다.

DFHNORESPONSE

서비스 요청자 파이프라인의 요청 단계에서 서비스 제공자가 응답을 리턴할 것으로 예상하지 않는 것을 표시합니다. 컨테이너 DFHNORESPONSE의 콘텐츠가 정의되지 않습니다. 서비스 제공자가 응답을 리턴할 것으로 예상되는지 여부를 알아야 하는 메시지 핸들러는 컨테이너가 존재하는지 여부를 판별해야만 합니다.

- 컨테이너 DFHNORESPONSE가 존재하면 응답이 예상되지 않습니다.
- 컨테이너 DFHNORESPONSE가 없으면 응답이 예상됩니다.

채널은 터미널 핸들러에 전달된 모든 컨텍스트 컨테이너도 전달합니다. 예를 들어, 헤더 처리 프로그램은 채널에 컨테이너를 추가할 수 있습니다. 이러한 컨테이너는 사용자 컨테이너로 전달됩니다. 애플리케이션 핸들러에 대한 자세한 정보는 327 페이지의 『애플리케이션 핸들러』의 내용을 참조하십시오.

메시지 핸들러

메시지 핸들러는 입력 동안 웹 서비스 요청을 처리하고 출력 동안 응답을 처리하기 위해 사용되는 CICS 프로그램입니다. 메시지 핸들러는 채널 및 컨테이너를 사용하여 서로 상호작용하고 시스템과 상호작용합니다.

메시지 핸들러 인터페이스는 사용자가 메시지 핸들러 프로그램에서 다음 태스크를 수행하도록 합니다.

- 이를 변경하지 않고 XML 또는 JSON 요청자 또는 응답자의 콘텐츠를 검사하십시오
- XML 또는 JSON 요청 또는 응답의 콘텐츠를 변경하십시오
- 비터미널 메시지 핸들러에서 XML 또는 JSON 요청 또는 응답을 파이프라인의 다음 메시지 핸들러로 전달하십시오

- 터미널 메시지 핸들러에서 애플리케이션 프로그램을 호출하고 응답을 생성하십시오
- 파이프라인의 요청 단계에서 요청을 흡수하고 응답을 생성하여 응답 단계로 상태를 강제 실행하십시오
- 오류 핸들링

팁: SOAP 핸들러, <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java>를 사용하여 SOAP 메시지에 대해 작업하는 것이 좋습니다. 이러한 핸들러는 SOAP 메시지(SOAP 헤더 및 SOAP 본문)에서 기본 요소에 대해 직접적으로 작업하도록 합니다.

메시지 핸들러로 사용되는 모든 프로그램은 동일한 인터페이스로 호출됩니다. 이는 많은 컨테이너를 보유하는 채널로 호출됩니다. 컨테이너가 다음 유형으로 분류될 수 있습니다.

제어 컨테이너

이는 파이프라인의 조작에 필수적입니다. 메시지 핸들러는 제어 컨테이너를 사용하여 후속 핸들러가 처리되는 순서를 수정할 수 있습니다.

컨텍스트 컨테이너

일부 상황에서 메시지 핸들러 프로그램에는 호출되는 컨텍스트에 대한 정보가 필요합니다. CICS에서는 프로그램에 전달되는 컨텍스트 컨테이너 세트에 이 정보를 제공합니다.

일부 컨텍스트 컨테이너는 메시지 핸들러에서 변경할 수 있는 정보를 보유합니다. 예를 들어, 서비스 제공자 파이프라인에서 적합한 컨텍스트 컨테이너의 콘텐츠를 수정하여 대상 애플리케이션 프로그램의 트랜잭션 ID 및 사용자 ID를 변경할 수 있습니다.

사용자 컨테이너

여기에는 하나의 메시지 핸들러가 다른 메시지 핸들러로 전달해야 하는 정보가 포함됩니다. 사용자 컨테이너 사용은 전적으로 메시지 핸들러의 문제입니다.

제한사항: 사용자 컨테이너에 대해 DFH로 시작하는 이름을 사용하지 마십시오.

메시지 핸들러 프로토콜

파이프라인의 메시지 핸들러는 요청 및 응답 메시지를 처리합니다. 핸들러의 동작은 메시지 핸들러가 제공된 상황에서 취할 수 있는 조치를 설명하는 프로토콜 세트에 의해 처리됩니다.

파이프라인의 각 비터미널 메시지 핸들러가 두 번 호출됩니다.

1. 처음에 요청(서비스 제공자 파이프라인에 대한 인바운드 요청, 서비스 요청자에 대한 아웃바운드 요청)을 처리하기 위해 진행됩니다.

2. 두 번째에 세 가지 이유 중 하나에 대해 진행됩니다.

- 응답을 처리하기 위해(서비스 제공자 파이프라인에 대한 아웃바운드 응답, 서비스 요청자에 대한 인바운드 응답)
- 파이프라인의 어느 곳에서든 오류를 따르는 복구를 수행하기 위해
- 응답이 없을 때 필요한 추가 처리를 수행하기 위해.

서비스 제공자 파이프라인의 터미널 메시지 핸들러는 요청을 처리하기 위해 한 번 호출됩니다.

메시지 핸들러는 다양한 이유로 파이프라인에서 제공되고 각 핸들러가 수행하는 처리가 매우 다를 수 있습니다. 특히,

- 일부 메시지 핸들러는 메시지 콘텐츠를 변경하지 않고, 파이프라인의 일반 처리 순서를 변경하지 않습니다
- 일부 메시지 핸들러가 메시지 콘텐츠를 변경하지만, 파이프라인의 일반 처리 순서를 변경하지 마십시오
- 일부 메시지 핸들러는 파이프라인의 처리 순서를 변경합니다.

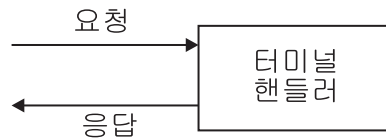
각 핸들러에는 수행할 수 있는 조치의 선택사항이 있습니다. 선택사항은 다음에 따라 다릅니다.

- 핸들러가 서비스 제공자 또는 서비스 요청자로 호출되는지 여부
- 서비스 제공자에서 핸들러가 터미널 핸들러인지 여부
- 핸들러가 요청 또는 응답 메시지에 대해 호출되는지 여부.

터미널 핸들러 프로토콜

일반 요청 및 응답

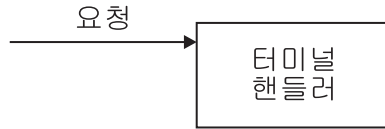
이는 터미널 핸들러에 대한 일반 프로토콜입니다. 핸들러가 요청 메시지에 대해 호출되고 응답을 구성합니다.



응답을 구성하려면 일반 터미널 핸들러가 대상 애플리케이션 프로그램에 링크 되지만, 이는 필수적인 것은 아닙니다.

어떤 응답 없이, 정상 요청

이는 터미널 핸들러에 대한 다른 공통 프로토콜입니다.

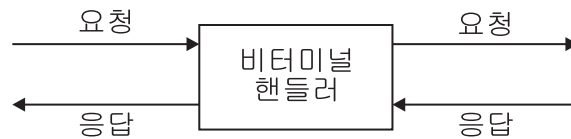


이 프로토콜은 의사결정이 터미널 핸들러에서 작성될 수도 있을지라도 요청에 대한 응답이 없어야 한다는 것을 대상 애플리케이션이 판별할 때 주로 발생합니다.

비터미널 핸들러 프로토콜

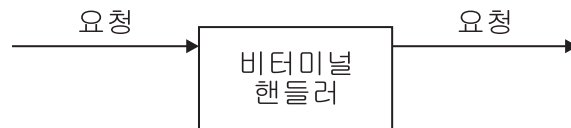
일반 요청 및 응답

이는 비터미널 핸들러에 대한 보통의 프로토콜입니다. 핸들러가 요청 메시지에 대해 호출되고 응답 메시지에 대해 다시 호출됩니다. 각각의 경우, 핸들러는 메시지를 처리하고 파이프라인에서 이를 다음 핸들러로 전달합니다.



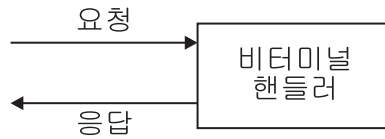
어떤 응답 없이, 정상 요청

이는 비터미널 핸들러에 대한 다른 공통 프로토콜입니다. 핸들러가 요청 메시지에 대해 호출되고 이를 처리한 후 파이프라인에서 다음 핸들러로 전송됩니다. 대상 애플리케이션(또는 다른 핸들러)은 응답이 없어야 한다는 것을 판별합니다. 핸들러가 두 번째 호출될 때, 처리할 응답 메시지가 없습니다.



핸들러는 응답을 작성합니다

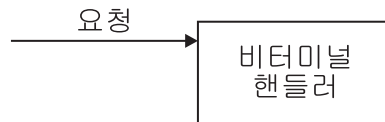
비터미널 핸들러가 다음 핸들러로 요청을 전달하지 않으므로 이 프로토콜은 일반적으로 비정상 상황에서 사용됩니다. 응답을 구성하는 대신 파이프라인에 리턴합니다.



이 프로토콜은 요청이 어떤 경우 올바르지 않고 요청의 추가 처리가 시도되지 않는다는 것을 핸들러가 판별할 때 사용될 수 있습니다. 이 상황에서 핸들러는 두 번 호출되지 않습니다.

핸들러는 응답을 억제합니다

비터미널 핸들러가 다음 핸들러로 요청을 전달하지 않으므로 이는 비정상 상황에서 주로 사용되는 다른 프로토콜입니다. 이 프로토콜에서 핸들러는 요청에 대한 응답이 없어야 한다는 것을 판별합니다.



이 프로토콜은 원래 요청에 대해 예상되는 응답이 없을 때 사용될 수 있고 요청이 어떤 경우 올바르지 않으므로 시도되어야 하는 요청의 추가 처리가 없습니다.

자체 메시지 핸들러 제공

서비스 요청자와 서비스 제공자 사이에서 플로우하는 메시지에 대한 특수 처리를 수행하려고 하고 CICS가 사용자의 요구에 맞는 메시지 핸들러를 제공하지 않는 경우 자체 핸들러를 작성해야 합니다.

이 태스크 정보

대부분의 경우 CICS 제공 메시지 핸들러를 사용하여 필요한 모든 처리를 수행할 수 있습니다. 예를 들어, CICS가 제공하는 SOAP 1.1 및 1.2 메시지 핸들러를 사용하여 SOAP 메시지를 처리할 수 있습니다. 그러나 웹 서비스 요청 및 응답에 대해 고유의 특수화된 조작을 수행해야 하는 경우가 있습니다. 이를 수행하려면 자체 메시지 핸들러를 제공해야 합니다.

프로시저

1. 메시지 핸들러 프로그램을 작성하십시오. 메시지 핸들러는 채널 인터페이스가 있는 CICS 프로그램입니다. CICS가 지원하는 언어로 프로그램을 작성하고 프로그램 내 DPL 서브세트에 있는 모든 CICS 명령을 사용할 수 있습니다.
2. 프로그램을 컴파일 및 링크 편집하십시오. 메시지 핸들러 프로그램은 일반적으로 속성 TASKDATALOC(ANY)로 정의되는 트랜잭션 CPIH에서 실행됩니다. 따라서 프로그램을 링크 편집할 때에는 AMODE(31) 옵션을 지정해야 합니다.
3. 평소 방식대로 CICS 시스템에 프로그램을 설치하십시오.
4. 파이프라인 구성 파일에서 프로그램을 정의하십시오. <handler> 요소를 사용하여 메시지 핸들러를 정의하십시오. <handler> 요소 내에서 프로그램의 이름을 포함하는 <program> 요소를 코드화하십시오.

비터미널 메시지 핸들러에서 메시지에 대한 작업

일반적인 비터미널 메시지 핸들러는 메시지를 처리한 다음 파이프라인의 다른 메시지 핸들러로 제어를 전달합니다.

이 태스크 정보

비터미널 메시지 핸들러에서 변경을 하거나 변경 없이 요청 또는 응답에 대한 작업을 수행한 후 다음 메시지 핸들러에게 전달할 수 있습니다.

참고: 일반적으로 웹 서비스는 XML을 포함하는 SOAP 메시지를 사용하지만 메시지 핸들러는 다른 메시지 형식에 대해서도 작업합니다.

프로시저

1. 컨테이너 DFHFUNCTION의 콘텐츠를 사용하여 이 메시지 핸들러에 전달된 메시지가 요청인지 또는 응답인지를 판별하십시오.

DFHFUNCTION	요청 또는 응답	메시지 핸들러의 유형	인바운드 또는 아웃바운드
RECEIVE-REQUEST	요청	비터미널	인바운드
SEND-RESPONSE	응답	비터미널	아웃바운드
SEND-REQUEST	요청	비터미널	아웃바운드
RECEIVE-RESPONSE	응답	비터미널	인바운드

팁:

- DFHFUNCTION이 PROCESS-REQUEST를 포함하는 경우 이 메시지 핸들러는 터미널 메시지 핸들러이고 다음 단계는 적용되지 않습니다.
- DFHFUNCTION이 HANDLER-ERROR를 포함하는 경우 오류 처리를 위해 핸들러를 호출 중이며 다음 단계는 적용되지 않습니다.

2. 해당 컨테이너에서 요청 또는 응답을 검색하십시오.

- 메시지가 요청인 경우 컨테이너 DFHREQUEST의 프로그램에 전달됩니다. 길이가 0인 컨테이너 DFHRESPONSE도 있습니다.
 - 메시지가 응답인 경우 컨테이너 DFHRESPONSE의 프로그램에 전달됩니다.
3. 필요한 메시지 처리를 수행하십시오. 메시지 핸들러의 용도에 따라 다음을 수행할 수 있습니다.
- 변경 없이 메시지를 조사하고 이 메시지를 파이프라인의 다음 메시지 핸들러로 전달하십시오.
 - 요청을 변경하고 이 요청을 파이프라인의 다음 메시지 핸들러로 전달하십시오.
 - 메시지가 요청인 경우 응답 메시지를 구성하는 대신 파이프라인에서 다음 메시지 핸들러를 우회할 수 있습니다.

참고: 메시지 핸들러가 리턴하는 컨테이너의 콘텐츠이며 다음에 호출될 메시지 핸들러를 판별합니다.

메시지 핸들러가 전달된 컨테이너 중 어느 것도 변경하지 않으면 오류입니다.

다음 중 하나가 리턴되면 메시지 핸들러 프로그램의 오류입니다.

- 비어 있는 DFHRESPONSE 컨테이너
- 비어 있지 않은 DFHREQUEST 컨테이너 및 비어 있지 않은 DFHRESPONSE 컨테이너
- 아웃바운드 요청의 비어 있는 DFHREQUEST 컨테이너

메시지를 파이프라인의 다음 메시지 핸들러로 전달

일반적인 비터미널 메시지 핸들러에서 변경을 하거나 변경 없이 요청 또는 응답을 처리하고 이를 다음 메시지 핸들러로 전달합니다.

프로시저

1. 적절한 컨테이너에서 변경되거나 변경되지 않은 메시지를 파이프라인에 리턴하십시오.
 - 메시지가 요청이고 이를 변경한 경우 이 메시지를 컨테이너 DFHREQUEST에서 리턴합니다.
 - 메시지가 응답이고 이를 변경한 경우 이 메시지를 컨테이너 DFHRESPONSE에 넣습니다.
 - 메시지를 변경하지 않은 경우 이 메시지는 이미 적절한 컨테이너에 있습니다.
2. 메시지가 요청인 경우 컨테이너 DFHRESPONSE를 삭제하십시오. 요청에 대해 메시지 핸들러가 호출되면 컨테이너 DFHREQUEST 및 DFHRESPONSE가 프로그램에 전달되고 DFHRESPONSE의 길이는 0입니다. 그러나 DFHREQUEST 및 DFHRESPONSE가 둘 다 리턴되는 것은 오류입니다.

결과

메시지가 파이프라인의 다음 메시지 핸들러로 전달됩니다.

파이프라인의 응답 단계로 상태 전이를 강제 실행

요청을 처리할 때 요청을 파이프라인의 다음 메시지 핸들러로 전달하는 대신 즉각적 응답을 생성해야 하는 경우가 있습니다.

프로시저

1. 컨테이너 DFHREQUEST를 삭제하십시오.
2. 응답을 구성하고 이 응답을 컨테이너 DFHRESPONSE에 넣으십시오.

결과

응답이 파이프라인의 응답 단계에 있는 다음 메시지 핸들러에 전달됩니다.

응답 억제

응답을 전송하지 않고 요청을 무시해야 하는 경우가 있습니다.

프로시저

1. 컨테이너 DFHREQUEST를 삭제하십시오.
2. 컨테이너 DFHRESPONSE를 삭제하십시오.

서비스 요청자 파이프라인에서 단방향 메시지 핸들링

서비스 요청자 파이프라인이 서비스 제공자에게 요청을 전송할 때 일반적으로 응답이 있을 것이고 요청 전송에 따라 응답이 도착하면 파이프라인의 메시지 핸들러가 다시 호출될 것으로 예상합니다. 일부 웹 서비스는 응답을 전송하지 않기 때문에 두 번째로 메시지 핸들러를 호출하기 전에 CICS가 응답을 대기하지 말아야 함을 표시하기 위한 특별한 조치를 취해야 합니다.

이 태스크 정보

이를 수행하려면 컨테이너 DFHNORESPONSE가 요청 단계에서 처리 중인 파이프라인의 끝에 있는지 확인하십시오. 일반적으로 이 작업은 애플리케이션 레벨 코드에 의해 수행됩니다. 응답이 예상되는지 여부에 대한 지식이 애플리케이션에 저장되기 때문입니다.

- CICS 웹 서비스 지원을 사용하여 배치된 애플리케이션의 경우 CICS 코드가 컨테이너를 작성합니다.
- 지원을 사용하여 배치되지 않은 애플리케이션은 일반적으로 애플리케이션을 호출하기 전에 컨테이너를 작성합니다.

메시지 핸들러에서 컨테이너 DFHNORESPONSE를 작성하거나 영구 삭제하는 경우 그러한 작업으로 서비스 요청자와 제공자 간의 메시지 프로토콜이 방해되지 않는지 확인해야 합니다.

터미널 메시지 핸들러에서 메시지에 대한 작업

일반적인 터미널 핸들러는 요청을 처리하고 애플리케이션 프로그램을 호출하며 응답을 생성합니다.

이 태스크 정보

참고: 일반적으로 웹 서비스는 XML을 포함하는 SOAP 메시지를 사용하지만 메시지 핸들러는 다른 메시지 형식에 대해서도 작업합니다.

터미널 메시지 핸들러에서는 요청에 대해 작업하고 선택적으로 응답을 생성하여 파이프라인을 따라 이 응답을 다시 전달할 수 있습니다. 일반적인 터미널 핸들러는 요청을 애플리케이션 프로그램에 대한 입력으로 사용하고 애플리케이션 프로그램의 응답을 사용하여 응답을 구성합니다.

프로시저

1. 컨테이너 DFHFUNCTION의 콘텐츠를 사용하여 이 핸들러로 전달된 메시지가 요청인지 및 핸들러가 터미널 핸들러로 호출되고 있는지 판별하십시오.

DFHFUNCTION	요청 또는 응답	핸들러의 유형	인바운드 또는 아웃바운드
PROCESS-REQUEST	요청	터미널	인바운드

팁:

- DFHFUNCTION에 다른 값이 포함된 경우 이 핸들러는 터미널 핸들러가 아니며 다음 단계는 적용되지 않습니다.
2. 컨테이너 DFHREQUEST에서 요청을 검색하십시오. 길이가 0인 컨테이너 DFHRESPONSE도 있습니다.
 3. 필요한 메시지 처리를 수행하십시오. 일반적으로 터미널 핸들러는 애플리케이션 프로그램을 호출합니다.
 4. 응답을 구성하고 이 응답을 컨테이너 DFHRESPONSE에 넣으십시오. 응답이 없는 경우 컨테이너 DFHRESPONSE를 삭제해야 합니다.

결과

응답이 파이프라인의 응답 단계에 있는 다음 핸들러로 전달됩니다. 이 핸들러는 함수 SEND-RESPONSE에 대해 호출됩니다. 응답이 없는 경우 함수 NO-RESPONSE에 대해 다음 핸들러가 호출됩니다.

오류 핸들링

메시지 핸들러는 파이프라인에서 발생할 수 있는 오류를 처리할 수 있도록 디자인되어야 합니다.

이 태스크 정보

메시지 핸들러 프로그램에 오류가 발생하면 오류 처리를 위해 프로그램이 다시 호출됩니다. 오류 처리는 항상 파이프라인의 응답 단계에서 발생하고 오류가 요청 단계에서 발생한 경우에는 요청 단계의 후속 핸들러는 우회됩니다.

따라서 대부분의 경우에는 발생할 수 있는 모든 오류를 처리할 수 있도록 핸들러 프로그램을 작성해야 합니다.

프로시저

1. 컨테이너 DFHFUNCTON에 오류 처리를 위해 메시지 핸들러가 호출되었음을 나타내는 HANDLER-ERROR가 포함되어 있는지 확인하십시오.

답:

- DFHFUNCTON에 다른 값이 포함되어 있을 경우 메시지 핸들러가 오류 처리를 위해 호출되지 않은 것이기 때문에 다음과 같은 단계는 적용되지 않습니다.
2. 오류 정보를 분석하고 메시지 핸들러가 적절한 응답을 구성하여 오류를 복구할 수 있는지 판별하십시오.

컨테이너 DFHERROR은 오류에 대한 정보를 보유합니다. 이 컨테이너에 대한 자세한 정보는 347 페이지의 『DFHERROR 컨테이너』의 내용을 참조하십시오.

길이가 0인 컨테이너 DFHRESPONSE도 있습니다.

3. 복구 처리를 수행하십시오.
 - 메시지 핸들러가 복구할 수 있는 경우 응답을 구성하고 이 응답을 컨테이너 DFHRESPONSE에서 리턴합니다.
 - 메시지 핸들러가 복구할 수 있지만 응답이 필요하지 않은 경우에는 컨테이너 DFHRESPONSE를 삭제하고 대신 컨테이너 DFHNORESPONSE를 리턴합니다.
 - 메시지 핸들러가 복구할 수 없는 경우 변경되지 않은(즉, 길이가 0임) 컨테이너 DFHRESPONSE를 리턴합니다.

결과

메시지 핸들러가 오류를 복구할 수 있는 경우에는 일반적으로 파이프라인 처리가 계속됩니다. 그렇지 않으면 CICS가 오류에 대한 정보를 포함하는 SOAP 결함을 생성합니다. 트랜잭션 이상 종료의 경우 이상 종료 코드는 결함에 포함됩니다.

메시지 핸들러 인터페이스

CICS 파이프라인은 여러 컨테이너를 포함하는 채널을 사용하여 메시지 핸들러에 링크됩니다. 일부 컨테이너는 선택적이며 모든 메시지 핸들러에 필요한 컨테이너도 있고 특정 메시지 핸들러는 사용하고 다른 핸들러는 사용하지 않는 컨테이너도 있습니다.

핸들러가 호출되기 전에 핸들러가 작업을 수행하기 위해 사용할 수 있는 정보로 일부 또는 모든 컨테이너를 채웁니다. 핸들러가 리턴하는 컨테이너가 후속 처리를 결정하며 파이프라인에서 다음 핸들러로 전달됩니다.

SOAP 메시지 핸들러

SOAP 메시지 핸들러는 SOAP 1.1 및 SOAP 1.2 메시지를 처리하기 위해 파이프라인에 포함시킬 수 있는 CICS 제공 메시지 핸들러입니다. 서비스 요청자 또는 서비스 제공자 파이프라인에서 SOAP 메시지 핸들러를 사용할 수 있습니다.

입력 시, SOAP 메시지 핸들러가 인바운드 SOAP 메시지를 구문 분석하고 사용자 애플리케이션 프로그램용으로 SOAP <Body> 요소를 추출합니다. 출력 시, 애플리케이션이 제공하는 <Body> 요소를 사용하여 핸들러가 완료 SOAP 메시지를 구성합니다.

메시지에서 SOAP 헤더를 사용하는 경우, SOAP 핸들러는 사용자가 인바운드 메시지에서 헤더를 처리하고 이를 아웃바운드 메시지에 추가할 수 있도록 하는 사용자 작성 헤더 처리 프로그램을 호출할 수 있습니다.

SOAP 메시지 핸들러 및 헤더 처리 프로그램이 파이프라인 구성 파일에 지정됩니다. Java를 지원하지 않는 파이프라인의 경우, <cics_soap_1.1_handler> 또는 <cics_soap_1.2_handler> 메시지 핸들러가 지정되어야 합니다. Java를 지원하는 파이프라인의 경우, <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 메시지 핸들러가 지정되어야 합니다.

일반적으로, 파이프라인에 하나의 SOAP 핸들러만 필요합니다. 그러나, 하나 이상이 필요한 일부 상황이 있습니다. 예를 들어, SOAP 헤더가 여러 SOAP 핸들러를 정의하여 특정 순서로 처리되는지 확인할 수 있습니다.

<cics_soap_1.1_handler> 및 <cics_soap_1.2_handler> 메시지 핸들러 또는 <cics_soap_1.1_handler_java> 및 <cics_soap_1.2_handler_java> 메시지 핸들러를 동일한 파이프라인에서 정의하지 않아야 합니다. 파이프라인이 SOAP 1.1과 SOAP 1.2 메시지 모두를 처리할 것으로 예상하는 경우, <cics_soap_1.2_handler> 또는 <cics_soap_1.2_handler_java> 메시지 핸들러를 사용해야 합니다.

헤더 처리 프로그램

헤더 처리 프로그램은 SOAP 헤더 블록을 처리하기 위해 CICS 제공 SOAP 1.1 및 SOAP 1.2 메시지 핸들러로부터 링크되는 사용자 작성 CICS 프로그램입니다.

CICS가 지원하는 임의의 언어로 헤더 처리 프로그램을 쓰고 DPL 서브세트에서 CICS 명령을 사용할 수 있습니다. 헤더 처리 프로그램은 기타 CICS 프로그램에 링크될 수 있습니다.

헤더 처리 프로그램에는 채널 인터페이스가 있습니다. 컨테이너는 프로그램이 호출되는 SOAP 헤더 블록 및 SOAP 메시지 본문을 포함하여 헤더 프로그램이 검사하거나 수정할 수 있는 정보를 보유합니다.

헤더 처리 프로그램이 사용할 수 있는 채널 및 컨테이너가 342 페이지의 『헤더 처리 프로그램 인터페이스』에서 설명됩니다.

기타 컨테이너는 헤더 프로그램이 호출되는 환경에 대한 정보를 보유합니다. 예를 들어, 다음과 같습니다.

- 헤더 프로그램이 호출된 트랜잭션 ID
- 프로그램이 서비스 제공자 또는 요청자 파이프라인에 대해 호출되는지 여부
- 처리되는 메시지가 요청 또는 응답인지 여부

헤더 처리 프로그램은 일반적으로 트랜잭션 CPIH에서 실행되고, 이는 TASKDATALOC(ANY) 속성으로 정의됩니다. 따라서 프로그램을 링크 편집할 때에는 AMODE(31) 옵션을 지정해야 합니다.

헤더 처리 프로그램이 SOAP 요청을 위해 호출되는 방법

파이프라인 구성의 <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 및 <cics_soap_1.2_handler_java> 요소에는 0, 1 또는 그 이상의 <headerprogram> 요소가 포함되고, 각각은 다음 하위를 포함합니다.

```
<program_name>
  <namespace>
    <localname>
      <mandatory>
```

파이프라인이 인바운드 SOAP 메시지를 처리 중일 때(서비스 제공자의 경우 요청, 서비스 요청자의 경우 응답), <program_name> 요소에 지정되는 헤더 프로그램은 다음 항목에 따라 호출되거나 호출되지 않습니다.

- <namespace>, <localname> 및 <mandatory> 요소의 콘텐츠
- SOAP 헤더 자체의 루트 요소의 특정 속성 값(SOAP 1.1의 경우 **actor** 속성, SOAP 1.2의 경우 **role** 속성)

다음 규칙은 헤더 프로그램이 제공된 케이스로 호출되는지 여부를 판별합니다.

파이프라인 구성 파일의 <mandatory> 요소

요소에 true(또는 1)가 포함되는 경우, 남은 규칙이 처리를 위해 선택하는 SOAP 메시지에 헤더가 없더라도 최소 한 번 헤더 처리 프로그램이 호출됩니다.

- 선택되는 헤더 블록이 없는 경우, 헤더 처리 프로그램이 한 번 호출됩니다.
- 헤더 블록이 남아 있는 규칙에 의해 선택되는 경우, 헤더 처리 프로그램은 선택된 각 헤더에 대해 한 번 호출됩니다.

SOAP 헤더 블록의 속성

SOAP 1.1의 경우, 헤더 블록은 **actor** 속성이 없거나

`http://schemas.xmlsoap.org/soap/actor/next`의 값을 갖는 경우에만 처리될 수 있습니다.

SOAP 1.2의 경우, 헤더 블록은 **role** 속성이 없거나 다음 값 중 하나를 갖는 경우에만 처리될 수 있습니다.

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

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

처리될 수 있는 헤더 블록은 다음 규칙에 의해 선택되지 않으면 처리되지 않습니다.

파이프라인 구성 파일의 <namespace> 및 <localname> 요소

이전 규칙에 따라 처리할 수 있는 헤더 블록은 다음 조건이 충족되는 경우에만 처리되도록 선택됩니다.

- 헤더 블록의 루트 요소 이름은 파이프라인 구성 파일의 <localname> 요소와 일치합니다
- 루트 요소의 네임스페이스는 파이프라인 구성 파일의 <namespace> 요소와 일치합니다

예를 들어, 이 헤더 블록을 고려하십시오.

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

기타 규칙을 따르면 다음 행이 파이프라인 구성 파일에서 코드화될 때 헤더 블록이 처리되도록 선택됩니다.

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

<localname> 요소에는 네임스페이스의 모든 헤더 블록이 처리되어야 한다는 것을 표시하기 위한 *가 포함될 수 있습니다. 따라서, 동일한 헤더 블록이 다음 코드에 의해 선택됩니다.

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

SOAP 메시지에 둘 이상의 헤더가 있는 경우 헤더 처리 프로그램은 헤더 일치가 발생할 때마다 한 번씩 호출되지만 헤더가 처리되는 순서는 정의되지 않습니다.

CICS 제공 SOAP 메시지 핸들러는 메시지 핸들러가 수신할 때 SOAP 메시지에 존재하는 헤더 블록을 기반으로 호출되는 헤더 처리 프로그램을 선택합니다. 따라서, 헤더 처리 프로그램은 동일한 SOAP 메시지 핸들러의 메시지에 추가되는 헤더 블록의 결과로 절대 호출되지 않습니다. 사용자 파이프라인에서 새 헤더(또는 수정된 헤더)를 처리하려는 경우, 사용자 파이프라인에 다른 SOAP 메시지 핸들러를 정의해야 합니다.

아웃바운드 메시지(서비스 요청자의 요청, 서비스 제공자의 응답)의 경우, CICS 제공 SOAP 메시지 핸들러는 헤더를 포함하지 않는 SOAP 메시지를 작성합니다. 하나 이상의 헤더를 메시지에 추가하려면 헤더를 추가하기 위한 헤더 핸들러 프로그램을 작성해야 합니다. 이 헤더 핸들러가 호출된다는 것을 확인하려면, 이를 사용자 파이프라인 구성 파일에서 정의해야 하고 `<mandatory>true</mandatory>`를 지정해야 합니다.

헤더 핸들러가 파이프라인의 요청 단계에서 호출되면, 응답 단계에서 플로우하는 메시지에 일치하는 헤더가 포함되지 않을 지라도 응답 단계에서 다시 호출됩니다.

헤더 처리 프로그램 인터페이스

CICS 제공 SOAP 1.1 및 SOAP 1.2 메시지 핸들러는 채널 DFHHHC-V1을 사용하여 헤더 처리 프로그램에 링크됩니다. 채널에서 전달되는 컨테이너에는 헤더 처리 프로그램 인터페이스에 고유한 여러 컨테이너와 파이프라인의 모든 헤더 처리 프로그램과 메시지 핸들러 프로그램이 액세스할 수 있는 컨텍스트 컨테이너 및 사용자 컨테이너의 세트가 포함됩니다.

컨테이너 DFHHEADER는 헤더 처리 프로그램 인터페이스에 고유합니다. 기타 컨테이너는 파이프라인 내 다른 위치에서 사용할 수 있지만 헤더 처리 프로그램에서 특정 용도를 갖습니다. 이 카테고리의 컨테이너는 DFHWS-XMLNS, DFHWS-BODY, DFHXMLSS-PARSE입니다.

참고: Axis2를 사용하여 SOAP 메시지를 처리하는 웹 서비스는 헤더 처리 프로그램 인터페이스를 사용할 수 있지만 고유한 Axis2 핸들러를 Java로 작성하여 SOAP 헤더를 처리하는 것이 보다 효율적입니다. Axis2 핸들러 작성에 대한 자세한 정보는 Axis2 모듈 직접 작성을 참조하십시오.

컨테이너 DFHHEADER

헤더 처리 프로그램이 호출되면 헤더 처리 프로그램을 구동시킨 단일 헤더 블록이 DFHHEADER에 포함됩니다. 헤더 프로그램이 파이프라인 구성 파일에서 `<mandatory>true</mandatory>` 또는 `<mandatory>1</mandatory>`로 지정되면 SOAP 메시지에 일치하는 헤더 블록이 없더라도 해당 프로그램이 호출됩니다. 이 경우 컨테이

너 DFHHEADER의 길이는 0입니다. 이 경우 헤더 블록이 없는 SOAP 메시지에 헤더 블록을 추가하기 위해 헤더 처리 프로그램이 호출됩니다.

CICS가 작성하는 SOAP 메시지에는 처음에 헤더가 없습니다. 메시지에 헤더를 추가하려면 `<mandatory>true</mandatory>` 또는 `<mandatory>1</mandatory>`를 지정하여 최소 하나의 헤더 처리 프로그램이 호출되도록 해야 합니다.

헤더 프로그램이 리턴되는 경우 컨테이너 DFHHEADER는 CICS가 원래 헤더 블록 대신 SOAP 메시지에 삽입하는 0개 이상의 헤더 블록을 포함해야 합니다.

- 원래 헤더 블록을 변경하지 않고 리턴할 수 있습니다.
- 헤더 블록의 콘텐츠를 수정할 수 있습니다.
- 원래 블록에 하나 이상의 새 헤더 블록을 추가할 수 있습니다.
- 원래 헤더 블록을 하나 이상의 다른 블록으로 바꿀 수 있습니다.
- 헤더 블록은 완전히 삭제할 수 있습니다.

컨테이너 DFHWS-XMLNS

헤더 처리 프로그램이 호출되면 SOAP 엔벨로프에서 선언된 XML 네임스페이스에 대한 정보가 DFHWS-XMLNS에 포함됩니다. 헤더 프로그램은 이 정보를 사용하여 다음 태스크를 수행할 수 있습니다.

- 헤더 블록에 나타나는 규정된 이름을 해석합니다.
- 새로운 또는 수정된 헤더 블록에서 규정된 이름을 구성합니다.

네임스페이스 정보는 네임스페이스 선언을 위해 표준 XML 표기법을 사용하는 네임스페이스 선언의 목록으로 구성됩니다. DFHWS-XMLNS의 네임스페이스 선언은 공백으로 구분됩니다. 예를 들어, 다음과 같습니다.

```
xmlns:na='http://abc.example.org/schema' xmlns:nx='http://xyz.example.org/schema'
```

DFHWS-XMLNS 콘텐츠에 추가하는 방식으로 SOAP 엔벨로프에 네임스페이스 선언을 더 추가할 수 있습니다. 그러나 범위가 SOAP 헤더 블록 또는 SOAP 본문인 네임스페이스가 헤더 블록 또는 본문에서 각각 가장 올바르게 선언됩니다. 프로그램에 표시되지 않는 XML 요소가 종속될 수 있으므로 헤더 처리 프로그램의 컨테이너 DFHWS-XMLNS에서 네임스페이스 선언을 삭제하지 않는 것이 좋습니다.

컨테이너 DFHWS-BODY

이 컨테이너는 SOAP 엔벨로프의 본문 섹션을 포함합니다. 헤더 처리 프로그램이 콘텐츠를 수정할 수 있습니다.

헤더 처리 프로그램이 호출되면 DFHWS-BODY에 SOAP `<Body>` 요소가 포함됩니다.

헤더 프로그램이 리턴되는 경우 컨테이너 DFHWS-BODY는 다시 CICS가 원래 본문 대신 SOAP 메시지에 삽입하는 올바른 SOAP <Body>를 포함해야 합니다.

- 원래 본문을 변경하지 않고 리턴할 수 있습니다.
- 본문의 콘텐츠를 수정할 수 있습니다.

SOAP 본문을 완전히 삭제해서는 안됩니다. 모든 SOAP 메시지에 <Body> 요소가 포함되어야 하기 때문입니다.

컨테이너 DFHXMLSS-PARSE

파이프라인 구성에서 <cics_soap_1.1_handler> 또는 <cics_soap_1.2_handler> 요소를 사용하고 헤더 프로그램이 호출되면 해당 헤더에 대한 XMLSS(XML System Services) 레코드가 DFHXMLSS-PARSE에 포함됩니다. 이 컨테이너는 <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소가 사용되는 경우에는 작성되지 않습니다.

제어, 컨텍스트 및 사용자 컨테이너

설명한 컨테이너뿐 아니라 인터페이스는 제어 컨테이너, 컨텍스트 컨테이너, 사용자 컨테이너를 채널 DFHHHC-V1에서 전달합니다.

이 컨테이너에 대한 자세한 정보는 345 페이지의 『파이프라인에서 사용되는 컨테이너』를 참조하십시오.

터미널 핸들러에서 인바운드 요청의 동적 라우팅

서비스 제공자 파이프라인의 터미널 핸들러가 CICS 제공 SOAP 메시지 핸들러 중 하나인 경우, 컨테이너 DFHWS-APPHANDLER에 지정된 대상 애플리케이션 핸들러 프로그램이 동적 라우팅에 적합한 경우도 있습니다. 애플리케이션 핸들러 프로그램 이전의 모든 파이프라인 처리는 항상 SOAP 메시지를 수신한 CICS 리전에서 로컬로 수행됩니다.

대상 애플리케이션 핸들러 프로그램을 실행하는 트랜잭션은 다음 조건 중 하나에 해당되는 경우 라우팅에 적합합니다.

- 파이프라인이 메시지를 처리하는 트랜잭션은 DYNAMIC 또는 REMOTE로 정의됩니다. 이 트랜잭션은 인바운드 SOAP 메시지에서 URI를 맵핑하는 데 사용되는 URIMAP에서 정의됩니다.
- 파이프라인의 프로그램이 컨테이너 DFHWS-USERID의 콘텐츠를 초기값에서 변경했습니다.
- 파이프라인의 프로그램이 컨테이너 DFHWS-TRANID의 콘텐츠를 초기값에서 변경했습니다.
- WS-AT SOAP 헤더가 인바운드 SOAP 메시지에 존재합니다.

이전 시나리오 모두의 경우 파이프라인 처리 중에 태스크가 전환됩니다. 두 번째 태스크는 DFHWS-TRANID 컨테이너에서 지정된 트랜잭션에서 실행됩니다. 이 스위치는 동적 라우팅을 수행할 기회를 제공하지만 이 경우 MRO를 사용하여 CICS 리전을 함께 연결해야 합니다. 또한 라우팅하는 CICS 리전이 채널과 컨테이너를 지원해야 합니다.

라우팅은 DFHWS-TRANID에서 이름이 지정된 트랜잭션에 대한 TRANSACTION 정의가 다음 속성 세트 중 하나를 지정하는 경우에만 발생합니다.

DYNAMIC(YES)

트랜잭션이 분산 라우팅 모델을 사용하여 라우팅됩니다. 이 모델에서는 라우팅 프로그램이 **DSRTPGM** 시스템 초기화 매개변수에 지정됩니다.

DYNAMIC(NO) REMOTESYSTEM(sysid)

트랜잭션이 *sysid*로 식별된 시스템으로 라우팅됩니다.

웹 서비스 요청 라우팅에 대한 자세한 정보는 기술 노트, *제공자 모드 CICS 웹 서비스 라우팅*을 참조하십시오.

CICS 웹 서비스 지원과 함께 배치된 애플리케이션의 경우에는 요청을 동적으로 라우팅할 수 있는 두 번째 기회가 있습니다. 이 시점에서 CICS가 사용자 프로그램에 링크됩니다. 그런 다음 동적 라우팅 모델을 사용하여 요청이 라우팅됩니다. 이 모델에서는 라우팅 프로그램이 **DTRPGM** 시스템 초기화 매개변수에 지정됩니다. 이 경우 프로그램 특성으로 라우팅 적합성이 결정됩니다. 프로그램에 링크할 때 채널과 컨테이너를 사용하는 경우에는 V3.1 이상의 CICS 리전으로만 요청을 동적으로 라우팅할 수 있습니다. COMMAREA를 사용하는 경우에는 이 제한사항이 적용되지 않습니다.

"테이지 체인"은 지원되지 않습니다. 즉, 일단 요청이 대상 리전에 동적으로 라우팅되면 트랜잭션이 ROUTABLE(YES) 및 DYNAMIC(YES)로 정의되더라도 대상에서 세 번째 리전으로 동적으로 라우팅될 수 없습니다. 그러나 대상 리전에서 세 번째 리전으로 정적으로 트랜잭션이 라우팅될 수는 있습니다.

자세한 정보는 *CICS 사용자 정의 안내서*를 참조하십시오.

파이프라인에서 사용되는 컨테이너

파이프라인은 일반적으로 여러 메시지 핸들러 프로그램으로 구성되며 CICS 제공 SOAP 메시지 핸들러를 사용하는 경우 여러 헤더 처리 프로그램으로 구성됩니다. CICS는 컨테이너를 사용하여 이러한 프로그램과 정보를 주고 받습니다. 프로그램은 또한 컨테이너를 사용하여 파이프라인의 다른 프로그램과 통신합니다.

CICS 파이프라인은 컨테이너가 여러 개인 채널을 사용하여 메시지 핸들러와 헤더 처리 프로그램에 링크됩니다. 선택적인 컨테이너도 있고 모든 메시지 핸들러에서 필요한 컨테이너도 있습니다. 일부 메시지 핸들러는 사용하고 다른 메시지 핸들러는 사용하지 않는 컨테이너도 있습니다.

핸들러가 호출되기 전에 핸들러가 작업을 수행하기 위해 사용할 수 있는 정보로 일부 또는 모든 컨테이너를 채웁니다. 핸들러가 리턴하는 컨테이너가 후속 처리를 결정하며 파이프라인에서 다음 핸들러로 전달됩니다.

컨테이너는 다음과 같이 분류할 수 있습니다.

제어 컨테이너

이 컨테이너는 파이프라인 조작에 필수적입니다. 핸들러는 제어 컨테이너를 사용하여 핸들러가 처리되는 순서를 수정할 수 있습니다. 제어 컨테이너의 이름은 CICS로 정의되고 문자 DFH로 시작됩니다.

컨텍스트 컨테이너

이 컨테이너는 핸들러가 호출되는 환경에 대한 정보를 포함합니다. CICS는 첫 번째 메시지 핸들러를 호출하기 전에 이 컨테이너에 정보를 저장하지만 경우에 따라 핸들러가 콘텐츠를 변경하거나 컨테이너를 삭제할 수도 있습니다. 컨텍스트 컨테이너 변경사항은 핸들러가 호출되는 순서에 직접적으로 영향을 주지 않습니다. 컨텍스트 컨테이너의 이름은 CICS로 정의되고 문자 DFH로 시작됩니다.

헤더 처리 프로그램 컨테이너

이 컨테이너는 CICS 제공 SOAP 메시지 핸들러에서 호출되는 헤더 처리 프로그램이 사용하는 정보를 포함합니다. 이 컨테이너에 대한 정보는 헤더 처리 프로그램 인터페이스를 참조하십시오.

보안 컨테이너

이 컨테이너는 신뢰 클라이언트 인터페이스와 보안 메시지 핸들러가 보안 토큰 서비스(STS)를 사용하여 보안 토큰을 처리하기 위해 사용하는 정보를 포함합니다. 보안 컨테이너의 이름은 CICS로 정의되고 문자 DFH로 시작됩니다.

생성된 컨테이너

이 컨테이너는 처리하기 위해 애플리케이션 프로그램과 주고 받는 가변 배열 및 긴 문자열과 같은 SOAP 메시지의 데이터를 포함합니다. CICS는 파이프라인 처리 중에 이 컨테이너를 자동으로 작성하며 이름은 문자 DFH로 시작됩니다.

사용자 컨테이너

이 컨테이너는 특정 메시지 핸들러가 다른 메시지 핸들러에 전달해야 하는 정보를 포함합니다. 사용자 컨테이너 사용은 전적으로 메시지 핸들러의 문제입니다. 이 컨테이너의 고유한 이름을 선택할 수 있지만 DFH로 시작하는 이름을 사용해서는 안 됩니다.

제어 컨테이너

제어 컨테이너는 파이프라인 조작에 필수적입니다. 핸들러는 제어 컨테이너를 사용하여 핸들러가 처리되는 순서를 수정할 수 있습니다.

DFHERROR 컨테이너

DFHERROR은 파이프라인 오류에 대한 정보를 다른 메시지 핸들러로 전달하는 데 사용되는 DATATYPE(BIT) 컨테이너입니다.

표 6. DFHERROR 컨테이너의 구조

필드 이름	길이(바이트)	컨텐츠
PIISNEB-MAJOR-VERSION	1	『1』
PIISNEB-MINOR-VERSION	1	『1』
PIISNEB-ERROR-TYPE	1	오류 유형을 나타내는 숫자 값. 이 값은 표 7에 설명되어 있습니다.
PIISNEB-ERROR-MODE	1	P 제공자 파이프라인에서 발생한 오류 R 요청자 파이프라인에서 발생한 오류 T 신뢰 클라이언트에서 발생한 오류
PIISNEB-ABCODE	4	오류가 트랜잭션 이상 종료와 연관되는 경우 이상 종료 코드.
PIISNEB-ERROR-CONTAINER1	16	오류가 컨테이너와 연관되는 경우 컨테이너의 이름.
PIISNEB-ERROR-CONTAINER2	16	오류가 여러 컨테이너와 연관되는 경우 두 번째 컨테이너의 이름.
PIISNEB-ERROR-NODE	8	오류가 발생한 핸들러 프로그램의 이름.

표 7. PIISNEB-ERROR-TYPE 필드의 값

PIISNEB-ERROR-TYPE 값	의미
1	핸들러 프로그램이 실패했습니다. 이상 종료 코드는 PIISNEB-ABCODE 필드에 있습니다.
2	핸들러에 필요한 컨테이너가 비어 있습니다. 컨테이너 이름은 PIISNEB-ERROR-CONTAINER1 필드에 있습니다.
3	핸들러에 필요한 컨테이너가 누락되었습니다. 컨테이너 이름은 PIISNEB-ERROR-CONTAINER1 필드에 있습니다.
4	컨테이너가 하나만 예상될 때 두 개 컨테이너가 핸들러로 전달되었습니다. 컨테이너의 이름은 PIISNEB-ERROR-CONTAINER1 및 PIISNEB-ERROR-CONTAINER2 필드에 있습니다.
5	대상 프로그램에 대한 링크 시도가 실패했습니다. 대상 프로그램이 실패한 경우 이상 종료 코드는 PIISNEB-ABCODE 컨테이너에 있습니다.
6	기본 전송 오류로 인해 파이프라인 관리자가 원격 서버와 통신하는 데 실패했습니다.

표 7. PIISNEB-ERROR-TYPE 필드의 값 (계속)

PIISNEB-ERROR-TYPE 값	의미
7	DFHWS-STSACTION 컨테이너에 오류가 있습니다. 컨테이너가 누락되었거나 손상되었거나 잘못된 값을 포함합니다.
8	DFHPIRT가 파이프라인을 시작하는 데 실패했습니다.
9	DFHPIRT가 컨테이너에 메시지를 저장하는 데 실패했습니다.
10	DFHPIRT가 컨테이너에서 메시지를 가져오는 데 실패했습니다.
11	처리되지 않은 오류가 발생했습니다.

컨테이너 구조의 COBOL 선언은 다음과 같습니다.

```
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).
```

다음 표에는 컨테이너를 맵핑하는 언어 카피북이 나와 있습니다.

표 8. 컨테이너를 맵핑하는 카피북

언어	카피북
COBOL	DFHPIUCO
PL/I	DFHPIUCL
C 및 C++	dfhpiuch.h
어셈블러	DFHPIUCD

DFHFUNCTION 컨테이너

DFHFUNCTION은 파이프라인에서 프로그램이 호출되는 위치를 나타내는 16자 문자열을 포함하는 DATATYPE(CHAR) 컨테이너입니다.

문자열은 다음 중 한 가지 값을 갖습니다. 맨 오른쪽 문자 위치는 공백 문자로 채워집니다.

RECEIVE-REQUEST

핸들러는 서비스 제공자 파이프라인의 비터미널 핸들러이며 인바운드 요청 메시지를 처리하기 위해 호출됩니다. 핸들러에 입력하는 경우, 메시지는 제어 컨테이너 DFHREQUEST에 있습니다.

SEND-RESPONSE

핸들러는 서비스 제공자 파이프라인의 비터미널 핸들러이며 아웃바운드 응답 메시지를 처리하기 위해 호출됩니다. 핸들러에 입력하는 경우, 메시지는 제어 컨테이너 DFHRESPONSE에 있습니다.

SEND-REQUEST

요청을 보내는 파이프라인이 즉, 아웃바운드 메시지를 처리하는 서비스 요청자에서 핸들러를 호출합니다.

RECEIVE-RESPONSE

응답을 수신하는 파이프라인이 즉, 인바운드 메시지를 처리하는 서비스 요청자에서 핸들러를 호출합니다.

PROCESS-REQUEST

핸들러가 서비스 제공자 파이프라인의 터미널 핸들러로 호출됩니다.

NO-RESPONSE

응답을 처리하지 않는 경우 요청을 처리한 후 핸들러가 호출됩니다.

HANDLER-ERROR

오류가 발견되었으므로 핸들러가 호출됩니다.

요청을 처리하고 응답을 리턴하는 서비스 제공자 파이프라인의 경우 발생하는 DFHFUNCTION 값은 RECEIVE-REQUEST, PROCESS-REQUEST, SEND-RESPONSE입니다. 그림 27은 핸들러가 호출되는 순서와 각 핸들러로 전달되는 DFHFUNCTION 값을 보여줍니다.

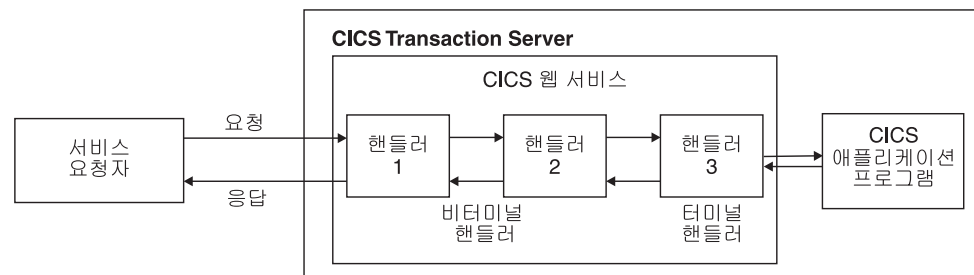


그림 27. 서비스 제공자 파이프라인의 핸들러 시퀀스

시퀀스	핸들러	DFHFUNCTION
1	핸들러 1	RECEIVE-REQUEST
2	핸들러 2	RECEIVE-REQUEST
3	핸들러 3	PROCESS-REQUEST
4	핸들러 2	SEND-RESPONSE
5	핸들러 1	SEND-RESPONSE

요청을 보내고 응답을 수신하는 서비스 요청자 파이프라인의 경우 발생하는 DFHFUNCTION 값은 SEND-REQUEST 및 RECEIVE-RESPONSE입니다. 그림 28은 핸들러가 호출되는 시퀀스와 각 핸들러로 전달되는 DFHFUNCTION 값을 보여줍니다.

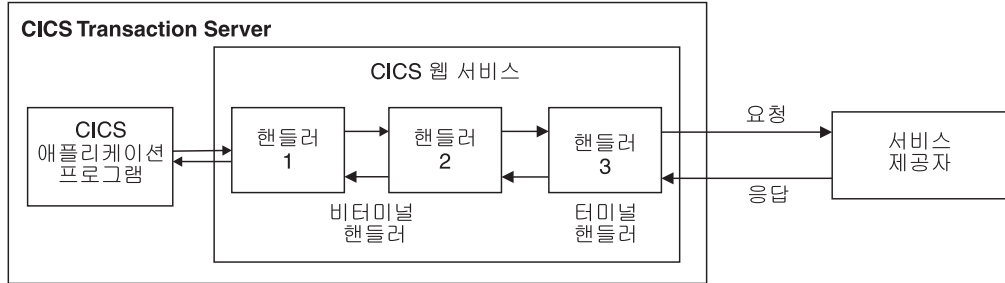


그림 28. 서비스 요청자 파이프라인의 핸들러 시퀀스

시퀀스	핸들러	DFHFUNCTION
1	핸들러 1	SEND-REQUEST
2	핸들러 2	SEND-REQUEST
3	핸들러 3	SEND-REQUEST
4	핸들러 3	RECEIVE-RESPONSE
5	핸들러 2	RECEIVE-RESPONSE
6	핸들러 1	RECEIVE-RESPONSE

지정된 메시지 핸들러에서 나타날 수 있는 DFHFUNCTION 값은 파이프라인이 제공자 또는 요청자인지 여부, 파이프라인이 요청 또는 응답 단계(Phase)에 있는지 여부, 핸들러가 터미널 핸들러 또는 비터미널 핸들러인지 여부에 따라 다릅니다. 다음 표에는 각 값이 발생할 수 있는 경우가 요약되어 있습니다.

DFHFUNCTION 값	제공자 또는 요청자 파이프라인	파이프라인 단계(Phase)	터미널 또는 비터미널 핸들러 라인
RECEIVE-REQUEST	제공자	요청 단계(Phase)	비터미널
SEND-RESPONSE	제공자	응답 단계(Phase)	비터미널
SEND-REQUEST	요청자	요청 단계(Phase)	비터미널
RECEIVE-RESPONSE	요청자	응답 단계(Phase)	비터미널
PROCESS-REQUEST	제공자	요청 단계(Phase)	터미널
NO-RESPONSE	둘 다	응답 단계(Phase)	비터미널
HANDLER-ERROR	둘 다	둘 다	둘 다

DFHHTTPMETHOD 컨테이너

모든 HTTP 제공자 모드 CICS 파이프라인에서 애플리케이션 프로그램이 사용할 수 있는 DATATYPE(CHAR) 컨테이너입니다.

이 컨테이너의 길이는 8자이며 수신 요청에서 사용된 HTTP 메소드의 이름을 포함합니다. 요청이 HTTP를 통해 도달하지 않은 경우에는 채워지지 않습니다.

DFHHTTPSTATUS 컨테이너

DFHHTTPSTATUS는 서비스 제공자 파이프라인의 응답 단계에서 생성되는 메시지에 대한 HTTP 상태 코드와 상태 텍스트를 지정하는 데 사용되는 DATATYPE(CHAR) 컨테이너입니다.

DFHHTTPSTATUS 컨테이너의 콘텐츠는 다음 구조를 갖는 HTTP 응답 메시지의 초기 상태 표시줄과 같아야 합니다.

```
HTTP/1.1 nnn tttttttt
```

HTTP/1.1

HTTP의 버전과 릴리스.

nnn 리턴할 3자리 10진수 HTTP 상태 코드.

tttttttt

상태 코드 nnn과 연관되고 사용자가 읽을 수 있는 상태 텍스트.

다음 문자열은 콘텐츠 예입니다.

```
HTTP/1.1 412 Precondition Failed
```

파이프라인이 WebSphere MQ 전송을 사용하는 경우에는 DFHHTTPSTATUS 컨테이너가 무시됩니다.

DFHMEDIATYPE 컨테이너

DFHMEDIATYPE은 서비스 제공자 파이프라인의 응답 단계에서 생성되는 메시지에 대한 매체 유형을 지정하는 데 사용되는 DATATYPE(CHAR) 컨테이너입니다.

DFHMEDIATYPE 컨테이너의 콘텐츠는 슬래시 문자로 구분되는 유형과 하위 유형으로 구성되어야 합니다. 다음 문자열은 DFHMEDIATYPE 컨테이너의 올바른 콘텐츠에 대한 두 가지 예를 보여줍니다.

```
text/plain
```

```
image/svg+xml
```

파이프라인이 WebSphere MQ 전송을 사용하는 경우에는 DFHMEDIATYP 컨테이너가 무시됩니다.

DFHNORESPONSE 컨테이너

DFHNORESPONSE는 서비스 요청자 파이프라인의 요청자 단계(Phase)에서 서비스 제공자가 응답을 리턴할 것으로 예상되지 않음을 나타내는 DATATYPE(CHAR) 컨테이너입니다.

DFHNORESPONSE 컨테이너의 콘텐츠는 정의되지 않습니다. 서비스 제공자가 응답을 리턴할 수 있는지 여부를 알아야 하는 메시지 핸들러는 컨테이너가 있는지 여부만 판별합니다.

- 컨테이너 DFHNORESPONSE가 있으면 응답이 예상되지 않습니다.
- 컨테이너 DFHNORESPONSE가 없으면 응답이 예상됩니다.

이 정보는 처음에 서비스 제공자와 함께 사용되는 프로토콜에 따라 서비스 요청자 애플리케이션이 제공합니다. 따라서 메시지 핸들러에서 이 컨테이너를 삭제하거나 메시지 핸들러가 없는 경우 메시지 핸들러를 작성해서는 안됩니다. 엔드포인트간 프로토콜을 방해할 수 있기 때문입니다.

서비스 요청자 파이프라인의 요청 단계(Phase) 이외에는 이 컨테이너 사용이 정의되지 않습니다.

DFHREQUEST 컨테이너

DFHREQUEST는 파이프라인의 요청 단계(Phase)에서 처리되는 요청 메시지를 포함하는 DATATYPE(CHAR) 컨테이너입니다.

CICS 제공 SOAP 메시지 핸들러 중 하나가 파이프라인에서 구성되는 경우, 컨테이너 DFHREQUEST는 SOAP 엔벨로프에 SOAP 메시지 헤더를 포함하도록 갱신됩니다. 메시지가 CICS 제공 SOAP 메시지 핸들러로 구성되고 이후에 변경되지 않은 경우, DFHREQUEST에 전체 SOAP 엔벨로프가 포함되며 해당 콘텐츠는 모두 UTF-8 코드 페이지에 저장됩니다.

DFHREQUEST 컨테이너는 메시지 핸들러가 호출될 때 요청에 존재하며 DFHFUNCTION 컨테이너에는 RECEIVE-REQUEST 또는 SEND-REQUEST가 포함됩니다.

이 경우, 일반 프로토콜은 동일한 콘텐츠 또는 수정된 콘텐츠와 함께 파이프라인으로 DFHREQUEST를 리턴하는 것입니다. 파이프라인의 다음 메시지 핸들러 프로그램(있는 경우)과 함께 파이프라인 요청 단계가 정상적으로 계속 처리됩니다.

또는 메시지 핸들러가 컨테이너 DFHREQUEST를 삭제하고 DFHRESPONSE 컨테이너에 응답을 배치할 수 있습니다. 이러한 방식으로 일반적인 처리 순서가 역전되며 파이프라인의 응답 단계(Phase)와 함께 계속 처리됩니다.

DFHRESPONSE 컨테이너

DFHRESPONSE는 파이프라인의 응답 단계(Phase)에서 처리되는 응답 메시지를 포함하는 DATATYPE(CHAR) 컨테이너입니다. 메시지가 CICS 제공 SOAP 메시지 핸들러로 구성되고 이후에 변경되지 않은 경우, DFHRESPONSE에는 전체 SOAP 엔벨로프와 UTF-8 코드 페이지의 모든 해당 콘텐츠가 포함됩니다.

메시지 핸들러가 호출될 때 DFHRESPONSE 컨테이너가 있고 DFHFUNCTION 컨테이너에는 SEND-RESPONSE 또는 RECEIVE-RESPONSE가 포함됩니다.

이 경우, 일반 프로토콜은 동일한 콘텐츠 또는 수정된 콘텐츠와 함께 파이프라인으로 DFHRESPONSE를 리턴하는 것입니다. 파이프라인의 다음 메시지 핸들러 프로그램(있는 경우)과 함께 파이프라인이 정상적으로 계속 처리됩니다.

DFHRESPONSE 컨테이너는 DFHFUNCTION에 RECEIVE-REQUEST, SEND-REQUEST, PROCESS-REQUEST 또는 HANDLER-ERROR가 포함되는 경우에도 존재하며 길이는 0입니다.

DFHWS-CCSID 컨테이너

DFHWS-CCSID는 응답 컨테이너에서 데이터의 CCSID를 지정하는 실어(4바이트)를 포함하는 DATATYPE(BIT) 컨테이너입니다.

이 컨테이너는 언어 구조를 XML로 변환하기 위해 CICS 코드를 사용하는 제공자 모드 파이프라인에만 유효합니다.

CCSID는 WSBIND 파일을 생성하는 데 사용되는 CCSID와 호환 가능해야 합니다. 그렇지 않은 경우, 생성된 SOAP 응답에 잘못되거나 올바르지 않은 문자가 포함될 수 있습니다.

CCSID는 930, 1390, 5026, 1026과 교환될 수 없습니다. 또한 CICS는 CCSID가 클라이언트 CCSID로 사용할 수 있는 CCSID로 변경될 수 있도록 허용하지 않습니다.

DFHWS-CCSID 컨테이너에서 값을 처리하는 데 문제점이 있으면 WSBIND 파일의 CCSID를 사용하여 계속 처리합니다.

DFHWS-CCSID 컨테이너는 채널 구동 애플리케이션 프로그램에서 리턴되는 경우에만 확인합니다.

컨테이너의 파이프라인 프로토콜 제어 방식

DFHFUNCTION, DFHREQUEST, DFHRESPONSE 컨테이너의 콘텐츠가 함께 파이프라인 프로토콜을 제어합니다.

파이프라인 실행의 두 단계(Phase)(요청 단계(Phase) 및 응답 단계(Phase))에서 DFHFUNCTION 값이 각 메시지 핸들러로 전달되는 제어 컨테이너를 결정합니다.

DFHFUNCTION	컨텍스트	DFHREQUEST	DFHRESPONSE
RECEIVE-REQUEST	서비스 제공자; 요청 단계(Phase)	존재(길이 > 0)	존재(길이 = 0)
SEND-RESPONSE	서비스 제공자; 응답 단계(Phase)	부재	존재(길이 > 0)
SEND-REQUEST	서비스 요청자; 요청 단계(Phase)	존재(길이 > 0)	존재(길이 = 0)
RECEIVE-RESPONSE	서비스 요청자; 응답 단계(Phase)	부재	존재(길이 > 0)

DFHFUNCTION	컨텍스트	DFHREQUEST	DFHRESPONSE
PROCESS-REQUEST	서비스 제공자; 터미널 핸들러	존재(길이 > 0)	존재(길이 = 0)
HANDLER-ERROR	서비스 요청자 또는 제 공자; 두 단계(Phase) 중 하나	부재	존재(길이 = 0)
NO-RESPONSE	서비스 요청자 또는 제 공자; 응답 단계 (Phase)	부재	부재

후속 처리는 메시지 핸들러가 파이프라인으로 다시 전달하는 컨테이너로 결정됩니다.

요청 단계(Phase)

- 메시지 핸들러가 DFHREQUEST 컨테이너를 리턴할 수 있습니다. 다음 핸들러로 요청 단계(Phase)에서 계속 처리됩니다. 컨테이너의 데이터 길이는 0이어야 합니다.
- 메시지 핸들러가 DFHRESPONSE 컨테이너를 리턴할 수 있습니다. 응답 단계(Phase)로 처리가 전환되고 DFHFUNCTION가 SEND-RESPONSE(서비스 제공자의 경우) 및 RECEIVE-RESPONSE(서비스 요청자의 경우)로 설정되어 동일한 핸들러가 호출됩니다. 컨테이너의 데이터 길이는 0이어야 합니다.
- 메시지 핸들러가 컨테이너를 리턴할 수 없습니다. 처리는 응답 단계(Phase)로 전환되며 DFHFUNCTION이 NO-RESPONSE로 설정되어 동일한 핸들러가 호출됩니다.

터미널 핸들러(서비스 제공자만 해당)

- 메시지 핸들러가 DFHRESPONSE 컨테이너를 리턴할 수 있습니다. 처리가 응답 단계(Phase)로 전환되며 이전 핸들러가 DFHFUNCTION의 새 값(SEND-RESPONSE)으로 호출됩니다. 컨테이너의 데이터 길이는 0이어야 합니다.
- 메시지 핸들러가 컨테이너를 리턴할 수 없습니다. 처리가 응답 단계(Phase)로 전환되며 이전 핸들러가 DFHFUNCTION의 새 값(NO-RESPONSE)으로 호출됩니다.

응답 단계(Phase)

- 메시지 핸들러가 DFHRESPONSE 컨테이너를 리턴할 수 있습니다. 응답 단계(Phase)에서 처리가 계속되고 다음 핸들러가 호출됩니다. 컨테이너의 데이터 길이는 0이어야 합니다.
- 메시지 핸들러가 컨테이너를 리턴할 수 없습니다. 처리가 응답 단계(Phase)에서 계속되며 시퀀스 내 다음 핸들러가 DFHFUNCTION의 새 값(NO-RESPONSE)으로 호출됩니다.

중요사항: 요청 단계(Phase)에서는 메시지 핸들러가 DFHREQUEST 또는 DFHRESPONSE를 리턴할 수 있지만 둘 다 리턴할 수는 없습니다. 메시지 핸들러가 호출될 때 두 컨테이너가 모두 있어야 하므로 그 중 하나를 삭제해야 합니다.

이 표는 DFHFUNCTION의 모든 값에 대해 파이프라인이 수행하는 조치와 각 메시지 핸들러가 리턴하는 DFHREQUEST 및 DFHRESPONSE의 모든 조합을 보여줍니다.

DFHFUNCTION	컨텍스트	DFHREQUEST	DFHRESPONSE	조치
RECEIVE-REQUEST	서비스 제공자; 요청 단계(Phase)	존재(길이 > 0)	존재	(오류)
RECEIVE-REQUEST	서비스 제공자; 요청 단계(Phase)	존재(길이 > 0)	부재	RECEIVE-REQUEST 기능으로 다음 핸들러 호출
RECEIVE-REQUEST	서비스 제공자; 요청 단계(Phase)	존재(길이 = 0)	적용되지 않음	(오류)
RECEIVE-REQUEST	서비스 제공자; 요청 단계(Phase)	부재	존재(길이 > 0)	응답 단계(Phase)로 전환 및 SEND-RESPONSE 기능으로 동일한 핸들러 호출
RECEIVE-REQUEST	서비스 제공자; 요청 단계(Phase)	부재	존재(길이 = 0)	(오류)
RECEIVE-REQUEST	서비스 제공자; 요청 단계(Phase)	부재	부재	NO-RESPONSE 기능으로 동일한 핸들러 호출
SEND-RESPONSE	서비스 제공자; 응답 단계(Phase)	적용되지 않음	존재(길이 > 0)	SEND-RESPONSE 기능으로 이전 핸들러 호출
SEND-RESPONSE	서비스 제공자; 응답 단계(Phase)	적용되지 않음	존재(길이 = 0)	(오류)
SEND-RESPONSE	서비스 제공자; 응답 단계(Phase)	적용되지 않음	부재	NO-RESPONSE 기능으로 동일한 핸들러 호출
SEND-REQUEST	서비스 요청자; 요청 단계(Phase)	존재(길이 > 0)	존재(길이 ≥ 0)	(오류)
SEND-REQUEST	서비스 요청자; 요청 단계(Phase)	존재(길이 > 0)	부재	SEND-REQUEST 기능으로 다음 핸들러 호출
SEND-REQUEST	서비스 요청자; 요청 단계(Phase)	존재(길이 = 0)	적용되지 않음	(오류)
SEND-REQUEST	서비스 요청자; 요청 단계(Phase)	부재	존재(길이 > 0)	응답 단계(Phase)로 전환 및 RECEIVE-RESPONSE 기능으로 이전 핸들러 호출
SEND-REQUEST	서비스 요청자; 요청 단계(Phase)	부재	존재(길이 = 0)	(오류)
SEND-REQUEST	서비스 요청자; 요청 단계(Phase)	부재	부재	NO-RESPONSE 기능으로 동일한 핸들러 호출
RECEIVE-RESPONSE	서비스 요청자; 응답 단계(Phase)	적용되지 않음	존재(길이 > 0)	RECEIVE-RESPONSE 기능으로 이전 핸들러 호출
RECEIVE-RESPONSE	서비스 요청자; 응답 단계(Phase)	적용되지 않음	존재(길이 = 0)	(오류)
RECEIVE-RESPONSE	서비스 요청자; 응답 단계(Phase)	적용되지 않음	부재	NO-RESPONSE 기능으로 동일한 핸들러 호출
PROCESS-REQUEST	서비스 제공자; 터미널 핸들러	적용되지 않음	존재(길이 > 0)	RECEIVE-RESPONSE 기능으로 이전 핸들러 호출
PROCESS-REQUEST	서비스 제공자; 터미널 핸들러	적용되지 않음	존재(길이 = 0)	(오류)

DFHFUNCTION	컨텍스트	DFHREQUEST	DFHRESPONSE	조치
PROCESS-REQUEST	서비스 제공자; 터미널 핸들러	적용되지 않음	부재	NO-RESPONSE 기능으로 동일한 핸들러 호출
HANDLER-ERROR	서비스 요청자 또는 제공자; 두 단계(Phase) 중 하나	적용되지 않음	존재(길이 > 0)	SEND-RESPONSE 기능 또는 RECEIVE-RESPONSE 기능으로 이전 핸들러 호출
HANDLER-ERROR	서비스 요청자 또는 제공자; 두 단계(Phase) 중 하나	적용되지 않음	존재(길이 = 0)	(오류)
HANDLER-ERROR	서비스 요청자 또는 제공자; 두 단계(Phase) 중 하나	적용되지 않음	부재	NO-RESPONSE 기능으로 동일한 핸들러 호출

컨텍스트 컨테이너

경우에 따라, 사용자 작성 메시지 핸들러 프로그램과 헤더 처리 프로그램에는 호출되는 컨텍스트에 대한 정보가 필요합니다. CICS는 프로그램으로 전달되는 컨텍스트 컨테이너 세트에 이 정보를 제공합니다.

CICS는 각 컨텍스트 컨테이너의 콘텐츠를 초기화하지만 경우에 따라 메시지 핸들러 프로그램과 헤더 처리 프로그램의 콘텐츠를 변경할 수 있습니다. 예를 들어, 터미널 핸들러가 CICS 제공 SOAP 핸들러 중 하나인 서비스 제공자 파이프라인의 경우, 해당 컨텍스트 컨테이너의 콘텐츠를 수정하여 대상 애플리케이션 프로그램의 사용자 ID와 트랜잭션 ID를 변경할 수 있습니다.

컨테이너에서 제공되는 일부 정보는 서비스 제공자 또는 서비스 요청자에만 적용되므로 일부 컨텍스트 컨테이너는 서비스 제공자와 서비스 요청자에서 모두 사용할 수 없습니다.

DFH-EXIT-HEADER1 컨테이너

DFH-EXIT-HEADER1은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 CICS의 웹 서비스 제공자 애플리케이션으로부터의 응답에 추가되는 하나 이상의 SOAP 헤더를 포함합니다.

글로벌 사용자 종료 XWSPPRWO를 실행하는 프로그램이 SOAP 응답에 헤더를 추가할 수 있습니다. 헤더는 올바른 SOAP여야 하며 네임스페이스는 헤더 XML에 자체 포함되어야 합니다. 이 컨테이너에 데이터를 저장하는 프로그램은 데이터가 있는지 확인하고 데이터 끝에 새 헤더를 추가해야 합니다. 이 우수 사례에 따라 필요한 경우 동일한 종료점에서 여러 프로그램이 구동될 수 있습니다.

DFH-HANDLERPLIST 컨테이너

DFH-HANDLERPLIST는 파이프라인 구성 파일의 해당 <handler_parameter_list> 요소 콘텐츠와 함께 초기화되는 DATATYPE(CHAR) 컨테이너입니다.

파이프라인 구성 파일에 핸들러 매개변수 목록을 지정하지 않은 경우에는 컨테이너가 비어 있습니다. 즉, 길이가 0입니다.

이 컨테이너의 콘텐츠는 변경할 수 없습니다.

DFH-SERVICEPLIST 컨테이너

DFH-SERVICEPLIST는 파이프라인 구성 파일의 <service_parameter_list> 요소 콘텐츠를 포함하는 DFH-SERVICEPLIST 컨테이너입니다.

파이프라인 구성 파일에 서비스 매개변수 목록을 지정하지 않은 경우에는 컨테이너가 비어 있습니다. 즉, 길이가 0입니다.

이 컨테이너의 콘텐츠는 변경할 수 없습니다.

DFHWS-APPHANDLER 컨테이너

DFHWS-APPHANDLER는 서비스 제공자 파이프라인의 경우 파이프라인 구성 파일의 <apphandler> 요소 콘텐츠와 함께 초기화되는 DATATYPE(CHAR) 컨테이너입니다.

<apphandler> 요소를 포함하는 파이프라인의 터미널 핸들러의 경우, 제공된 SOAP 핸들러가 이 컨테이너에서 대상 애플리케이션 프로그램의 이름을 가져옵니다.

이 컨테이너의 콘텐츠는 메시지 핸들러 또는 헤더 처리 프로그램에서 변경할 수 있습니다.

CICS는 서비스 요청자 파이프라인에서 이 컨테이너를 제공하지 않습니다.

관련 개념:

327 페이지의 『애플리케이션 핸들러』

애플리케이션 핸들러는 SOAP 서비스 제공자 파이프라인의 터미널 핸들러가 런타임 시 링크되는 CICS 프로그램입니다.

관련 참조:

281 페이지의 『<apphandler> 요소』

파이프라인의 터미널 핸들러가 기본적으로 링크되는 애플리케이션 핸들러의 이름을 지정합니다.

DFHWS-APPHANCLAS 컨테이너

DFHWS-APPHANCLAS는 서비스 제공자 파이프라인의 경우 파이프라인 구성 파일의 <apphandler_class> 요소 콘텐츠와 함께 초기화되는 DATATYPE(CHAR) 컨테이너입니다.

Java 기반 파이프라인의 터미널 핸들러에서, 제공된 SOAP 핸들러 (<cics_soap_1.1_handler_java> 및 <cics_soap_1.2_handler_java>)가 이 컨테이너에서 대상 애플리케이션 프로그램의 이름을 가져옵니다.

CICS는 서비스 요청자 파이프라인에서 이 컨테이너를 제공하지 않습니다.

관련 개념:

327 페이지의 『애플리케이션 핸들러』

애플리케이션 핸들러는 SOAP 서비스 제공자 파이프라인의 터미널 핸들러가 런타임 시 링크되는 CICS 프로그램입니다.

관련 참조:

281 페이지의 『<apphandler_class> 요소』

파이프라인의 터미널 핸들러가 Axis2 애플리케이션 핸들러에 링크되도록 지정합니다.

DFHWS-DATA 컨테이너

DFHWS-DATA는 서비스 요청자 애플리케이션에서 및 CICS 웹 서비스 지원과 함께 배치되는 서비스 제공자 애플리케이션에서 선택적으로 사용되는 DATATYPE(BIT) 컨테이너입니다. 이 컨테이너는 SOAP 요청에 매핑되거나 이 요청에서 매핑되는 최상위 레벨 데이터 구조를 포함합니다.

서비스 요청자 애플리케이션의 경우, 서비스 요청자 프로그램이 **EXEC CICS INVOKE SERVICE** 명령을 실행할 때 DFHWS-DATA 컨테이너가 있어야 합니다. 이 명령이 실행되면 CICS가 컨테이너에 있는 데이터 구조를 SOAP 요청으로 변환합니다. SOAP 응답이 수신되면 CICS가 동일한 컨테이너의 애플리케이션으로 리턴되는 다른 데이터 구조로 변환합니다.

서비스 제공자 애플리케이션의 경우, DFHLS2WS 또는 DFHWS2LS 일괄처리 작업에 대해 **CONTID** 매개변수를 지정하지 않으면 기본적으로 DFHWS-DATA 컨테이너를 사용합니다. CICS는 SOAP 요청 메시지를 DFHWS-DATA 컨테이너의 애플리케이션으로 전달되는 데이터 구조로 변환합니다. 그런 다음 동일한 컨테이너에 응답이 저장되며 CICS가 데이터 구조를 SOAP 응답 메시지로 변환합니다.

DFHWS-FAULT 컨테이너

DFHWS-FAULT는 CICS가 생성하는 SOAP 결함의 유형에 대한 정보를 포함하는 DATATYPE(BIT) 컨테이너입니다.

이 컨테이너에는 웹 서비스 응답에 대한 추가 처리에 사용될 수 있는 결함 유형을 나타내는 2진 실어가 포함됩니다.

1. 최신 SOAP 결함은 CICS 결함(예를 들어, CICS 또는 사용자 이상 종료)에 대한 것입니다.

2. 최신 SOAP 결함은 애플리케이션 결함에 대한 것입니다. EXEC CICS SOAPFAULT DELETE 명령을 실행하면 컨테이너가 삭제됩니다. 두 번째 또는 새 SOAP 결함이 작성되면 CICS가 그에 따라 새 컨테이너를 갱신합니다.

이 컨테이너의 콘텐츠는 변경할 수 없습니다.

DFHWS-LOCATION 컨테이너

DFHWS-LOCATION은 HTTP 응답이 302, 303 또는 307인 경우 제공된 위치 헤더를 포함하는 DATATYPE(CHAR) 컨테이너입니다.

DFHWS-MEP 컨테이너

DFHWS-MEP는 인바운드 또는 아웃바운드 SOAP 메시지의 메시지 교환 패턴(MEP)에 대한 대표 값을 포함하는 DATATYPE(BIT) 컨테이너입니다. 이 값의 길이는 1바이트입니다.

CICS는 서비스 요청자와 서비스 제공자 모두에 대한 네 가지 메시지 교환 패턴을 지원합니다. 메시지 교환 패턴은 웹 서비스에 대한 WSDL 2.0 문서에서 정의되며 CICS가 제공자로 응답하는지 또한 CICS가 외부 제공자로부터의 응답을 예상하는지 여부를 결정합니다. 요청자 모드에서 CICS가 응답을 대기하는 시간은 PIPELINE 자원을 사용하여 구성됩니다.

CICS 웹 서비스 지원을 사용하여 애플리케이션을 배치한 경우에는 다음과 같이 CICS로 이 컨테이너를 채웁니다.

- 서비스 제공자 파이프라인의 경우, 이 컨테이너는 터미널 핸들러에서 인바운드 메시지를 수신할 때 DFHPITP 애플리케이션 핸들러로 채워집니다.
- 서비스 요청자 파이프라인의 경우, 이 컨테이너는 애플리케이션이 **INVOKE SERVICE** 명령을 사용할 때 채워집니다.

애플리케이션이 DFHPIRT 채널을 사용하여 파이프라인을 시작하는 경우에는 애플리케이션이 이 컨테이너를 채웁니다. 컨테이너가 없거나 값을 갖지 않는 경우에는 CICS에서 DFHNORESPONSE 컨테이너가 채널에 있는지 여부에 따라 요청이 In-Out 또는 In-Only MEP를 사용하는 것으로 가정합니다.

이 컨테이너는 제공된 애플리케이션 핸들러 프로그램, DFHPITP로 채워집니다. 다른 애플리케이션 핸들러를 사용하는 경우에는 이 컨테이너를 사용할 수 없습니다.

표 9. DFHWS-MEP 컨테이너에 나타날 수 있는 값

값	MEP	URI
1	In-Only	http://www.w3.org/ns/wsd/in-only
2	In-Out	http://www.w3.org/ns/wsd/in-out
4	Robust-In-Only	http://www.w3.org/ns/wsd/robust-in-only
8	In-Optional-Out	http://www.w3.org/ns/wsd/in-opt-out

DFHWS-OPERATION 컨테이너

DFHWS-OPERATION은 CICS 웹 서비스 지원과 함께 배치되는 서비스 제공자 애플리케이션에서 일반적으로 사용되는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 SOAP 요청에 지정된 조작의 이름을 포함합니다.

서비스 제공자의 경우, 이 컨테이너는 애플리케이션이 호출되는 조작의 이름을 제공합니다. 이 컨테이너는 제공된 SOAP 메시지 핸들러가 대상 애플리케이션 프로그램으로 제어를 전달할 때 채워지며 대상 프로그램이 채널 인터페이스와 함께 호출되는 경우에만 표시됩니다.

서비스 요청자 파이프라인의 경우, 이 컨테이너는 **EXEC CICS INVOKE SERVICE** 명령의 **OPERATION** 옵션에 지정된 이름을 포함합니다. 명령을 실행하는 애플리케이션은 이 컨테이너를 사용할 수 없습니다.

이 컨테이너는 제공된 애플리케이션 핸들러 프로그램, DFHPITP로 채워집니다. 다른 애플리케이션 핸들러를 사용하는 경우에는 이 컨테이너를 사용할 수 없습니다.

DFHWS-PIPELINE 컨테이너

DFHWS-PIPELINE은 프로그램이 실행되는 PIPELINE의 이름을 포함하는 DATATYPE(CHAR) 컨테이너입니다.

이 컨테이너의 콘텐츠는 변경할 수 없습니다.

DFHWS-RESPWAIT 컨테이너

DFHWS-RESPWAIT은 아웃바운드 웹 서비스 요청 및 응답 메시지에 적용되는 제한 시간(초)을 나타내는 부호 없는 실어 2진 숫자를 포함하는 DATATYPE(BIT) 컨테이너입니다.

이 컨테이너의 값은 PIPELINE 정의의 RESPWAIT 속성으로 제공하며 INVOKE SERVICE 명령이 실행될 때 CICS로 설정됩니다. INVOKE SERVICE 명령이 실행되기 전에 사용자 애플리케이션이 이 컨테이너에서 설정한 값은 모두 무시됩니다.

파이프라인 처리 중에 호출되는 메시지 핸들러 프로그램은 DFHWS-RESPWAIT 컨테이너의 값을 겹쳐쓸 수 있습니다. 이 작업을 수행하는 경우, PIPELINE 정의가 DEFT로 설정되지 않거나 공백으로 남겨진 RESPWAIT 속성을 갖는 경우에만 갱신된 값이 사용됩니다. PIPELINE 정의에서 RESPWAIT 속성이 DEFT로 설정되거나 공백으로 남아 있는 경우, DFHWS-RESPWAIT 컨테이너의 값에 관계 없이 항상 전송 프로토콜의 기본 제한시간 값이 사용됩니다.

이 컨테이너는 요청자 모드 파이프라인에서만 사용됩니다.

DFHWS-SOAPLEVEL 컨테이너

DFHWS-SOAPLEVEL은 처리하는 메시지에서 사용되는 SOAP 레벨에 대한 정보를 포함하는 DATATYPE(BIT) 컨테이너입니다.

이 컨테이너는 웹 서비스 요청 또는 응답에 사용되는 SOAP 레벨을 나타내는 2진 실어를 포함합니다.

- 1 요청 또는 응답은 SOAP 1.1 메시지입니다.
- 2 요청 또는 응답은 SOAP 1.2 메시지입니다.
- 10 요청 또는 응답은 SOAP 메시지가 아닙니다.

이 컨테이너의 콘텐츠는 변경할 수 없습니다.

DFHWS-TRANID 컨테이너

DFHWS-TRANID는 파이프라인이 실행 중인 태스크의 트랜잭션 ID로 초기화되는 DATATYPE(CHAR) 컨테이너입니다.

터미널 핸들러가 CICS 제공 SOAP 핸들러 중 하나인 서비스 제공자 파이프라인에서 이 컨테이너의 콘텐츠를 변경하는 경우(또한 대상 애플리케이션 프로그램으로 제어가 전달되기 전에 이 작업을 수행하는 경우) 대상 애플리케이션은 새 트랜잭션 ID로 새 태스크에서 실행됩니다.

파이프라인의 애플리케이션 핸들러와 터미널 핸들러가 둘 다 동일한 JVM 서버에서 실행되는 경우에는 새 태스크가 시작되지 않습니다. 이러한 이유로, JAX-WS Axis2 애플리케이션을 CICS에 배치하는 경우 DFHWS-TRANID를 사용하여 사용자 ID를 변경할 수 없습니다.

DFHWS-URI 컨테이너

DFHWS-URI는 서비스 URI를 포함하는 DATATYPE(CHAR) 컨테이너입니다.

서비스 제공자 파이프라인의 경우, CICS가 수신 메시지에서 상대 URI를 추출하여 DFHWS-URI 컨테이너에 배치합니다.

예를 들어, 웹 서비스의 URI가 `http://example.com/location/address` 또는 `jms://queue?destination=INPUT.QUEUE&targetService=/location/address`인 경우 상대 URI는 `/location/address`입니다.

요청자 파이프라인에서 WS-Addressing을 사용하는 경우, 이 컨테이너는 다음 순서로 작성, 갱신됩니다.

1. INVOKE SERVICE 명령이 실행되면 DFHWS-URI 컨테이너를 작성하며 WSDL 서비스 엔드포인트 주소 값으로 컨테이너를 시작합니다. WSACONTEXT BUILD API 명령을 사용하여 주소 지정 컨텍스트를 작성한 경우에는 INVOKE SERVICE 명령에서 URI 또는 URIMAP 매개변수를 지정해서는 안 됩니다.

2. WS-Addressing 핸들러(DFHWSADH)가 실행될 때 비익명 URI를 포함하는 주소 지정 컨텍스트에 <wsa:To> EPR이 존재하는 경우, DFHWS-URI 컨테이너의 URI를 <wsa:To> EPR의 값으로 겹쳐씁니다. 익명 URI는 무시됩니다.

SOAP 메시지는 DFHWS-URI에서 URI가 정의한 서비스로 전송됩니다.

서비스 요청자 파이프라인에서, CICS는 **INVOKE SERVICE** 명령에 지정된 URI 또는 누락된 경우 웹 서비스 바인딩의 URI를 DFHWS-URI 컨테이너에 둡니다. 이 URI는 파이프라인에서 메시지 핸들러를 사용하여 대체할 수 있습니다.

서비스는 외부 서비스에 대한 HTTP, HTTPS, JMS 또는 WebSphere MQ URI를 사용할 수 있습니다. 서비스는 또한 다른 CICS 애플리케이션이 제공하는 서비스에 CICS URI를 사용할 수 있습니다.

URI	조회 문자열	설명
cics://PROGRAM/ <i>program</i>	? <i>options</i>	CICS 전송 핸들러는 EXEC CICS LINK PROGRAM 명령을 사용하여 현재 채널과 컨테이너를 전달함으로써 지정된 프로그램에 링크됩니다. 애플리케이션 데이터에 대한 데이터 변환은 수행되지 않습니다.
cics://SERVICE/ <i>service</i>	?targetServiceUri= <i>targetServiceUri</i> & <i>options</i>	CICS 전송 핸들러는 서비스 경로 (<i>targetServiceUri</i> 로 표시됨)를 사용하여 제공자 파이프라인을 통해 요청을 실행할 URIMAP 자원을 일치시킵니다. 이 URI 유형을 사용하는 경우에는 targetServiceUri 매개변수의 값을 지정해야 합니다.
cics://PIPELINE/ <i>pipeline</i>	?targetServiceUri= <i>targetServiceUri</i>	CICS 전송 핸들러가 다른 서비스 요청자 파이프라인을 시작합니다.

parameter=value 형식을 사용하여 CICS URI의 각 유형에 매개변수를 추가할 수 있습니다. 여기서 각 매개변수는 앰퍼샌드로 구분됩니다. 다음 규칙은 CICS URI에 적용됩니다.

- 조회 문자열의 첫 번째 매개변수에는 물음표 접두부가 추가되어야 합니다. URI에서 이 지점 앞에 물음표를 사용할 수 없습니다.
- 매개변수 값에 앰퍼샌드를 포함하려면 문자를 이스케이프해야 합니다. 자세한 정보는 이 주제 끝의 예제 섹션을 참조하십시오.
- CICS는 *program* 및 *pipeline*의 소문자 값을 대문자로 변경합니다.

조회 문자열의 매개변수가 요청자 파이프라인 끝에서 CICS가 요청을 처리하는 방법을 결정합니다.

maxCommareaLength=value

대상 애플리케이션 프로그램에 필요한 COMMAREA의 최대 크기(바이트)를 지정

합니다. 값은 32 763을 초과해서는 안됩니다. 이 매개변수가 조회 문자열에 있으면 CICS가 COMMAREA를 사용하여 지정된 프로그램에 링크됩니다. 이 매개변수가 조회 문자열에 없으면 CICS가 채널을 사용하여 지정된 프로그램에 링크됩니다.

이 매개변수는 대소문자를 구분하지 않으며 `cics://PROGRAM` URI에만 유효합니다.

newTask=yes|no

전송 핸들러가 요청을 새 태스크로 실행하는지 여부를 지정합니다.

이 매개변수는 대소문자를 구분하지 않습니다.

`cics://PROGRAM/testapp?newTask=yes`와 `cics://PROGRAM/testapp?NEWTASK=Yes`는 같습니다.

targetServiceUri=uri

호출된 서비스의 경로를 지정합니다. SERVICE 대상 유형의 경우, 전송 핸들러는 `host=localhost`인 값을 사용하여 서비스 제공자 파이프라인을 시작할 URIMAP 자원을 찾습니다. PIPELINE 대상 유형의 경우, 전송 핸들러는 이 값을 사용하여 다른 요청자 파이프라인을 시작합니다.

이 매개변수는 대소문자를 구분합니다.

transid=char(4)

요청이 실행될 트랜잭션을 지정합니다. 전송 핸들러는 지정된 트랜잭션 ID를 사용하여 요청 스트림을 시작합니다.

이 매개변수는 대소문자를 구분합니다.

userid=char(8)

요청이 실행될 사용자 ID를 지정합니다. 전송 핸들러는 지정된 사용자 ID를 사용하여 요청 스트림을 시작합니다.

이 매개변수는 대소문자를 구분하지 않습니다.

대상 유형	URI 매개변수	
PROGRAM	userid	선택사항
PROGRAM	transid	선택사항
PROGRAM	maxCommareaLength	선택사항
PROGRAM	newTask	선택사항. 'yes'이거나 userid 또는 transid 를 지정하는 경우 전혀 지정되지 않아야 합니다.
PROGRAM	targetServiceUri	지원되지 않음
SERVICE	userid	선택사항
SERVICE	transid	선택사항
SERVICE	maxCommareaLength	지원되지 않음

대상 유형	URI 매개변수	
SERVICE	newTask	선택사항. 'yes'이거나 userid 또는 transid 를 지정하는 경우 전혀 지정되지 않아야 합니다.
SERVICE	targetServiceUri	필수사항
PIPELINE	userid	지원되지 않음
PIPELINE	transid	지원되지 않음
PIPELINE	maxCommareaLength	지원되지 않음
PIPELINE	newTask	지원되지 않음
PIPELINE	targetServiceUri	필수사항

CICS URI 예제

이 첫 번째 예제에서 DFHWS-URI 컨테이너는 파이프라인 끝에 도달할 때까지 다음 URI를 갖습니다.

```
cics://PROGRAM/testapp?newTask=yes&userid=user1
```

전송 핸들러는 채널과 컨테이너를 전달하여 CICS 프로그램 testapp에 링크됩니다. 데이터 변환은 일어나지 않으므로 대상 프로그램이 현재 채널에서 컨테이너의 콘텐츠를 처리할 수 있어야 합니다. CICS는 새 작업 단위와 다른 사용자 ID, user1로 프로그램에 링크됩니다.

이 두 번째 예제에서 DFHWS-URI 컨테이너는 파이프라인 끝에 도달할 때까지 다음 URI를 갖습니다.

```
cics://SERVICE/getStockQuote?targetServiceUri=/stock/getQuote&newTask=yes&userid=user2
```

전송 핸들러는 DFHWS-URI 컨테이너의 URI를 값 /stock/getQuote로 바꾸고 **targetServiceUri** 매개변수의 경로를 사용하여 URIMAP를 찾은 다음 새 태스크와 다른 사용자 ID로 제공자 파이프라인을 시작합니다.

이 세 번째 예제에서 DFHWS-URI 컨테이너는 파이프라인 끝에 도달할 때까지 다음 URI를 갖습니다.

```
cics://PIPELINE/reqpipeA?targetServiceUri=cics://PROGRAM/testapp?newTask=yes%26userid=user1
```

전송 핸들러는 DFHWS-URI 컨테이너의 URI를 값 `cics://PROGRAM/testapp?newTask=yes&userid=user1`로 바꾸고 현재 채널과 컨테이너를 전달하여 요청자 파이프라인 reqpipeA를 시작합니다. %26 문자는 앰퍼샌드를 이스케이프하므로 전송 핸들러가 전체 URI를 DFHWS-URI 컨테이너에 둡니다.

관련 개념:

386 페이지의 『요청자 파이프라인 처리 제어를 위한 옵션』

서비스 요청자 파이프라인에서 메시지 핸들러는 웹 서비스 요청이 URI를 변경하여 전송되는 위치를 판별할 수 있습니다. CICS에서는 파이프라인 처리 웹 서비스가 요청하

는 방식으로 더 많은 유연성을 가지도록 다른 URI 형식에 대한 지원을 제공합니다.

관련 태스크:

388 페이지의 『URI을 사용하여 요청자 파이프라인 처리 제어』

서비스 요청자 파이프라인에서 메시지 핸들러는 URI를 변경하여 웹 서비스 요청을 전송할 위치를 판별할 수 있습니다. URI 형식을 변경하여 특정 최적화 수행(예: 네트워크를 통한 요청 전송 없이 다른 요청자 파이프라인 시작 또는 서비스 제공자 파이프라인 시작)을 선택할 수 있습니다.

DFHWS-URI-RESID 컨테이너

JSON 파이프라인으로 연결되는 애플리케이션만 사용할 수 있는 DATATYPE(CHAR) 컨테이너입니다.

이 컨테이너는 URIMAP 일치에 사용된 경로 URI 단편이 제거된 URI 경로(RESource Identifier)의 단순 사본을 포함합니다. 예를 들어, 다음과 같습니다.

수신 요청을 일시치킨 URIMAP의 PATH가 다음과 같고

/JSONServices/CustomDetails/*

클라이언트로부터의 수신 URI가 다음과 같은 경우

<http://www.example.org:10000/JSONServices/CustomDetails/customerNumber/13388?action=query>

DFHWS-URI-RESID 콘텐츠는 다음과 같습니다.

customerNumber/13388

RESTful JSON 애플리케이션은 이 컨테이너를 사용하여 와일드카드 URIMAP를 사용하여 일치시킨 RESTful 자원의 자원 ID(또는 기본 키)를 식별하는 데 도움을 줄 수 있습니다. 이는 DFHWS-URI 콘텐츠 전체를 구문 분석하는 것보다 훨씬 간단해야 합니다.

참고: 일치 URIMAP의 PATH 속성이 와일드카드가 아닌 경우(즉, URI의 전체 경로를 포함한 경우) 이 컨테이너의 콘텐츠가 비어 있습니다.

참고: 일치 URIMAP의 PATH 속성에는 선택적인 조회 문자열 단편이 포함될 수 있습니다. 이러한 경우 이 컨테이너를 구성할 때 조회 문자열 단편이 무시됩니다.

DFHWS-URI-QUERY 컨테이너

모든 HTTP 제공자 모드 CICS 파이프라인에서 애플리케이션 프로그램이 사용할 수 있는 DATATYPE(CHAR) 컨테이너입니다.

이 컨테이너는 URI의 조회 문자열 단편을 포함합니다. 예를 들어, 다음과 같습니다.

클라이언트로부터의 수신 URI가 다음과 같은 경우

<http://www.example.org:10000/JSONServices/CustomDetails/customerNumber/13388?action=query&page=1>

DFHWS-URI-QUERY 컨텐츠는 다음과 같습니다.

action=query&page=1

애플리케이션은 이 컨테이너 컨텐츠 전체를 구문 분석하여 URI에서 개별 **name=value** 매개변수를 식별할 수 있습니다.

참고: 수신 URI에 조회 문자열이 포함되지 않는 경우에는 이 컨테이너가 채널에 존재하지 않습니다.

DFHWS-URIMAPPATH 컨테이너

DATATYPE(CHAR) 컨테이너로 수신 URI와 일치시키는 데 사용된 URIMAP에서 PATH 데이터의 사본을 보유합니다.

파이프라인 연결 애플리케이션은 이 컨테이너를 사용하여 자세한 연결 방법을 이해할 수 있습니다.

DFHWS-USERID 컨테이너

DFHWS-USERID는 파이프라인이 실행되는 태스크의 사용자 ID로 초기화되는 DATATYPE(CHAR) 컨테이너입니다.

터미널 핸들러가 CICS 제공 SOAP 핸들러 중 하나인 서비스 제공자 파이프라인에서 이 컨테이너의 컨텐츠를 변경하는 경우(및 대상 애플리케이션 프로그램으로 제어가 전달되기 전에 이 작업을 수행하는 경우) 대상 애플리케이션은 새 사용자 ID와 연관된 새 태스크에서 실행됩니다. 컨테이너 DFHWS-TRANID의 컨텐츠를 변경하지 않는 경우 새 태스크는 파이프라인이 실행 중인 태스크와 동일한 트랜잭션 ID를 갖습니다.

파이프라인의 애플리케이션 핸들러와 터미널 핸들러가 둘 다 동일한 JVM 서버에서 실행되는 경우에는 새 태스크가 시작되지 않습니다. 이러한 이유로, JAX-WS Axis2 애플리케이션을 CICS에 배치하는 경우 DFHWS-USERID를 사용하여 사용자 ID를 변경할 수 없습니다.

DFHWS-WEBSERVICE 컨테이너

DFHWS-WEBSERVICE는 서비스 제공자 파이프라인에서만 사용되는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 대상 애플리케이션이 웹 서비스 지원을 사용하여 배치된 경우 실행 환경을 지정하는 웹 서비스의 이름입니다.

CICS는 서비스 요청자 파이프라인에서 이 컨테이너를 제공하지 않습니다.

DFHWS-CID-DOMAIN 컨테이너

DFHWS-CID-DOMAIN은 DATATYPE(CHAR) 컨테이너입니다. 2진 첨부 파일을 참조하기 위해 컨텐츠-ID 값을 생성하는 데 사용되는 도메인 이름을 포함합니다.

도메인 이름의 값은 기본적으로 `cicsts`입니다. 파이프라인 구성 파일에서 `<mime_options>` 요소를 지정하여 이 값을 대체할 수 있습니다.

이 컨테이너의 콘텐츠는 변경할 수 없습니다.

DFHWS-MTOM-IN 컨테이너

DFHWS-MTOM-IN은 파이프라인 구성 파일의 `<cics_mtom_handler>` 요소에 지정된 옵션에 대한 정보와 파이프라인에서 수신된 메시지 형식에 대한 정보를 포함하는 DATATYPE(BIT) 컨테이너입니다.

파이프라인에서 인바운드 MTOM 메시지를 처리하기 위한 정보를 포함합니다. 인바운드 메시지는 웹 서비스 요청자로부터의 요청 메시지 또는 웹 서비스 제공자로부터의 응답 메시지가 될 수 있습니다.

파이프라인 구성 파일에 `<cics_mtom_handler>` 요소를 지정하지 않거나 MTOM 메시지 대신 SOAP 메시지가 수신되는 경우에는 이 컨테이너가 작성되지 않습니다.

파이프라인에서 웹 서비스 보안이 구성되거나 웹 서비스에 대한 유효성 검증이 활성화되는 경우, 컨테이너가 작성될 때 DFHWS-MTOM-IN에서 필드 `XOP_MODE`의 콘텐츠를 CICS가 대체할 수 있습니다. 예를 들어, 파이프라인이 MTOM 메시지의 직접 모드에서 콘텐츠를 처리하도록 구성한 다음 웹 서비스에 대한 유효성 검증을 활성화하면 CICS가 파이프라인 구성 파일에서 정의된 값을 대체하고 XOP 처리가 호환 모드에서 실행되도록 설정합니다. CICS는 파이프라인의 XOP 문서 및 2진 첨부 파일 처리에 대한 지원 제한사항으로 인해 대체를 수행합니다.

이 컨테이너의 콘텐츠는 변경할 수 없습니다.

표 10. DFHWS-MTOM-IN 컨테이너 구조

필드 이름	길이(바이트)	콘텐츠
MTOM_STATUS	4	CICS가 수신한 메시지가 MTOM 형식임을 나타내는 값 "1"을 포함합니다.
MTOMNOXOP_STATUS	4	다음 값 중 하나를 포함합니다. 0 MTOM 메시지에 2진 첨부 파일이 포함됩니다. 1 MTOM 메시지에 2진 첨부 파일이 포함되지 않습니다.
XOP_MODE	4	다음 값 중 하나를 포함합니다. 0 XOP 처리가 수행되지 않습니다. 1 XOP 처리가 호환 모드에서 수행됩니다. 2 XOP 처리가 직접 모드에서 수행됩니다.

DFHWS-MTOM-OUT 컨테이너

DFHWS-MTOM-OUT은 파이프라인 구성 파일의 <cics_mtom_handler> 요소에 지정된 옵션에 대한 정보를 포함하는 DATATYPE(BIT) 컨테이너입니다.

웹 서비스 요청자에 대한 응답 메시지 또는 웹 서비스 제공자에 대한 요청 메시지 여부에 관계 없이 파이프라인에서 아웃바운드 MTOM 메시지를 처리하기 위한 정보가 포함됩니다.

파이프라인 구성 파일에서 <cics_mtom_handler> 요소를 지정하지 않거나 파이프라인 구성 파일의 <mtom_options> 요소가 속성 send_mtom="no"를 갖는 경우에는 이 컨테이너가 작성되지 않습니다.

제공자 모드에서는 DFHWS-MTOM-IN 컨테이너와 동시에 이 컨테이너가 작성됩니다. 파이프라인 구성 파일의 <mtom_options> 요소가 속성 send_mtom="same"을 가지는 경우, MTOM_STATUS 필드는 웹 서비스 요청자가 MTOM 또는 SOAP 응답 메시지를 원하는지 여부를 나타내기 위해 설정됩니다.

파이프라인에서 웹 서비스 보안이 구성되거나 웹 서비스에 대해 해당 유효성 검증이 활성화되는 경우, 컨테이너가 작성될 때 CICS가 DFHWS-MTOM-OUT의 XOP_MODE 필드를 변경할 수 있습니다. 예를 들어, 파이프라인에서 직접 모드를 사용하여 XOP 문서와 2진 첨부 파일을 처리하도록 구성한 다음 웹 서비스에 대한 유효성 검증을 활성화하면 CICS가 파이프라인 구성 파일에서 정의된 값을 대체하고 컨테이너 작성 시 XOP 처리가 호환 모드에서 실행되도록 설정합니다. CICS는 파이프라인의 XOP 문서 및 2진 첨부 파일 처리에 대한 지원 제한사항으로 인해 대체를 수행합니다.

이 컨테이너의 콘텐츠는 변경할 수 없습니다.

표 11. DFHWS-MTOM-OUT 컨테이너 구조

필드 이름	길이(바이트)	콘텐츠
MTOM_STATUS	4	MTOM을 사용하는지 여부를 표시합니다. 0 MTOM을 사용하지 않습니다. 아웃바운드 메시지는 SOAP 형식으로 전송됩니다. 1 MTOM을 사용합니다. 아웃바운드 메시지는 MTOM 형식으로 전송됩니다.
MTOMNOXOP_STATUS	4	2진 첨부 파일이 없는 경우 MTOM을 사용할지 여부를 표시합니다. 0 2진 첨부 파일이 없는 경우 MTOM 메시지를 보내지 않습니다. 1 2진 첨부 파일이 없는 경우 MTOM 메시지를 보냅니다.
XOP_MODE	4	수행되어야 하는 XOP 처리를 표시합니다. 0 XOP 처리가 수행되지 않습니다. 1 XOP 처리가 호환 모드에서 수행됩니다. 2 XOP 처리가 직접 모드에서 수행됩니다.

DFHWS-WSDL-CTX 컨테이너

DFHWS-WSDL-CTX는 CICS 웹 서비스 지원과 함께 배치되는 서비스 요청자 애플리케이션 또는 서비스 제공자에서 사용되는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 모니터링 목적으로 사용할 수 있는 WSDL 컨텍스트 정보를 포함합니다.

DFHWS-WSDL-CTX는 WSDL 문서에 대한 다음 컨텍스트 정보를 포함합니다.

- 애플리케이션이 호출되는 조작의 이름과 네임스페이스
- 알려진 경우, 사용 중인 WSDL 1.1 포트 또는 WSDL 2.0 엔드포인트의 이름과 네임스페이스

이 값은 공백 문자로 구분됩니다. DFHWS-WSDL-CTX는 런타임 레벨 2.1 이상에서만 CICS로 채워집니다.

CICS 웹 서비스 지원을 사용하여 애플리케이션을 배치한 경우에는 다음과 같이 CICS로 이 컨테이너를 채웁니다.

- 서비스 제공자 파이프라인의 경우, 이 컨테이너는 터미널 핸들러에서 인바운드 메시지를 수신할 때 DFHPITP 애플리케이션 핸들러로 채워집니다.
- 서비스 요청자 파이프라인의 경우, 이 컨테이너는 애플리케이션이 **INVOKE SERVICE** 명령을 사용할 때 채워집니다.

애플리케이션이 DFHPIRT 프로그램을 사용하여 파이프라인을 시작하는 경우에는 필요에 따라 애플리케이션이 DFHWS-WSDL-CTX 컨테이너를 채웁니다.

DFHWS-XOP-IN 컨테이너

DFHWS-XOP-IN은 DATATYPE(BIT) 컨테이너입니다. 이 컨테이너는 인바운드 MIME 메시지에서 패키지가 해제되고 XOP 처리를 사용하여 컨테이너에 배치된 2진 첨부 파일에 대한 참조 목록을 포함합니다.

DFHWS-XOP-IN 컨테이너의 각 첨부 레코드는 다음 항목으로 구성됩니다.

- 2진 첨부 파일의 MIME 헤더를 포함하는 컨테이너의 16바이트 이름
- 2진 첨부 파일을 포함하는 컨테이너의 16바이트 이름
- 콘텐츠-ID의 2바이트 길이(부호 있는 실어 2진 형식)
- ASCII 문자열로 저장되는 콘텐츠-ID(< 및 > 구분 기호 포함)

이 컨테이너의 콘텐츠는 변경할 수 없습니다.

DFHWS-XOP-OUT 컨테이너

DFHWS-XOP-OUT은 DATATYPE(BIT) 컨테이너입니다. 2진 첨부 파일을 포함하는 컨테이너에 대한 참조 목록이 포함됩니다. 2진 첨부 파일은 MTOM 핸들러 프로그램에서 아웃바운드 MIME 메시지로 패키징화됩니다.

DFHWS-XOP-OUT 컨테이너의 각 첨부 레코드는 다음 항목으로 구성됩니다.

- 2진 첨부 파일의 MIME 헤더를 포함하는 컨테이너의 16바이트 이름
- 2진 첨부 파일을 포함하는 컨테이너의 16바이트 이름
- 콘텐츠-ID의 2바이트 길이(부호 있는 실어 2진 형식)
- ASCII 문자열로 저장되는 콘텐츠-ID(< 및 > 구분 기호 포함)

이 컨테이너의 콘텐츠는 변경할 수 없습니다.

헤더 처리 프로그램 컨테이너

CICS 제공 SOAP 1.1 및 SOAP 1.2 메시지 핸들러는 채널 DFHHHC-V1을 사용하여 헤더 처리 프로그램에 링크됩니다. 채널에서 전달되는 컨테이너에는 헤더 처리 프로그램 인터페이스에 고유한 여러 컨테이너와 파이프라인의 모든 헤더 처리 프로그램과 메시지 핸들러 프로그램이 액세스할 수 있는 컨텍스트 컨테이너 및 사용자 컨테이너의 세트가 포함됩니다.

컨테이너 DFHHEADER는 헤더 처리 프로그램 인터페이스에 고유합니다. 기타 컨테이너는 파이프라인 내 다른 위치에서 사용할 수 있지만 헤더 처리 프로그램에서 특정 용도를 갖습니다. 이 카테고리의 컨테이너는 DFHWS-XMLNS, DFHWS-BODY, DFHXMLSS-PARSE입니다.

참고: Axis2를 사용하여 SOAP 메시지를 처리하는 웹 서비스는 헤더 처리 프로그램 인터페이스를 사용할 수 있지만 고유한 Axis2 핸들러를 Java로 작성하여 SOAP 헤더를 처리하는 것이 보다 효율적입니다. Axis2 핸들러 작성에 대한 자세한 정보는 Axis2 모듈 직접 작성을 참조하십시오.

컨테이너 DFHHEADER

헤더 처리 프로그램이 호출되면 헤더 처리 프로그램을 구동시킨 단일 헤더 블록이 DFHHEADER에 포함됩니다. 헤더 프로그램이 파이프라인 구성 파일에서 <mandatory>true</mandatory> 또는 <mandatory>1</mandatory>로 지정되면 SOAP 메시지에 일치하는 헤더 블록이 없더라도 해당 프로그램이 호출됩니다. 이 경우 컨테이너 DFHHEADER의 길이는 0입니다. 이 경우 헤더 블록이 없는 SOAP 메시지에 헤더 블록을 추가하기 위해 헤더 처리 프로그램이 호출됩니다.

CICS가 작성하는 SOAP 메시지에는 처음에 헤더가 없습니다. 메시지에 헤더를 추가하려면 <mandatory>true</mandatory> 또는 <mandatory>1</mandatory>를 지정하여 최소 하나의 헤더 처리 프로그램이 호출되도록 해야 합니다.

헤더 프로그램이 리턴되는 경우 컨테이너 DFHHEADER는 CICS가 원래 헤더 블록 대신 SOAP 메시지에 삽입하는 0개 이상의 헤더 블록을 포함해야 합니다.

- 원래 헤더 블록을 변경하지 않고 리턴할 수 있습니다.
- 헤더 블록의 콘텐츠를 수정할 수 있습니다.

- 원래 블록에 하나 이상의 새 헤더 블록을 추가할 수 있습니다.
- 원래 헤더 블록을 하나 이상의 다른 블록으로 바꿀 수 있습니다.
- 헤더 블록은 완전히 삭제할 수 있습니다.

컨테이너 DFHWS-XMLNS

헤더 처리 프로그램이 호출되면 SOAP 엔벨로프에서 선언된 XML 네임스페이스에 대한 정보가 DFHWS-XMLNS에 포함됩니다. 헤더 프로그램은 이 정보를 사용하여 다음 태스크를 수행할 수 있습니다.

- 헤더 블록에 나타나는 규정된 이름을 해석합니다.
- 새로운 또는 수정된 헤더 블록에서 규정된 이름을 구성합니다.

네임스페이스 정보는 네임스페이스 선언을 위해 표준 XML 표기법을 사용하는 네임스페이스 선언의 목록으로 구성됩니다. DFHWS-XMLNS의 네임스페이스 선언은 공백으로 구분됩니다. 예를 들어, 다음과 같습니다.

```
xmlns:na='http://abc.example.org/schema' xmlns:nx='http://xyz.example.org/schema'
```

DFHWS-XMLNS 콘텐츠에 추가하는 방식으로 SOAP 엔벨로프에 네임스페이스 선언을 더 추가할 수 있습니다. 그러나 범위가 SOAP 헤더 블록 또는 SOAP 본문인 네임스페이스가 헤더 블록 또는 본문에서 각각 가장 올바르게 선언됩니다. 프로그램에 표시되지 않는 XML 요소가 종속될 수 있으므로 헤더 처리 프로그램의 컨테이너 DFHWS-XMLNS에서 네임스페이스 선언을 삭제하지 않는 것이 좋습니다.

컨테이너 DFHWS-BODY

이 컨테이너는 SOAP 엔벨로프의 본문 섹션을 포함합니다. 헤더 처리 프로그램이 콘텐츠를 수정할 수 있습니다.

헤더 처리 프로그램이 호출되면 DFHWS-BODY에 SOAP <Body> 요소가 포함됩니다.

헤더 프로그램이 리턴되는 경우 컨테이너 DFHWS-BODY는 다시 CICS가 원래 본문 대신 SOAP 메시지에 삽입하는 올바른 SOAP <Body>를 포함해야 합니다.

- 원래 본문을 변경하지 않고 리턴할 수 있습니다.
- 본문의 콘텐츠를 수정할 수 있습니다.

SOAP 본문을 완전히 삭제해서는 안 됩니다. 모든 SOAP 메시지에 <Body> 요소가 포함되어야 하기 때문입니다.

컨테이너 DFHXMLSS-PARSE

파이프라인 구성에서 <cics_soap_1.1_handler> 또는 <cics_soap_1.2_handler> 요소를 사용하고 헤더 프로그램이 호출되면 해당 헤더에 대한 XMLSS(XML System Services) 레코드가 DFHXMLSS-PARSE에 포함됩니다. 이 컨테이너는

<cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소가 사용되는 경우에는 작성되지 않습니다.

제어, 컨텍스트 및 사용자 컨테이너

설명한 컨테이너뿐 아니라 인터페이스는 제어 컨테이너, 컨텍스트 컨테이너, 사용자 컨테이너를 채널 DFHHHC-V1에서 전달합니다.

이 컨테이너에 대한 자세한 정보는 345 페이지의 『파이프라인에서 사용되는 컨테이너』를 참조하십시오.

보안 컨테이너

보안 컨테이너는 DFHWSTC-V1 채널에서 Tivoli Federated Identity Manager와 같은 보안 토큰 서비스(STS)의 ID 토큰을 송수신하는 데 사용됩니다. 이 인터페이스는 신뢰 클라이언트 인터페이스라고 하며 웹 서비스 요청자 및 제공자 파이프라인에서 사용할 수 있습니다.

DFHWS-IDTOKEN 컨테이너

DFHWS-IDTOKEN은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 보안 토큰 서비스(STS)가 메시지의 ID 토큰을 발행하기 위해 사용하거나 유효성을 검증하는 토큰을 포함합니다.

토큰은 XML 형식이어야 합니다.

이 컨테이너는 신뢰 클라이언트 인터페이스의 채널 DFHWSTC-V1에만 사용하십시오.

DFHWS-RESTOKEN 컨테이너

DFHWS-RESTOKEN은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 보안 토큰 서비스(STS)로부터의 응답을 포함합니다.

응답은 DFHWS-STSACTION 컨테이너의 STS에서 요청된 조치에 따라 다릅니다.

- 조치가 실행인 경우, 이 컨테이너는 DFHWS-IDTOKEN 컨테이너에서 전송된 토큰에 대해 STS가 교환한 토큰을 포함합니다.
- 조치가 유효성 검증인 경우, 이 컨테이너는 DFHWS-IDTOKEN 컨테이너에서 전송된 보안 토큰이 올바른지 올바르지 않은지 여부를 나타내는 URI를 포함합니다. 리턴될 수 있는 URI는 다음과 같습니다.

URI	설명
http://schemas.xmlsoap.org/ws/2005/02/trust/status/valid	보안 토큰이 올바른 토큰입니다.
http://schemas.xmlsoap.org/ws/2005/02/trust/status/invalid	보안 토큰이 올바르지 않습니다.

이 컨테이너는 신뢰 클라이언트 인터페이스를 사용하는 경우 채널 DFHWSTC-V1에 리턴됩니다.

DFHWS-SERVICEURI 컨테이너

DFHWS-SERVICEURI는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 보안 토큰 서비스(STS)가 AppliesTo 범위로 사용하는 URI를 포함합니다.

AppliesTo 범위는 보안 토큰이 연관되는 웹 서비스를 판별하는 데 사용됩니다.

이 컨테이너는 신뢰 클라이언트 인터페이스의 채널 DFHWSTC-V1에만 사용하십시오.

DFHWS-STSACTION 컨테이너

DFHWS-STSACTION은 DATATYPE(CHAR) 컨테이너입니다. 보안 토큰 서비스(STS)가 보안 토큰을 발행하거나 유효성을 검증하기 위해 수행하는 조치의 URI를 포함합니다.

이 컨테이너에서 지정할 수 있는 URI 값은 다음과 같습니다.

URI	설명
http://schemas.xmlsoap.org/ws/2005/02/trust/Issue	STS는 DFHWS-IDTOKEN 컨테이너에서 전송되는 토큰을 대신할 토큰을 발행합니다.
http://schemas.xmlsoap.org/ws/2005/02/trust/Validate	STS는 DFHWS-IDTOKEN 컨테이너에서 전송되는 토큰의 유효성을 검증합니다.

이 컨테이너는 신뢰 클라이언트 인터페이스의 채널 DFHWSTC-V1에만 사용하십시오.

DFHWS-STSFault 컨테이너

DFHWS-STSFault는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 보안 토큰 서비스(STS)에서 리턴된 오류를 포함합니다.

오류가 발생하면 STS가 SOAP 결함을 발행합니다. SOAP 결함 콘텐츠는 이 컨테이너에서 리턴됩니다.

이 컨테이너는 신뢰 클라이언트 인터페이스를 사용하는 경우 채널 DFHWSTC-V1에 리턴됩니다.

DFHWS-STReason 컨테이너

DFHWS-STReason은 DATATYPE(CHAR) 컨테이너입니다. 이 요소가 보안 토큰 서비스(STS)의 응답 메시지에 있는 경우 이 컨테이너에는 <wst:Reason> 요소의 콘텐츠가 포함됩니다.

<wst:Reason> 요소에는 CICS가 STS로 보낸 유효성 검증 요청의 상태와 관련된 정보를 제공하는 선택적 문자열이 포함됩니다. 보안 토큰이 올바르지 않으면 이 요소에서 STS가 제공하는 정보로 토큰이 올바른 이유를 판별할 수 있습니다.

자세한 정보는 *WSTL(Web Services Trust Language)* 스펙(<http://www.ibm.com/developerworks/library/specification/ws-trust/>)을 참조하십시오.

DFHWS-STSURI 컨테이너

DFHWS-STSURI는 DATATYPE(CHAR) 컨테이너입니다. SOAP 메시지의 ID 토큰을 발행하거나 유효성을 검증하는 데 사용되는 보안 토큰 서비스(STS)의 절대 URI를 포함합니다.

URI 형식은 `http://www.example.com:8080/TrustServer/SecurityTokenService`입니다. 보안 요구사항에 따라 HTTP 또는 HTTPS를 사용할 수 있습니다.

이 컨테이너는 신뢰 클라이언트 인터페이스의 채널 DFHWSTC-V1에만 사용하십시오.

DFHWS-TOKENTYPE 컨테이너

DFHWS-TOKENTYPE은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 보안 토큰 서비스(STS)가 SOAP 메시지의 ID 토큰으로 발행하는 요청된 토큰 유형의 URI를 포함합니다.

올바른 토큰 유형을 지정할 수 있지만 STS가 해당 유형을 지원해야 합니다.

이 컨테이너는 신뢰 클라이언트 인터페이스의 채널 DFHWSTC-V1에만 사용하십시오.

SAML 지원 컨테이너

CICS SAML 지원이 사용하는 읽기 전용 컨테이너입니다.

다음 주제에서 *nnn*은 컨테이너가 여러 개 존재할 수 있음을 의미합니다. 컨테이너는 001에서 *nnn*(리턴된 이 유형의 컨테이너 수)까지 번호가 지정됩니다. 999개보다 많은 특정 유형의 컨테이너는 지원되지 않으며 SAML 어설션에서 컨테이너와 관련된 데이터는 무시됩니다. DSECT에 맵핑되지 않은 컨테이너는 길이가 가변적입니다.

DFHSAML-AnnnVmmm 컨테이너

DFHSAML-AnnnVmmm은 DATATYPE(CHAR) 컨테이너입니다. 속성 *nnn*의 속성 값 *mmm*을 포함합니다. 여기서 *nnn*과 *mmm*은 3자리 숫자입니다.

속성 수는 DFHSAML-COUNTS 컨테이너의 SAMLC-ATTRNUM입니다.

이 속성의 값 수는 DFHSAML-ATTRAnnn에 있습니다.

DFHSAML-ASSQNAME 컨테이너

DFHSAML-ASSQNAME은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 SAML 어설션 네임스페이스를 포함합니다.

가능한 값은 다음과 같습니다.

SAML 1.1

urn:oasis:names:tc:SAML:1.0:assertion

SAML 2.0

urn:oasis:names:tc:SAML:2.0:assertion

이 어설션은 URI여야 합니다. 어설션이 보다 복잡하면 세 파트로 추출됩니다.

DFHSAML-ATTRAnnn 컨테이너

DFHSAML-ATTRAnnn은 DATATYPE(BIT) 컨테이너입니다. 이 컨테이너는 속성 *nnn*의 값 수와 BIN(31) 필드를 포함합니다. 최대 값 수는 999입니다.

속성 수는 DFHSAML-COUNTS 컨테이너의 SAMLC-ATTRNUM입니다.

DFHSAML-ATTRFnnn 컨테이너

DFHSAML-ATTRFnnn은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 속성 *nnn*의 속성 이름 형식을 포함합니다. 여기서 *nnn*은 3자리 숫자입니다.

속성 수는 DFHSAML-COUNTS 컨테이너의 SAMLC-ATTRNUM입니다.

DFHSAML-ATTRNnnn 컨테이너

DFHSAML-ATTRNnnn은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 속성 *nnn*의 속성 이름을 포함합니다. 여기서 *nnn*은 3자리 숫자입니다.

속성 수는 DFHSAML-COUNTS 컨테이너의 SAMLC-ATTRNUM입니다.

DFHSAML-ATTRSnnn 컨테이너

DFHSAML-ATTRSnnn은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 속성 *nnn*의 속성 네임스페이스를 포함합니다. 여기서 *nnn*은 3자리 숫자입니다.

속성 수는 DFHSAML-COUNTS 컨테이너의 SAMLC-ATTRNUM입니다.

DFHSAML-ATTRYnnn 컨테이너

DFHSAML-ATTRYnnn은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 속성 *nnn*의 속성 친화적 이름을 포함합니다. 여기서 *nnn*은 3자리 숫자입니다.

속성 수는 DFHSAML-COUNTS 컨테이너의 SAMLC-ATTRNUM입니다.

DFHSAML-AUDNRnnn 컨테이너

DFHSAML-AUDNRnnn은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 AudienceRestriction 이름을 포함합니다.

리턴된 컨테이너의 수는 AUDNRNUM입니다.

DFHSAML-AUTHMETH 컨테이너

DFHSAML-AUTHMETH는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 토큰 홀더를 인증하는 데 사용되는 메소드를 포함합니다.

메소드에는 password, Kerberos, ltpa가 포함됩니다.

DFHSAML-CERTIDN 컨테이너

DFHSAML-CERTIDN은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 SAML 서명자 X.509 인증서의 발행자 식별 이름을 포함합니다.

DFHSAML-CERTSDN 컨테이너

DFHSAML-CERTSDN은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 SAML 서명자 X.509 인증서의 주제 식별 이름을 포함합니다.

DFHSAML-CERTSNUM 컨테이너

DFHSAML-CERTSNUM은 DATATYPE(CHAR) 컨테이너입니다. SAML 서명자의 X.509 인증서 일련 번호를 포함하는 8자 필드입니다.

DFHSAML-CONFMETH 컨테이너

DFHSAML-CONFMETH는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 이 SAML 토큰에서 사용되는 SubjectConfirmation 메소드를 포함합니다.

올바른 메소드는 holder-of-key, bearer 또는 sender-vouches입니다. 리턴된 문자열은 OASIS SAML 토큰 프로파일 1.1 및 2.0을 기반으로 합니다.

참고: 확인 메소드가 여러 개인 SAML 토큰은 지원되지 않습니다. 확인 메소드가 여러 개인 경우에는 결과를 예측할 수 없습니다.

DFHSAML-COUNTS 컨테이너

DFHSAML-COUNTS는 DATATYPE(BIT) 컨테이너입니다. 이 컨테이너는 변수 길이 컨테이너 출력의 수를 포함합니다.

DFHSAML-FLAGS 컨테이너

DFHSAML-FLAGS는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 플래그 바이트의 컬렉션을 포함합니다.

DFHSAML-ISSUER 컨테이너

DFHSAML-ISSUER은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 발행자의 이름을 포함합니다.

DFHSAML-NAMID 컨테이너

DFHSAML-NAMID는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 이름 형식 특성의 값을 포함합니다.

DFHSAML-NAMIDF 컨테이너

DFHSAML-NAMIDF는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 문자열 기반 ID 정보의 분류를 나타내는 URI 참조를 포함합니다.

DFHSAML-NAMIDQ 컨테이너

DFHSAML-NAMIDQ는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 이름을 규정하는 보안 또는 관리 도메인을 포함합니다.

DFHSAML-NAMIDSP 컨테이너

DFHSAML-NAMIDSP는 DATATYPE(CHAR) 컨테이너입니다. 엔티티의 서비스 제공자 또는 제공자 그룹이 설정한 이름 ID를 포함합니다.

DFHSAML-NAMIDSPQ 컨테이너

DFHSAML-NAMIDSPQ는 DATATYPE(CHAR) 컨테이너입니다. 서비스 제공자 또는 제공자 그룹의 이름을 포함합니다.

DFHSAML-OUTTOKEN 컨테이너

DFHSAML-OUTTOKEN은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 SAML 토큰을 포함합니다.

입력 컨테이너인 경우 이전에 유효성을 검증하고 다른 서비스 제공자로 라우팅되거나 확장된 후 라우팅되는 토큰을 포함합니다.

출력 컨테이너인 경우 DFHSAML 처리의 SAML 토큰 출력을 포함합니다. 유효성 검증 또는 추출 처리인 경우 이 토큰은 유효성을 검증하거나 추출되거나 수정되고 취소된 토큰입니다.

DFHSAML-PROXYnnn 컨테이너

DFHSAML-PROXYnnn은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 ProxyRestriction Audience 이름을 포함합니다.

DFHSAML-RESPONSE 컨테이너

DFHSAML-RESPONSE는 DATATYPE(BIT) 컨테이너입니다. 이 컨테이너는 내부에서 사용되는 응답 코드를 포함합니다.

DFHSAML-SAMLID 컨테이너

DFHSAML-SAMLID는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 ID(SAML 2.0) 또는 AssertionID(SAML 1.1)를 나타내는 문자열을 포함합니다.

DFHSAML-SUBJADDR 컨테이너

DFHSAML-SUBJADDR은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 SubjectLocality의 IP 주소를 포함합니다.

제한사항: SAML 2.0의 경우에는 이 컨테이너가 리턴되지 않습니다.

DFHSAML-SUBJDNS 컨테이너

DFHSAML-SUBJDNS는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 SubjectLocality의 DNSAddress를 포함합니다.

DFHSAML-TIMES 컨테이너

DFHSAML-TIMES는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 시간 값의 컬렉션을 포함합니다.

CICS에서 생성된 컨테이너

CICS는 가변 배열 및 긴 문자열과 같은 데이터를 저장할 컨테이너를 생성합니다. 이 컨테이너는 파이프라인 처리 중에 작성되며 애플리케이션 프로그램에 대한 입력 또는 출력으로 사용됩니다. 이 컨테이너에는 DFH 접두부가 추가됩니다.

이 컨테이너에 대한 이름 지정 규칙은 컨테이너를 작성하고 숫자 접미부와 결합되어 요청에서 컨테이너 이름을 고유하게 만드는 CICS 모듈을 사용하는 것입니다. 이 컨테이너 이름은 파이프라인 처리 중에 나타납니다.

DFHPIAXIS-nnnnnnnn

Axis2 파이프라인에서 애플리케이션으로 전달되는 문자열과 가변 배열을 저장하는 데 사용되는 컨테이너. 이 컨테이너에는 또한 2진 데이터가 포함될 수 있습니다.

DFHPICC-nnnnnnnnnn

애플리케이션으로 전달되는 문자열과 가변 배열을 저장하는 데 사용되는 컨테이너. 이 컨테이너에는 또한 2진 데이터가 포함될 수 있습니다.

DFHPIII-nnnnnnnnnn

파이프라인이 MTOM 메시지 핸들러와 함께 사용되고 직접 모드에서 실행될 때 작성되는 아웃바운드 첨부 파일 컨테이너입니다. 이 컨테이너는 애플리케이션 프로그램이 컨테이너가 아닌 필드에 2진 데이터를 제공할 때 작성됩니다.

DFHPIMM-nnnnnnnnnn

MIME 메시지 처리 중에 작성되는 인바운드 첨부 파일 컨테이너. 이 컨테이너는 MTOM 메시지 핸들러를 파이프라인에서 사용할 때 CICS로 생성됩니다. 직접 모드 처리를 사용하는 경우 이 컨테이너가 애플리케이션으로 직접 전달될 수 있습니다.

DFHPIXO-nnnnnnnnnn

파이프라인이 MTOM 메시지 핸들러와 함께 사용되고 호환 모드에서 실행될 때 작성되는 아웃바운드 첨부 파일 컨테이너입니다.

번호가 매겨진 컨테이너 이름은 웹 서비스 요청마다 1부터 시작됩니다(예를 들어, DFHPICC-00000001). 그러나 애플리케이션 프로그램이 **INVOKE SERVICE** 명령을 사용하여 동일한 채널에서 여러 웹 서비스 요청을 시작하는 경우에는 추가 요청 시 하나의 응답에 대해 애플리케이션으로 리턴된 컨테이너가 계속 존재합니다. 이러한 경우 CICS는 컨테이너가 이미 있는지 여부를 확인하며 이름 충돌을 방지하기 위해 생성된 컨테이너 수를 늘립니다.

사용자 컨테이너

이 컨테이너는 특정 메시지 핸들러가 다른 메시지 핸들러에 전달해야 하는 정보를 포함합니다. 사용자 컨테이너 사용은 전적으로 메시지 핸들러의 문제입니다. 이 컨테이너의 고유한 이름을 선택할 수 있지만 DFH로 시작하는 이름을 사용해서는 안 됩니다.

웹 서비스를 위한 런타임 처리

웹 서비스 제공자에게 요청을 전달하거나 웹 서비스 요청자로부터 요청을 수신하려면 사용자 애플리케이션(또는 랩퍼 프로그램)이 CICS의 웹 서비스 지원과 올바르게 상호작용해야 합니다. 인바운드 및 아웃바운드 요청이 핸들링되는 방법을 결정하기 위해 파이프라인에서 발생하는 처리를 제어할 수도 있습니다.

CICS가 웹 서비스 지원으로 배치되는 서비스 제공자 프로그램을 호출하는 방법

CICS 웹 서비스 지원을 사용하여 배치되는 서비스 제공자 애플리케이션이 호출될 때, CICS가 COMMAREA 또는 채널로 링크됩니다.

PGMINT 매개변수로 JCL 프로시저 DFHWS2LS 또는 DFHLS2WS를 실행할 때 사용되는 인터페이스의 종류를 지정합니다. 채널을 지정하면 **CONTID** 매개변수에서 컨테이너의 이름을 지정할 수 있습니다.

- COMMAREA 인터페이스로 프로그램이 호출되면, COMMAREA에는 SOAP 요청으로부터 CICS가 작성한 최상위 레벨 데이터 구조가 포함됩니다.
- 프로그램이 채널 인터페이스로 호출되면, DFHWS2LS 또는 DFHLS2WS의 **CONTID** 매개변수에 지정된 컨테이너의 프로그램으로 최상위 레벨 데이터 구조가 전달됩니다. **CONTID** 매개변수를 지정하지 않으면, 데이터가 컨테이너 DFHWS-DATA에서 전달됩니다. 채널 인터페이스는 다양한 수의 요소를 가진 배열을 지원하고, 이는 일련의 컨테이너에서 일련의 연결된 데이터 구조로 표시됩니다. 이러한 컨테이너도 존재합니다.

컨테이너에 대해 작업하기 위해 API 명령을 코드화할 때, 모든 컨테이너가 현재 채널(프로그램으로 전달된 채널)과 연관되므로 CHANNEL 옵션을 지정할 필요가 없습니다. 채널의 이름을 알아야 하는 경우, **EXEC CICS ASSIGN CHANNEL** 명령을 사용하십시오.

프로그램이 요청을 처리할 때, 이는 응답을 리턴하기 위해 동일한 메커니즘을 사용해야 합니다. 요청이 COMMAREA에 수신되면 응답은 COMMAREA에서 리턴되어야 합니다. 요청이 컨테이너에 수신되면 응답은 동일한 컨테이너에서 리턴되어야 합니다.

애플리케이션 프로그램이 응답 메시지를 실행할 때 오류가 발생하면 CICS는 애플리케이션이 동기점을 수행하지 않는 경우 모든 변경사항을 롤백합니다.

사용자 프로그램이 제공하는 웹 서비스가 응답을 리턴하도록 설계되지 않은 경우, CICS는 프로그램이 리턴할 때 COMMAREA 또는 컨테이너에 있는 것을 무시합니다.

웹 서비스 지원으로 배치되는 애플리케이션으로부터 웹 서비스 호출

웹 서비스 지원으로 배치되는 서비스 요청자 애플리케이션은 **EXEC CICS INVOKE SERVICE** 명령을 사용하여 웹 서비스를 호출합니다. 요청 및 응답이 컨테이너 DFHWS-DATA에서 데이터 구조로 맵핑됩니다. 이 서비스 호출 방법은 JSON에 대해 지원되지 않습니다.

프로시저

1. 채널을 작성하고 이를 컨테이너로 채우십시오. 최소한도로 컨테이너 DFHWS-DATA가 존재해야 합니다. DFHWS-DATA는 CICS가 SOAP 요청으로 변환하는 상위 레벨 데이터 구조를 보유합니다. SOAP 요청에 다수의 요소를 가지는 배열이 포함되는 경우, 일련의 컨테이너에서 일련의 연결된 데이터 구조로 표시됩니다. 이러한 컨테이너는 채널에 존재해야 할 수도 있습니다.
2. 대상 웹 서비스를 호출하십시오. 다음 명령을 사용하십시오.

```
EXEC CICS INVOKE SERVICE(webservice)  
                        CHANNEL(userchannel)  
                        OPERATION(operation)
```

여기서,

- *webservice*는 호출될 웹 서비스를 정의하는 WEBSERVICE 자원의 이름입니다. WEBSERVICE 자원은 웹 서비스와 통신할 때 CICS가 사용하는 웹 서비스 바인딩 파일 및 웹 서비스 설명의 위치를 지정합니다.
- *userchannel*은 애플리케이션의 데이터 구조와 연관되는 기타 컨테이너 및 컨테이너 DFHWS-DATA를 보유하는 채널입니다.
- *operation*은 대상 웹 서비스에서 호출되는 조작의 이름입니다.

자세한 정보는 381 페이지의 『웹 서비스에 대한 로컬 최적화』의 내용을 참조하십시오.

3. 명령이 성공적이면 채널에서 응답 컨테이너를 검색하십시오. 최소한도로 컨테이너 DFHWS-DATA가 존재합니다. SOAP 응답으로부터 CICS가 작성한 상위 레벨 데

이터 구조를 보유합니다. 응답에 다수의 요소를 가지는 배열이 포함되는 경우, 일련의 컨테이너에서 일련의 연결된 데이터 구조로 표시됩니다. 이러한 컨테이너는 채널에 존재합니다.

4. 서비스 요청자가 호출된 웹 서비스로부터 SOAP 결함 메시지를 수신하는 경우, 애플리케이션 프로그램이 변경사항을 롤백해야 하는지 여부를 판별해야 합니다. SOAP 결함이 발생하면 6의 RESP2 값을 가진 INVREQ 오류가 애플리케이션 프로그램으로 리턴됩니다. 그러나, 최적화가 실시되면 동일한 트랜잭션이 요청자와 제공자 모두에서 사용됩니다. 오류가 로컬로 최적화된 웹 서비스 제공자에 발생하면 트랜잭션에 의해 수행되는 모든 작업은 제공자와 요청자 모두에서 롤백합니다. INVREQ 오류는 16의 RESP2 값을 가진 요청자로 리턴됩니다.

웹 서비스에 대한 로컬 최적화

WEBSERVICE 자원과 연관되는 웹 서비스 바인딩 파일에서 제공자 애플리케이션 이름을 사용하여 웹 서비스 요청의 로컬 최적화를 사용할 수 있습니다.

INVOKE SERVICE 명령을 사용하여 URIMAP(urimap) 또는 URI(uri)를 지정할 수 있습니다. 여기서, URI는 호출할 웹 서비스의 URI입니다. URIMAP이 지정되면, CICS는 URI를 해결하기 위해 표시된 클라이언트 모드 URIMAP를 사용합니다. 이러한 옵션이 생략되면, CICS는 WEBSERVICE가 생성된 웹 서비스 설명(WSDL)에 지정된 URI를 사용합니다.

표시된 WEBSERVICE가 요청자 모드 PIPELINE에 배치되는 경우, CICS는 원격 웹 서비스를 호출합니다. 이는 가장 일반적인 시나리오입니다.

표시된 WEBSERVICE가 제공자 모드 PIPELINE에 배치되는 경우, CICS는 서비스를 로컬로 호출합니다. 이 최적화를 사용하는 경우, EXEC CICS INVOKE SERVICE 명령이 EXEC CICS LINK 명령에 최적화됩니다. 이는 결과적으로 중요한 성능 이점이 되지만, 다음 제한사항을 소개합니다.

- PIPELINE이 구동되지 않으므로 핸들러 프로그램이 사용되지 않습니다.
- PIPELINE 제어 컨테이너는 채널에 존재하지 않습니다. DFHWS-DATA, DFHWS-OPERATION 및 DFHWS-URI 컨테이너를 포함하여 일부 컨테이너가 존재합니다. 정상적으로 XML을 포함하는 컨테이너가 존재하지 않습니다. 이는 DFH-REQUEST, DFHWS-BODY 및 DFHWS-XMLNS 컨테이너를 포함합니다.
- 제공자 및 요청자 애플리케이션 모두 동일한 카피북을 공유하고 동일한 프로그래밍 언어에 구현되어야 합니다.
- 제공자 및 요청자 애플리케이션 모두는 단일 작업 단위를 공유합니다.
- 웹 서비스가 응답을 리턴할 것으로 예상되지 않으면, EXEC CICS INVOKE SERVICE 명령은 대상 PROGRAM이 종료된 후에 애플리케이션에 제어를 리턴합니다.

최적화된 웹 서비스를 로컬로 사용하지만 PIPELINE을 통해 데이터가 처리되도록 하려는 경우, 386 페이지의 『요청자 파이프라인 처리 제어를 위한 옵션』에 설명된 cics URI 형식을 사용하십시오. 이 메커니즘은 전체적으로 최적화된 접근법을 사용하는 것 보다 덜 효율적이지만, 이는 네트워크로 향하는 처리 비용을 피합니다.

웹 서비스 자원에 의해 생성되는 코드에 대한 런타임 제한사항

런타임 시 CICS는 동등한 데이터 구조에서 웹 서비스 설명(WSDL)을 따르는 유효한 SOAP 메시지를 거의 변형시킬 수 있습니다. 그러나, 웹 서비스 지원 일괄처리 작업을 사용하여 서비스 요청자 또는 서비스 제공자 애플리케이션을 개발할 시기를 인식해야 하는 일부 제한사항이 있습니다.

코드 페이지

CICS는 코드 페이지를 식별하는 적절한 HTTP 또는 WebSphere MQ 헤더가 있는 경우 코드 페이지에서 전송된 SOAP 메시지를 지원할 수 있습니다. CICS는 애플리케이션 프로그램에 의해 요청되는 코드 페이지에 전송하기 전에 파이프라인에서 이를 처리하도록 UTF-8로 SOAP 메시지를 변환합니다. 성능 영향을 최소화하려면 SOAP 메시지를 CICS에 전송할 때 UTF-8 코드 페이지를 사용하는 것이 추천됩니다. CICS는 UTF-8에서 SOAP 메시지를 항상 전송합니다.

애플리케이션 데이터가 EBCDIC 또는 UTF-16으로 인코딩된 경우에만 CICS가 SOAP 메시지를 변환할 수 있습니다. 데이터가 UTF-8, ASCII 및 ISO8859-1과 같은 코드 페이지에서 인코딩될 것으로 예상하는 애플리케이션은 지원되지 않습니다. 사용자 데이터 구조 및 SOAP 메시지 내에서 DBCS 문자를 사용하려는 경우, DBCS를 지원하는 코드 페이지를 지정해야 합니다. 사용자가 선택하는 EBCDIC 코드 페이지는 Java 및 z/OS 변환 서비스 모두에 의해 지원되어야 합니다. z/OS 변환 서비스는 SOAP 메시지의 코드 페이지에서 UTF-8로 변환하는 것을 지원하도록 구성되어야 합니다. UTF-16 지원에 대한 자세한 정보는 애플리케이션 데이터에서 UTF-16에 대한 지원의 내용을 참조하십시오.

적절한 코드 페이지를 설정하기 위해 웹 서비스 지원 작업에서 **LOCALCCSID** 시스템 초기화 매개변수 또는 선택적 **CCSID** 매개변수를 사용할 수 있습니다. **CCSID** 매개변수를 사용하는 경우, 지정하는 값은 해당 특정 웹 서비스에 대한 **LOCALCCSID** 코드 페이지를 대체합니다. **CCSID** 매개변수를 지정하지 않는 경우, **LOCALCCSID** 코드 페이지가 데이터를 변환하기 위해 사용되고 웹 서비스 바인딩 파일은 US EBCDIC(Cp037)에 인코딩됩니다.

컨테이너

서비스 제공자 모드에서 **PGMINT** 매개변수에 CHANNEL의 값이 있다는 것을 지정하는 경우, 애플리케이션 데이터를 보유하는 컨테이너는 2진 모드로 작성되어야 하고 읽어야 합니다. 이 컨테이너는 기본적으로 DFHWS-DATA입니다. **PUT CONTAINER** 명령은

DATATYPE 옵션이 BIT로 설정되도록 하거나 BIT가 기본으로 남아 있도록 FROMCCSID 옵션을 생략해야 합니다. 예를 들어, 다음 코드는 현재 채널의 CUSTOMER-RECORD 컨테이너에 있는 데이터가 2진 모드로 작성되어야 한다는 것을 명백하게 언급합니다.

```
EXEC CICS PUT CONTAINER (CUSTOMER-RECORD)
      FROM (CREC)
      DATATYPE(BIT)
```

비록 컨테이너 자체가 모두 BIT 모드에 있을지라도, 이 데이터를 맵핑하는 언어 구조 내에 텍스트 필드는 **LOCALCCSID** 또는 **CCSID** 매개변수에 지정한 것과 동일한 코드 페이지인 EBCDIC 코드 페이지를 사용해야 합니다. DFHWS2LS를 사용하여 웹 서비스 바인딩 파일을 생성하는 경우, 완료 데이터 구조의 파트를 보유하는 채널에서 많은 컨테이너가 있을 수 있습니다. 이런 경우에는 각각의 해당 컨테이너에 있는 텍스트 필드를 동일한 코드 페이지 사용에 대해 작성하고 읽어야 합니다.

애플리케이션 프로그램이 SOAP 메시지로 변환될 컨테이너를 채우는 경우, 애플리케이션은 컨테이너에 정확한 양의 콘텐츠가 있다는 것을 확인할 책임이 있습니다. 컨테이너가 예상한 것보다 더 적은 데이터를 보유하는 경우, CICS는 변환 오류를 실행합니다.

애플리케이션 프로그램이 **INVOKE SERVICE** 명령을 사용하는 경우, CICS로 전달하는 컨테이너가 잠재적으로 재사용되고 컨테이너 내의 데이터가 대체됩니다. 이러한 컨테이너에서 데이터를 유지하려는 경우, 새 채널을 작성하고 프로그램을 실행하기 전에 컨테이너를 복사하십시오. 또한 요청자 모드 웹 서비스인 제공자 모드 웹 서비스를 가지는 경우, 원래 첨부된 기본 채널을 사용하는 것보다 **INVOKE SERVICE** 명령을 사용할 때 다른 채널을 사용하는 것이 권장됩니다. 사용자 애플리케이션 프로그램이 **INVOKE SERVICE** 명령을 여러 번 사용 중인 경우, CICS에 대한 각 호출에서 다른 채널을 사용하거나 첫 번째 요청의 중요한 모든 데이터가 두 번째 요청을 하기 전에 저장되는지 확인하는 것이 권장됩니다.

웹 서비스 설명 준수

웹 서비스 설명은 선택적으로 SOAP 메시지의 가능한 일부 콘텐츠에 대해 설명할 수 있습니다. 이 경우, DFHWS2LS는 생성된 언어 구조 내의 필드를 콘텐츠가 존재하는지 여부를 표시하기 위해 할당합니다. 따라서, 런타임 시 CICS는 이러한 필드를 채웁니다. 정보가 존재하지 않는다는 것을 필드(예: 존재 플래그 또는 발생 필드)가 표시하는 경우, 애플리케이션 프로그램은 해당 선택 콘텐츠와 연관된 필드를 처리하려고 시도하지 않아야 합니다.

CICS가 이를 전송할 때 SOAP 메시지가 일부 해당 콘텐츠를 누락하는 경우, 애플리케이션 프로그램으로 전달될 때 데이터 구조 내의 동등한 필드가 초기화되지 않습니다.

웹 서비스 설명은 SOAP 메시지를 읽고 CICS가 런타임 시 이러한 규칙을 구현할 때 사용할 공백 처리 규칙을 지정할 수도 있습니다.

- `xsd:whiteSpace` 패킷 값이 `replace`인 경우, 공백 문자(예: 『tab』 및 『carriage return』)가 공백으로 대체됩니다.
- `xsd:whiteSpace` 패킷 값이 `collapse`인 경우, SOAP 메시지를 생성할 때 후미 공백 문자가 제거됩니다. 런타임 시, 인바운드 SOAP 메시지는 XML 스키마 스펙에 따라 구문 분석되고 모든 선두, 후미 및 임베디드 공백이 제거됩니다.

SOAP 메시지

CICS는 SOAP 메시지 콘텐츠 도출을 지원하지 않습니다. 예를 들어, SOAP 메시지는 요소를 설명하는 스키마의 위치를 지정하기 위한 `xsi:schemaLocation` 속성과 함께 `xsi:type` 속성을 사용하여 요소가 특정 유형을 가지도록 지정할 수 있습니다. CICS는 스키마의 콘텐츠를 기반으로 요소 값을 전송하고 스키마를 동적으로 검색하는 기능을 지원하지 않습니다. CICS는 웹 서비스 지원에 설정되는 맵핑 레벨이 1.1 이상이지만, 지원되는 유일한 XML 스키마 인스턴스 속성일 때 `xsi:nil` 속성을 지원합니다.

DFHWS2LS는 SOAP 메시지에서 일부 값의 크기 또는 최대 길이에 대해 가정해야 합니다. 예를 들어, XML 스키마가 `xsd:string`에 대한 최대 길이를 지정하지 않으면 DFHWS2LS는 최대 길이가 255자이고 따라서 언어 구조를 생성한다고 가정합니다. DFHWS2LS에서 **DEFAULT-CHAR-MAXLENGTH** 매개변수를 사용하여 이 값을 변경할 수 있습니다. 런타임 시, CICS가 언어 구조에서 할당되는 공간 보다 큰 값을 가진 SOAP 메시지를 생성하는 경우, CICS가 변환 오류를 실행합니다.

CICS가 서비스 제공자인 경우 SOAP 결함 메시지가 요청자로 리턴됩니다. CICS가 서비스 요청자인 경우, 적당한 RESP2 코드가 **INVOKE SERVICE** 명령으로부터 리턴됩니다.

일부 문자는 < 및 > 문자와 같은 XML에서 특수 의미를 갖습니다. 런타임 시 CICS가 처리하는 문자 배열 내에 이러한 특수 문자가 표시되면 이는 동등한 엔티티로 대체됩니다. 지원되는 XML 엔티티는 다음과 같습니다.

문자	XML 엔티티
&	&
<	<
>	>
"	"
'	'

CICS는 공백 코드에 사용되는 숫자 문자 참조의 정규형을 지원합니다.

문자	XML 엔티티
탭		
캐리지 리턴	

줄 바꾸기	

이 지원은 호출되는 파이프라인 핸들러 프로그램으로 확장되지 않는다는 것을 참고하십시오.

널 문자(x'00')는 XML 문서에서 올바르지 않습니다. 애플리케이션 프로그램이 제공하는 문자 유형 필드에 널 문자가 포함되는 경우, CICS는 해당 위치에서 데이터를 자릅니다. 이를 통해 널 종료 문자열로 문자 배열을 처리할 수 있습니다. base64Binary의 DFHWS2LS 또는 hexBinary XML 스키마 데이터 유형이 생성하는 문자 유형 필드는 2진 데이터를 나타내고 자르지 않고 널 문자를 포함할 수 있습니다.

주의: CICS는 구조화된 애플리케이션 데이터로부터 XML 및 JSON 데이터를 생성합니다. 해당 애플리케이션 데이터가 사전 형식화된 XML, JSON, HTML, JPEG 이미지처럼 보이는 비트 패턴 또는 실제로 기타 유의미한 콘텐츠 유형을 포함하는 경우, CICS가 이 시맨틱 의미를 인식하지 못하고 해당 데이터를 일반 텍스트 또는 2진 데이터처럼 처리합니다. CICS는 데이터 내에서 패턴을 인식하거나 인코딩된 데이터를 다르게 처리하려고 시도하지 않습니다. 예를 들어, 데이터가 사전 형식화된 XML(예: CDATA 인코딩 텍스트)을 포함하는 경우, 해당 데이터가 기타 데이터와 동일한 방식으로 처리됩니다. "An example: <here>"과 같은 애플리케이션 데이터가 있다고 가정해 보십시오. 이 예제 애플리케이션에서 제공하는 데이터는 XML 태그처럼 보이는 내용을 포함하나 원시 텍스트로 처리되어 "An example: < here >"과 같은 XML 표시가 발생합니다. 애플리케이션이 XML 자체를 생성해야 하는 경우, XML 스키마에 xsd:any 구조를 사용하거나 XML-ONLY=TRUE를 보조적으로 사용하는 방법을 고려하십시오.

SOAP 결함 메시지

CICS가 서비스 제공자이고 애플리케이션 프로그램이 SOAP 결함 메시지를 실행하도록 하려는 경우, **SOAPFAULT CREATE** 명령을 사용하십시오. 이 API 명령을 사용하려면 웹 서비스 지원 **PGMINT** 매개변수에 **CHANNEL**의 값을 갖도록 지정해야 합니다. 이 값을 지정하지 않고 애플리케이션 프로그램이 **SOAPFAULT CREATE** 명령을 호출하는 경우, CICS는 SOAP 응답 메시지를 생성하려고 시도하지 않습니다.

파이프라인 처리 사용자 정의

자체 메시지 핸들러 제공에 추가하여 일련의 글로벌 사용자 종료점(GLUE)을 사용하여 파이프라인에서 인바운드 및 아웃바운드 웹 서비스에 대해 발생하는 처리를 사용자 정의할 수 있습니다.

시작하기 전에

파이프라인을 사용자 정의하기 전에 글로벌 사용자 종료 프로그램을 작성하기 위한 우수 사례를 이해해야 합니다. 서비스 제공자 파이프라인을 사용자 정의하는 경우 파이프라인에서 제공된 DFHPITP 또는 Axis2 애플리케이션 핸들러를 사용하고 있어야 합니다.

이 태스크 정보

파이프라인 도메인 종료를 사용하여 웹 서비스 제공자 파이프라인, 웹 서비스 요청자 파이프라인 또는 보안 핸들러를 포함하는 웹 서비스 요청자 파이프라인의 컨테이너에 액세스할 수 있습니다. 파이프라인 글로벌 사용자 종료에 대한 세부사항은 *CICS 사용자 정의 안내서*에 설명되어 있습니다.

프로시저

1. 사용할 글로벌 사용자 종료점을 선택하십시오.
 - XWSPRRWI, XWSPRROI, XWSPRROO 또는 XWSPRRWO GLUE를 사용하여 웹 서비스 제공자 파이프라인의 컨테이너에 액세스합니다.
 - XWSRQRWO, XWSRQROO, XWSROROI 또는 XWSRQRWI GLUE를 사용하여 웹 서비스 요청자 파이프라인의 컨테이너에 액세스합니다.
 - XWSSRRWO, XWSSRROO, XWSSRROI 또는 XWSSRRWI GLUE를 사용하여 보안 웹 서비스 요청자 파이프라인의 컨테이너에 액세스합니다.
2. DFH\$PIEX 샘플 종료 프로그램을 사용하여 자체 글로벌 사용자 종료 프로그램을 작성하십시오. DFH\$PIEX는 SDFHSAMP 라이브러리에 있습니다. 프로그램 `threadsafe`의 작성이 권장됩니다.
3. 글로벌 사용자 종료 프로그램을 사용으로 설정하십시오.
4. 글로벌 사용자 종료 프로그램을 테스트하여 올바르게 작동하는지 확인하십시오.

관련 정보:

파이프라인 도메인 종료

파이프라인 샘플 종료 프로그램

글로벌 사용자 종료 프로그램 작성

종료 프로그램 정의, 사용, 사용 안함

요청자 파이프라인 처리 제어를 위한 옵션

서비스 요청자 파이프라인에서 메시지 핸들러는 웹 서비스 요청이 URI를 변경하여 전송되는 위치를 판별할 수 있습니다. CICS에서는 파이프라인 처리 웹 서비스가 요청하는 방식으로 더 많은 유연성을 가지도록 다른 URI 형식에 대한 지원을 제공합니다.

서비스 요청자 파이프라인이 해당 처리의 끝에 도달할 때 다음 옵션을 갖습니다.

프로그램에 링크

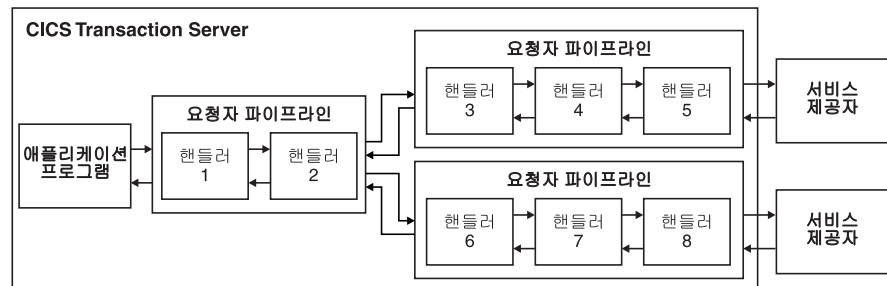
`cics://PROGRAM/program` 형식으로 URI를 변경하는 경우, 여기서 *program*은 대상 애플리케이션 프로그램의 이름이고 CICS는 현재 채널 및 해당 컨테이너 또는 COMMAREA를 **EXEC CICS LINK** 명령을 사용하여 프로그램에 전달합니다.

이 처리는 서비스 요청자 및 서비스 제공자 애플리케이션이 동일한 CICS 리전에 있을 때 발생하는 로컬 최적화와 유사합니다. 그러나, 이 URI 형식 사용은 파이프라인 및 사용자 정의 메시지 핸들러를 통해 먼저 요청을 실행하는 이점을 제공합니다. 대상 애플리케이션 프로그램은 컨테이너 또는 COMMAREA의 콘텐츠를 처리할 수 있어야 합니다.

다른 요청자 모드 파이프라인 시작

`cics://PIPELINE/pipeline?targetServiceUri=targetServiceUri` 형식으로 URI를 변경하는 경우, 여기서 *pipeline*은 PIPELINE 자원의 이름이고 *targetServiceUri*는 DFHWS-URI 컨테이너에 넣으려는 URI이고 CICS는 현재 채널 및 해당 컨테이너를 지정된 요청자 파이프라인에 전달합니다. 서비스 제공자에 요청을 전송하기 전에 둘 이상의 요청자 파이프라인에 링크하려고 할 때 이 URI를 사용하십시오. 함께 연결될 수 있는 요청자 파이프라인의 수는 제한되지 않습니다.

다음 예제에서 하나의 일반 요청자 파이프라인은 하나의 애플리케이션을 지원합니다. 메시지 핸들러 1 또는 2는 다른 메시지 핸들러를 포함하는 두 개의 요청자 파이프라인 중 하나에 요청을 전송하여 컨테이너에 있는 애플리케이션 데이터에 따라 각 요청에 대한 URI를 변경할 수 있습니다.



예제가 단지 하나의 서비스 요청자 애플리케이션을 표시한다고 해도, 많은 애플리케이션은 동일한 일반 요청 파이프라인을 사용하고 요청이 마침내 적합한 웹 서비스 제공자로 전송되기 전에 요청이 다른 요청자 파이프라인으로 전송되도록 할 수 있습니다.

제공자 모드 파이프라인에 일직선형으로 요청 보내기

URI를 `cics://SERVICE/service?targetServiceUri=targetServiceUri` 형식으로 변경하는 경우, 여기서 *service*는 대상 서비스의 이름이고 *targetServiceUri*는 서비스에 대한 대상이고 CICS는 URIMAP에 대한 경로를 일치하여 요청을 해결하고 요청을 올바른 제공자 파이프라인에 전달합니다. 네트워크를 사용하지 않고 요청자와 제공자 파이프라인 모두를 통해 요청을 처리하는 데 사용하려고 할 때 이 옵션을 사용하십시오.

이 URI는 요청자와 제공자 애플리케이션이 다른 언어로 작성되거나 다른 맵핑 레벨을 사용하고 다른 2진 데이터를 예상하는 위치에서 유용할 수도 있습니다.

관련 태스크:

『URI을 사용하여 요청자 파이프라인 처리 제어』

서비스 요청자 파이프라인에서 메시지 핸들러는 URI를 변경하여 웹 서비스 요청을 전송할 위치를 판별할 수 있습니다. URI 형식을 변경하여 특정 최적화 수행(예: 네트워크를 통한 요청 전송 없이 다른 요청자 파이프라인 시작 또는 서비스 제공자 파이프라인 시작)을 선택할 수 있습니다.

관련 참조:

361 페이지의 『DFHWS-URI 컨테이너』

DFHWS-URI는 서비스 URI를 포함하는 DATATYPE(CHAR) 컨테이너입니다.

URI을 사용하여 요청자 파이프라인 처리 제어

서비스 요청자 파이프라인에서 메시지 핸들러는 URI를 변경하여 웹 서비스 요청을 전송할 위치를 판별할 수 있습니다. URI 형식을 변경하여 특정 최적화 수행(예: 네트워크를 통한 요청 전송 없이 다른 요청자 파이프라인 시작 또는 서비스 제공자 파이프라인 시작)을 선택할 수 있습니다.

시작하기 전에

요청자 파이프라인에서 구현할 옵션을 결정하십시오. 세부사항은 386 페이지의 『요청자 파이프라인 처리 제어를 위한 옵션』의 내용을 참조하십시오.

이 태스크 정보

웹 서비스 요청자 애플리케이션은 **EXEC CICS INVOKE SERVICE** 명령을 사용하여 DFHWS-URI 컨테이너를 채울 수 있습니다. 애플리케이션에서 값을 제공하지 않으면 CICS가 웹 서비스 바인딩 파일의 값을 사용하여 컨테이너를 채웁니다. URI를 수정하려면 이 컨테이너의 콘텐츠를 변경하는 메시지 핸들러를 작성해야 합니다.

프로시저

1. 다음 URI 형식 중 하나에 따라 DFHWS-URI 컨테이너를 수정하도록 메시지 핸들러를 작성하십시오.
 - 애플리케이션 프로그램에 링크하려면 URI `cics://PROGRAM/program`를 사용하십시오. 여기서, *program*은 대상 애플리케이션 프로그램입니다. 데이터 변환이 발생하지 않습니다. 따라서 애플리케이션 프로그램이 현재 채널에서 컨테이너의 콘텐츠를 처리할 수 있는지 확인해야 합니다. 애플리케이션 프로그램은 현재 채널 및 컨테이너 또는 COMMAREA를 전달할 수 있습니다.
 - 네트워크를 거치지 않고 제공자 파이프라인을 시작하려면 URI `cics://SERVICE/service?targetServiceUri=targetServiceUri`을 사용하십시오. 여기서, *service*는 서비스의 이름이고 *targetServiceUri*는 서비스의 경로입니다. 전송 핸

들리는 서비스 경로를 사용하여 요청을 분석하고 올바른 제공자 파이프라인으로 전달하는 URIMAP 자원을 찾습니다. CICS는 처리 중 서비스의 이름을 사용하지 않습니다.

서비스에 URIMAP 자원이 설치되지 않은 경우 오류가 발생합니다. URIMAP 자원 정의는 USAGE(PIPELINE)도 지정해야 합니다. 전송 핸들러는 **targetServiceUri** 매개변수의 값을 DFHWS-URI 컨테이너에 넣고 제공자 파이프라인을 시작합니다.

- 다른 요청자 파이프라인을 시작하려면 URI `cics://PIPELINE/pipeline?targetServiceUri=targetServiceUri`를 사용하십시오. 여기서, *pipeline*은 시작할 PIPELINE 자원의 이름이고 *targetServiceUri*는 DFHWS-URI 컨테이너에서 다음 파이프라인에 전달할 값입니다.

각 유형의 URI에는 조회 문자열로 추가할 수 있는 추가 매개변수가 있습니다. 이러한 URI의 형식 및 코드화 규칙에 대한 자세한 정보는 361 페이지의 『DFHWS-URI 컨테이너』의 내용을 참조하십시오.

2. XML 편집기를 사용하여 파이프라인 구성 파일에 메시지 핸들러를 추가하십시오.

```
<service>
  <service_handler_list>
    <handler>
      <program>MYPROG</program>
    </handler>
  </service_handler_list>
</service>
```

3. 요청자 파이프라인의 PIPELINE 자원을 사용 안함으로 설정하고, 버리고, 재설치하여 새 메시지 핸들러 프로그램을 파이프라인에 포함시키십시오.
4. CICS 리전에서 메시지 핸들러 프로그램을 설치하십시오.

결과

요청자 파이프라인 전체를 실행하기 위한 다음 서비스 요청은 새 메시지 핸들러에 의해 처리됩니다.

다음에 수행할 작업

요청자 파이프라인에 대한 변경사항을 테스트하여 서비스 요청이 올바른 위치로 이동하는지 및 메시지 핸들러 프로그램이 디자인된 대로 작동하는지 확인하십시오.

관련 개념:

386 페이지의 『요청자 파이프라인 처리 제어를 위한 옵션』

서비스 요청자 파이프라인에서 메시지 핸들러는 웹 서비스 요청이 URI를 변경하여 전송되는 위치를 판별할 수 있습니다. CICS에서는 파이프라인 처리 웹 서비스가 요청하는 방식으로 더 많은 유연성을 가지도록 다른 URI 형식에 대한 지원을 제공합니다.

관련 참조:

JSON 웹 서비스 제한사항

이 참조 자료를 사용하여 JSON 웹 서비스에서 지원되지 않는 기능을 이해할 수 있습니다.

다음 기능은 지원되지 않습니다.

- JSON이 포함된 요청자 모드 파이프라인은 지원되지 않습니다.
- 스키마에 대한 JSON 데이터의 런타임 유효성 검증이 지원되지 않습니다. JSON 페이로드와 함께 사용되는 WEBSERVICE 자원의 VALIDATION 속성 값은 무시됩니다.
- JSON 데이터(Badgerfish 또는 맵핑된 규칙)에서 네임스페이스 사용은 지원되지 않습니다.
- CICS에 전송된 JSON 페이로드는 UTF-8로 인코딩해야 합니다. 다른 인코딩은 지원되지 않습니다. 마찬가지로, CICS에 의해 전송된 JSON은 항상 UTF-8로 인코딩됩니다.
- JSON 파이프라인을 사용한 WebSphere MQ 전송은 지원되지 않습니다.
- 공급업체 변환기 프로그램을 JSON 변환기에 사용하는 것은 지원되지 않습니다.
- SOAP 웹 서비스 애플리케이션을 위해 JSON 파이프라인에서 작성된 WSBind 파일의 재사용은 지원되지 않습니다. JSON 서비스 제공자 애플리케이션에서 사용할 WSBind 파일은 JSON 지원에 의해 생성되어야 합니다.
- CICS가 JSON 페이로드를 변환할 때 JSON 페이로드가 일부 필수 콘텐츠를 잃는 경우 애플리케이션 프로그램에 전달 시 데이터 구조 내 해당 필드가 초기화되지 않습니다.
- CICS는 따옴표로 묶인 정수 값 외에 부호 있는 긴 정수($2^{63} - 1$)의 최대값을 초과하는 정수 값을 변환할 수 없습니다.
- 단순 데이터 유형 또는 배열의 사용은 JSON 스키마의 루트에서 지원되지 않습니다. JSON 오브젝트는 단순 데이터 유형 및 배열로 구성할 수 있지만 JSON 오브젝트를 설명하는 데 JSON 스키마가 필요합니다.
- maxItems 값이 1인 JSON 스키마에서 배열이 선언되는 경우 런타임 시 JSON을 생성할 때 CICS는 이 배열을 단순 문자열 또는 정수로 직렬화합니다.

중요사항: JSON 특성 이름에 대해 지원되는 유일한 문자는 첫 번째 문자에 대해서는 A-Z a-z _ : 이고 모든 후속 문자에 대해서는 A-Z a-z 0-9 _ : . -입니다.

현재 Axis2 웹 서비스 지원에는 애플리케이션 및 사용자 정의의 개발 및 배치를 위한 다수의 옵션이 있습니다. 다음 옵션은 지원되지 않습니다.

- 사용자 제공 애플리케이션 핸들러 - CICS 제공 애플리케이션 핸들러 클래스 `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler`를 사용해야 합니다.
- 사용자 작성 Axis2 Java 애플리케이션
- SOAPFAULT 및 WS-Addressing API는 JSON 파이프라인에서 사용할 수 없습니다.

컨테이너 제한사항

JSON 요청을 처리할 때 일부 파이프라인 컨테이너는 채워지지 않음에 유의하십시오. 자세한 정보는 구성의 파이프라인에서 사용된 컨테이너의 내용을 참조하십시오.

RESTful 웹 서비스의 차이점

INQUIRE PIPELINE에서

- SOAPLEVEL은 NOTSOAP을 리턴함
- MTOMNOXOPST, MTOMST, SENDMTOMST, SOAPRNUM, SOAPVNUM, XOPDIRECTST 및 XOPSUPPORTST 속성은 사용되지 않습니다.

INQUIRE WEBSERVICE에서

- ARCHIVEFILE, BINDING, VALIDATIONST, XOPDIRECTST 및 XOPSUPPORTST 속성은 사용되지 않습니다.
- WSDLFILE은 WEBSERVICE와 연관된 JSON 스키마 파일의 이름을 리턴합니다.

WEBSERVICE 자원에서

- ARCHIVEFILE 및 VALIDATION 매개변수는 사용되지 않으며 해당 값은 무시됩니다.
- WSDLFILE은 WEBSERVICE와 연관된 JSON 스키마 파일의 이름입니다.

웹 서비스 트랜잭션 지원

웹 서비스 원자적 트랜잭션(또는 WS-AtomicTransaction) 스펙 및 웹 서비스 조정(또는 WS-Coordination) 스펙은 트랜잭션 처리 시스템이 웹 서비스 환경에서 상호 운용될 수 있도록 하는 단기 트랜잭션에 대한 프로토콜을 정의합니다. WS-AtomicTransaction을 사용하는 트랜잭션에는 ACID 특성인 원자성, 일관성, 격리 및 내구성이 있습니다.

스펙은 <http://www.ibm.com/developerworks/library/specification/ws-tx/>에서 찾을 수 있습니다.

참고: CICS는 스펙의 2004년 11월 레벨을 지원합니다.

웹 서비스 제공자 또는 요청자로 배치되는 CICS 애플리케이션은 이 스펙을 지원하는 다른 웹 서비스 구현과의 분배 트랜잭션에 참가할 수 있습니다.

등록 서비스

등록 서비스는 조정 프로토콜에 대해 애플리케이션을 등록할 수 있게 하는 WS-Coordination 모델의 해당 파트입니다. 분배 트랜잭션에서 참가 시스템의 등록 서비스는 서로 통신하여 연결된 애플리케이션이 해당 프로토콜에 참가할 수 있게 합니다.

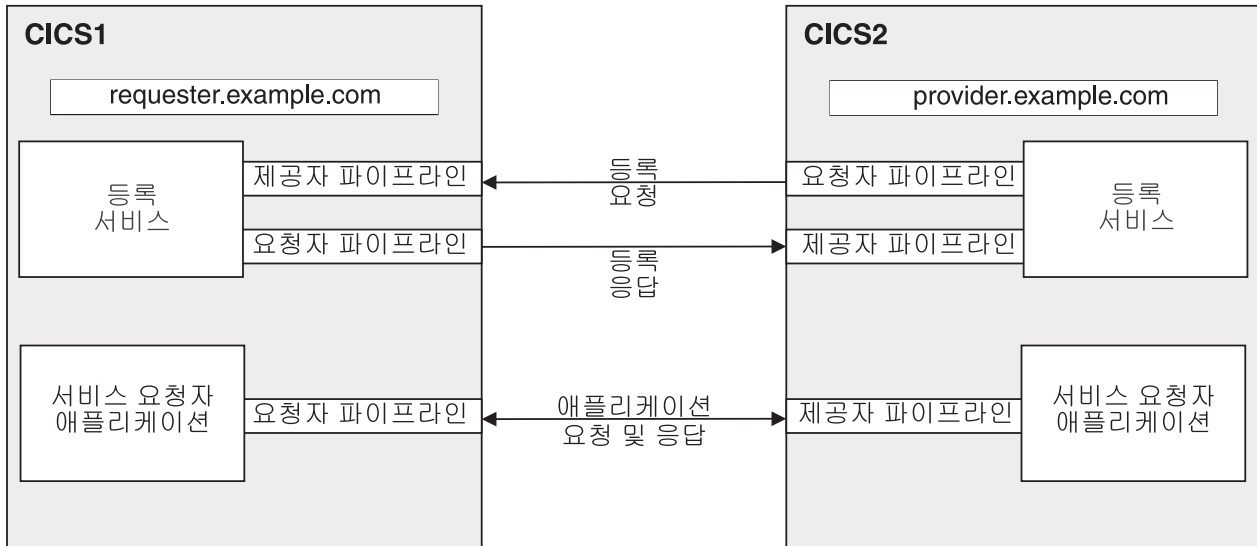


그림 29. 등록 서비스

그림 29에서는 두 CICS 시스템인 CICS1 및 CICS2를 보여줍니다. CICS1의 서비스 요청자 애플리케이션은 CICS2에서 서비스 제공자 애플리케이션을 호출합니다. 두 CICS 리전 및 애플리케이션은 두 애플리케이션이 WS-Coordination 프로토콜을 사용하여 단일 분배 트랜잭션에 참가할 수 있도록 구성됩니다. 서비스 요청자 애플리케이션은 조정자이고 서비스 제공자 애플리케이션은 참가자입니다.

이러한 프로토콜의 지원에서 두 CICS 리전의 등록 서비스는 트랜잭션 시작 시 상호 작용하고 트랜잭션 종료 중 다시 상호 작용합니다. 이러한 상호작용 중 두 리전의 등록 서비스는 서비스 제공자 및 요청자로 서로 다른 시간에 작동할 수 있습니다. 따라서 각 리전에서 등록 서비스는 서비스 제공자 파이프라인 및 서비스 요청자 파이프라인을 사용합니다. 파이프라인은 PIPELINE 및 연관된 자원을 사용하여 CICS에 정의됩니다.

각 리전의 등록 서비스는 엔드포인트 주소와 연관됩니다. 따라서 이 예제에서 CICS1의 등록 서비스 엔드포인트 주소는 requester.example.com이고 CICS2의 등록 서비스 엔드포인트 주소는 provider.example.com입니다.

CICSplex에서 등록 서비스 제공자 파이프라인을 다른 리전에 분배할 수 있습니다. 이는 393 페이지의 그림 30에 표시되어 있습니다.

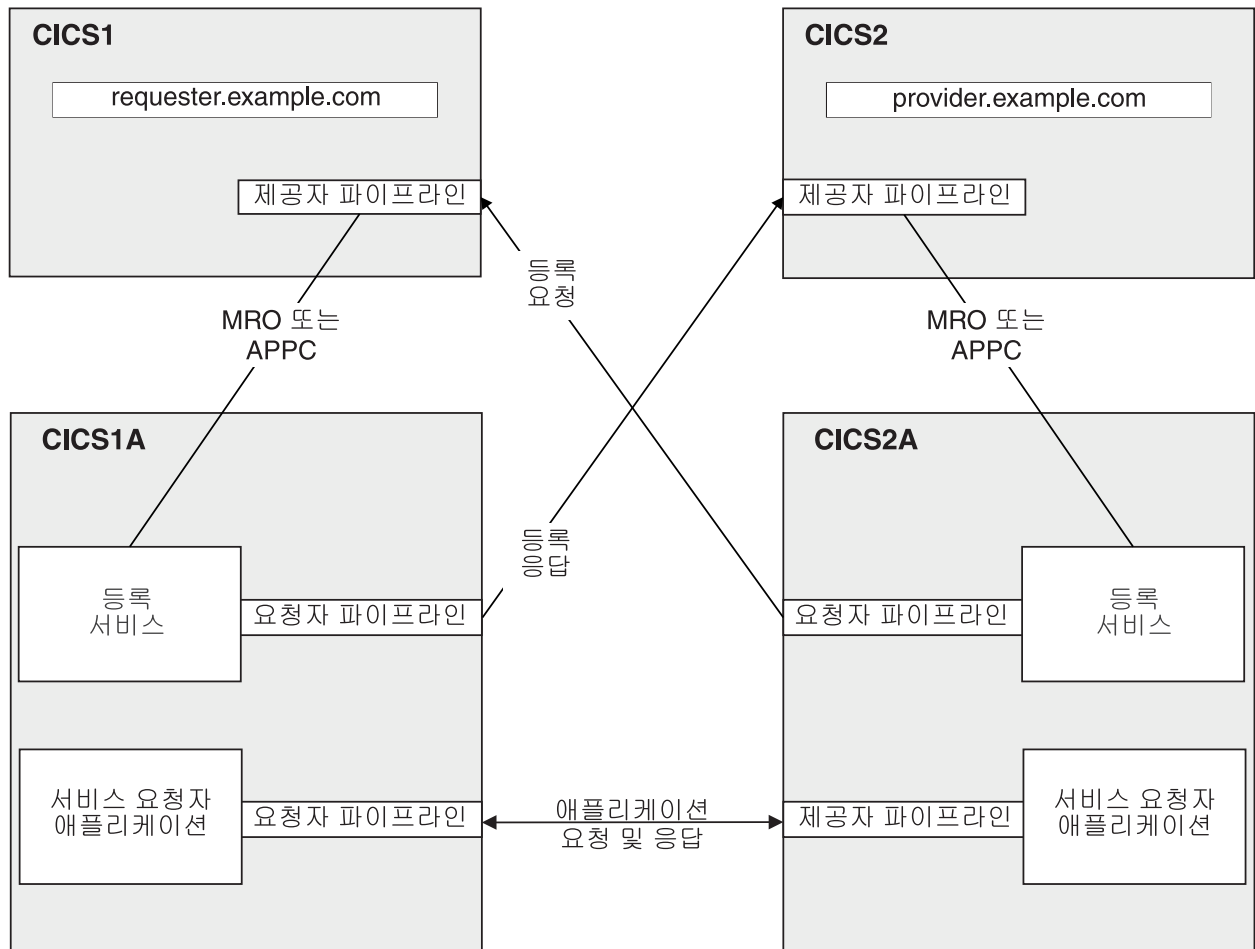


그림 30. CICSplex의 등록 서비스

이 구성에서 제공자 파이프라인은 MPO 또는 APPC를 사용하여 등록 서비스와 통신합니다. 등록 서비스 요청자 파이프라인은 애플리케이션의 요청자 파이프라인과 동일한 리전에 남아 있어야 합니다.

이 구성은 서비스 요청자 및 제공자 애플리케이션이 다수의 리전에 분배되는 경우에 유용합니다. 웹 서비스 트랜잭션에 참가하도록 애플리케이션의 파이프라인을 구성하는 경우 등록 서비스 제공자 파이프라인의 IP 주소 및 포트 번호를 제공하여 등록 서비스 엔드포인트에 대한 정보를 제공해야 합니다. 한 개의 엔드포인트를 가짐으로써 구성을 단순화할 수 있습니다. 모든 파이프라인에 같은 정보가 포함되기 때문입니다. 예를 들어, 그림 30에서 애플리케이션 요청자 파이프라인에서 지정하는 IP 주소는 `requester.example.com`입니다.

같은 인수가 서비스 제공자 애플리케이션에 적용됩니다. 예제에서 제공자 애플리케이션의 파이프라인은 `requester.example.com`의 IP 주소를 지정합니다.

웹 서비스 트랜잭션을 위한 CICS 구성

웹 서비스 트랜잭션에 웹 서비스 요청자 및 제공자 애플리케이션을 참가시키려면 그에 따라 여러 CICS 자원을 설치하여 CICS를 구성해야 합니다.

시작하기 전에

이러한 자원을 설치하려면 먼저 웹 서비스 트랜잭션의 지원에서 CICS가 제공하는 파이프라인 구성 파일의 위치를 알아야 합니다. 기본적으로 구성 파일은 `/usr/lpp/cicsts/cicsts53/pipeline/configs` 디렉토리에 제공되지만 기본 파일 경로는 CICS 설치 중 변경되었을 수도 있습니다.

이 태스크 정보

웹 서비스 트랜잭션을 위한 CICS 지원은 CICS 제공 등록 서비스를 사용합니다. 이 등록 서비스는 서비스 제공자 및 서비스 요청자로 구성됩니다. 애플리케이션이 모두 서비스 제공자이거나 서비스 요청자인 경우에도 서비스 제공자 및 서비스 요청자 둘 다에 대한 자원을 설치해야 합니다.

또한 서비스 제공자 및 요청자 애플리케이션을 실행할 때 호출되는 헤더 핸들러 프로그램의 프로그램 정의를 설치해야 합니다.

웹 서비스 트랜잭션을 위해 CICS를 구성하는 데 필요한 자원은 모두 DFHWSAT 그룹에서 제공되지만 DFHPIVS, DFHPIVR 또는 DFHPICF 그룹 중 하나에서 제공되는 DFHPIDIR은 제외됩니다. DFHWSAT 그룹은 DFHLIST 목록에 포함되지 않기 때문에 자동으로 설치되지 않습니다. DFHWSAT 그룹에서 CICS 제공 자원은 변경할 수 없습니다.

웹 서비스 트랜잭션을 위해 CICS를 구성하려면 다음을 수행하십시오.

프로시저

1. DFHPIDIR 데이터 세트를 추가하여 JCL을 시작하십시오. DFHPIDIR은 컨텍스트와 태스크 간의 맵핑을 저장합니다.
 - a. DFHPIDIR 데이터 세트의 새 DD 명령문을 CICS 시작 JCL에 추가하십시오.
 - b. DFHDEFDS.JCL의 정보를 사용하여 DFHPIDIR 데이터 세트를 작성하십시오. DFHPIDIR의 기본 RECORDSIZE는 1KB이며 이 값은 대부분의 경우에 적합합니다. 필요한 경우 더 큰 RECORDSIZE를 사용하여 DFHPIDIR을 작성할 수 있습니다.
 - c. CICS 시스템에 DFHPIVS, DFHPIVR 또는 DFHPICF 중 적합한 데이터 세트 그룹을 설치하십시오. 이러한 그룹에 대한 자세한 정보는 WS-AT 데이터 세트 정의를 참조하십시오.

CICS 리전 사이에서 DFHPIDIR 파일을 공유하려는 경우 리전은 MRO를 통해 논리적으로 연결되어 있어야 합니다. 논리 서버 역할을 하는 리전 그룹당 한 개의 데이터 세트를 설치해야 합니다.

팁: 논리적으로 연결되지 않은 리전 간 데이터 세트 공유는 권장되지 않습니다.

2. DFHWSAT 그룹의 콘텐츠를 다른 그룹에 복사하십시오. DFHWSAT 그룹에서 CICS 제공 자원은 변경할 수 없습니다. 그러나 PIPELINE 자원에서 CONFIGFILE 속성은 변경해야 합니다.
3. 등록 서비스의 제공자 PIPELINE 자원을 수정하십시오. PIPELINE은 DFHWSATP로 이름 지정되고 CONFIGFILE 속성에서 파이프라인 구성 파일 `/usr/lpp/cicsts/cicsts53/pipeline/configs/registrationservicePROV.xml`을 지정합니다.
 - a. 시스템에서 파일의 위치를 반영하도록 CONFIGFILE 속성을 변경하십시오.
 - b. 다른 속성은 변경하지 않는 상태로 두십시오.

제공된 그대로 파이프라인 구성 파일을 사용하고 그 콘텐츠를 변경하지 마십시오.

4. PIPELINE 자원을 설치하십시오. 등록 서비스 제공자 PIPELINE 자원은 서비스 요청자 또는 제공자 애플리케이션과 같은 CICS 리전에 있지 않아도 되지만 적절한 MRO 또는 APPC 연결을 사용하여 해당 리전과 연결되어야 합니다.
5. 등록 서비스 제공자에서 사용되는 URIMAP를 변경하지 않고 PIPELINE과 같은 리전에 설치하십시오. 이 URIMAP은 DFHRSURI로 이름 지정됩니다.
6. 등록 서비스의 요청자 PIPELINE 자원을 수정하십시오. PIPELINE은 DFHWSATR로 이름 지정되고 CONFIGFILE 속성에서 파이프라인 구성 파일 `/usr/lpp/cicsts/cicsts53/pipeline/configs/registrationserviceREQ.xml`을 지정합니다.
 - a. 시스템에서 파일의 위치를 반영하도록 CONFIGFILE 속성을 변경하십시오.
 - b. 다른 속성은 변경하지 않는 상태로 두십시오.

제공된 그대로 파이프라인 구성 파일을 사용하고 그 콘텐츠를 변경하지 마십시오.

7. PIPELINE 자원을 설치하십시오. 등록 서비스 요청자 PIPELINE 자원은 서비스 요청자 및 제공자 애플리케이션과 같은 CICS 리전에 설치해야 합니다.
8. 등록 서비스 제공자 파이프라인에서 사용되는 프로그램을 PIPELINE 자원과 같은 리전에 설치하십시오. 이러한 프로그램은 DFHWSATX, DFHWSATR 및 DFHPIRS입니다. 두 PIPELINE 자원이 서로 다른 리전에 있는 경우 이러한 프로그램을 두 리전 모두에 설치해야 합니다.
9. 헤더 핸들러 프로그램의 PROGRAM 자원 정의를 설치하십시오. 이 프로그램은 DFHWSATH로 이름 지정됩니다. 서비스 제공자 및 요청자 애플리케이션이 실행되는 리전에 PROGRAM을 설치하십시오.

결과

서비스 제공자 및 요청자 애플리케이션이 WS-AtomicTransaction 및 WS-Coordination 프로토콜을 사용하여 분배 트랜잭션에 참가할 수 있도록 CICS가 구성되었습니다.

다음에 수행할 작업

이제 각 참가 애플리케이션을 개별적으로 구성해야 합니다.

웹 서비스 트랜잭션을 위한 서비스 제공자 구성

서비스 제공자 애플리케이션이 웹 서비스 트랜잭션에 참가하는 경우 파이프라인 구성 파일은 <headerprogram> 요소 및 <service_parameter_list> 요소를 지정해야 합니다.

시작하기 전에

서비스 제공자 애플리케이션을 웹 서비스 트랜잭션에 참가시키려는 경우 이 애플리케이션은 SOAP 프로토콜을 사용하여 서비스 요청자와 통신해야 하고 파이프라인이 CICS 제공 SOAP 메시지 핸들러 중 하나를 사용하도록 구성해야 합니다. 서비스 제공자 애플리케이션을 올바르게 구성했다고 해도 이 애플리케이션은 요청자 애플리케이션이 참가하도록 설정된 경우에만 서비스 요청자와의 웹 서비스 트랜잭션에 참가합니다.

이 태스크 정보

사용자 애플리케이션에 특정한 파이프라인 구성 정보 외에도 구성 파일은 애플리케이션이 웹 서비스 트랜잭션에 참가하도록 하기 위해 CICS가 사용하는 정보를 포함해야 합니다.

CICS는 파일 /usr/lpp/cicsts/cicsts53/samples/pipelines/wsatprovider.xml 디렉토리(여기서, /usr/lpp/cicsts/cicsts53는 z/OS UNIX에서 CICS 파일의 기본 설치 디렉토리임)에 이 정보를 포함하는 파이프라인 구성 파일의 예제를 제공합니다.

프로시저

1. 터미널 핸들러의 정의에서 <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소의 <headerprogram> 요소를 코드화하십시오. <program_name>, <namespace>, <localname> 및 <mandatory> 요소를 이 예제에 표시된 그대로 코드화하십시오.

```
<terminal_handler>
  <cics_soap_1.1_handler>
    <headerprogram>
      <program_name>DFHWSATH</program_name>
      <namespace>http://schemas.xmlsoap.org/ws/2004/10/wscoord</namespace>
      <localname>CoordinationContext</localname>
```

```

        <mandatory>false</mandatory>
    </headerprogram>
</cics_soap_1.1_handler>
</terminal_handler>

```

애플리케이션이 필요로 하는 경우 다른 <headerprogram> 요소를 포함시키십시오.

2. <service_parameter_list>에 있는 <registration_service_endpoint> 요소를 코드화하십시오. 다음과 같이 <registration_service_endpoint>를 코드화하십시오.

```

<registration_service_endpoint>
http://address:port/cicswsat/RegistrationService
</registration_service_endpoint>

```

*address*는 등록 서비스 제공자 파이프라인이 있는 CICS 리전의 IP 주소입니다.

*port*는 등록 서비스 제공자 파이프라인에서 사용되는 포트 번호입니다.

그 나머지도 표시된 그대로 코드화하십시오. 문자열 *cicswsat/*

*RegistrationService*는 URIMAP DFHRSURI의 PATH 속성과 일치합니다.

```

<registration_service_endpoint>
http://provider.example.com:7160/cicswsat/RegistrationService
</registration_service_endpoint>

```

웹 서비스 트랜잭션을 위한 서비스 요청자 구성

서비스 요청자 애플리케이션이 웹 서비스 트랜잭션에 참가하는 경우 파이프라인 구성 파일은 <headerprogram> 요소 및 <service_parameter_list> 요소를 지정해야 합니다.

시작하기 전에

서비스 요청자 애플리케이션을 웹 서비스 트랜잭션에 참가시키려는 경우 이 애플리케이션은 SOAP 프로토콜을 사용하여 서비스 제공자와 통신해야 하고 파이프라인이 CICS 제공 SOAP 메시지 핸들러 중 하나를 사용하도록 구성해야 합니다. 서비스 요청자 애플리케이션을 올바르게 구성했다고 해도 이 애플리케이션은 제공자 애플리케이션이 참가하도록 설정된 경우에만 서비스 제공자와의 웹 서비스 트랜잭션에 참가합니다.

이 태스크 정보

사용자 애플리케이션에 특정한 파이프라인 구성 정보 외에도 구성 파일은 애플리케이션이 웹 서비스 트랜잭션에 참가하도록 하기 위해 CICS가 사용하는 정보를 포함해야 합니다.

CICS는 파일 */usr/lpp/cicsts/cicsts53/samples/pipelines/wsatrequester.xml* 디렉토리(여기서, */usr/lpp/cicsts/cicsts53*는 z/OS UNIX에서 CICS 파일의 기본 설치 디렉토리임)에 이 정보를 포함하는 파이프라인 구성 파일의 예제를 제공합니다.

프로시저

1. <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소의 <headerprogram> 요소를 코드화하십시오. <program_name>, <namespace>, <localname> 및 <mandatory> 요소를 다음 예제에 표시된 그대로 코드화하십시오.

```
<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>
```

애플리케이션이 필요로 하는 경우 다른 <headerprogram> 요소를 포함시킬 수 있습니다.

2. <service_parameter_list>에 있는 <registration_service_endpoint> 요소를 코드화하십시오. 다음과 같이 <registration_service_endpoint>를 코드화하십시오.

```
<registration_service_endpoint>
http://address:port/cicswsat/RegistrationService
</registration_service_endpoint>
```

*address*는 등록 서비스 제공자 파이프라인이 있는 CICS 리전의 IP 주소입니다.

*port*는 등록 서비스 제공자 파이프라인에서 사용되는 포트 번호입니다.

시작 <registration_service_endpoint> 요소, 해당 콘텐츠 및 종료 <registration_service_endpoint> 요소 사이에는 공백이 없어야 합니다. 이 예제에서는 명료성을 위해 공백이 포함되었습니다.

3. CICS가 같은 작업 단위에 있는 요청에 같은 트랜잭션 컨텍스트를 사용하는 대신 각 요청에 대해 새 트랜잭션 컨텍스트를 작성하게 하려는 경우

<service_parameter_list>에 있는 비어 있는 요소
<new_tx_context_required/>를 파이프라인 구성 파일에 추가하십시오.

```
<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> 요소의 시작, 해당 콘텐츠 및 <registration_service_endpoint> 요소의 종료 사이에는 공백이 없어야 합니다. 이 예제에서는 명료성을 위해 공백이 포함되었습니다.

<new_tx_context_required/> 설정은 CICS의 기본값이 아니며 예제 파이프라인 구성 파일 wsatprovider.xml에 포함되지 않습니다. <service_parameter_list>에 있는 <new_tx_context_required/>를 파이프라인 구성 파일에 추가하면 CICS에 대한 루프백 호출이 허용되어 이 상황에서 교착 상태가 발생할 수 있음을 알 수 있습니다.

SOAP 메시지가 원자적 트랜잭션의 파트인지 판별

원자적 트랜잭션 파이프라인에서 CICS 웹 서비스가 호출되면 SOAP 메시지는 원자적 트랜잭션의 파트가 아니어도 됩니다.

이 태스크 정보

SOAP 메시지가 원자적 트랜잭션의 파트인 경우 <soapenv:Header> 요소에는 특정 정보가 포함됩니다. SOAP 메시지가 원자적 트랜잭션의 파트인 경우 다음 중 하나를 수행할 수 있습니다.

프로시저

- 추적을 사용하여 <soapenv:Header> 요소의 콘텐츠 내부를 보십시오.
 - 컴포넌트 PI를 사용하여 보조 추적을 수행하고 추적 레벨을 2로 설정하십시오.
 - 요청 컨테이너에 대한 정보를 포함하는 추적점 PI 0A31을 찾으십시오. 특히 <cicswsa:Action> 요소 바로 앞에 표시되는 PIIS EVENT - REQUEST_CNT를 찾으십시오.
- DFHWSATP 파이프라인에서 사용자 작성 메시지 핸들러 프로그램을 사용하여 DFHREQUEST 컨테이너가 RECEIVE-REQUEST 데이터를 포함할 경우 이 컨테이너의 콘텐츠를 표시하십시오. 이 접근 방식을 선택하는 경우 파이프라인 구성 파일에서 메시지 핸들러 프로그램을 정의해야 합니다.

예

다음 예제는 원자적 트랜잭션의 SOAP 엔벨로프 헤더에서 볼 수 있는 정보를 표시합니다.

```
<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/wsdl</wscoor:CoordinationType> 2
    <wscoor:RegistrationService 3
      xmlns:wscoor="http://schemas.xmlsoap.org/ws/2004/10/wsdl">
        <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: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. CoordinationContext는 SOAP 메시지가 원자적 트랜잭션에 참가하도록 계획됨을 표시합니다. CoordinationContext는 조정 서비스의 일부가 될 웹 서비스 제공자의 필요 정보를 포함하고 제공자가 헤더를 인식하고 처리하도록 구성되는 것으로 가정합니다.
2. CoordinationType은 조정 컨텍스트가 준수하는 WS-AT 스펙의 버전을 표시합니다.
3. 조정 RegistrationService는 조정자의 등록점이 있는 위치 및 참가 웹 서비스가 원자적 트랜잭션의 컴포넌트로서 등록을 시도할 때 조정자에게 리턴해야 하는 정보를 설명합니다.

원자적 트랜잭션의 진행상태 확인

CICS 웹 서비스가 원자적 트랜잭션의 파트로 호출되면 트랜잭션은 여러 상태를 통과합니다. 이러한 상태는 트랜잭션이 성공적이었는지 및 롤백해야 되는지 여부를 표시합니다.

이 태스크 정보

이 정보에 액세스해야 하는 경우 다음을 수행할 수 있습니다.

프로시저

- 추적을 사용하여 <cicswsa:Action> 요소의 콘텐츠 내부를 보십시오.
 1. 컴포넌트 PI를 사용하여 보조 추적을 수행하고 추적 레벨을 2로 설정하십시오.
 2. 요청 컨테이너에 대한 정보를 포함하는 추적점 PI 0A31을 찾으십시오. 특히 <cicswsa:Action> 요소 바로 앞에 표시되는 PIIS EVENT - REQUEST_CNT를 찾으십시오.
- DFHWSATR 및 DFHWSATP 파이프라인에서 사용자 작성 메시지 핸들러 프로그램을 사용하여 DFHWS-SOAPACTION 컨테이너의 콘텐츠를 표시하십시오. 이 접근 방식을 선택하는 경우 파이프라인 구성 파일에서 메시지 핸들러 프로그램을 정의해야 합니다.

예

성공적으로 완료되고 커밋된 트랜잭션의 상태는 다음과 같습니다.

```

"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 "

```


롤백되는 트랜잭션의 상태는 다음과 같습니다.

```
"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/wsat/Rollback"  
"http://schemas.xmlsoap.org/ws/2004/10/wsat/Aborted"
```

2진 데이터의 MTOM/XOP 최적화 지원

표준 SOAP 메시지에서 2진 오브젝트는 base64로 인코딩되며 메시지 본문에 포함됩니다. 메시지 본문에서는 2진 오브젝트 크기가 33% 증가합니다. 아주 큰 2진 오브젝트의 경우에는 이러한 크기 증가로 인해 전송 시간이 큰 영향을 받습니다. MTOM/XOP 구현은 이 문제점의 솔루션을 제공합니다.

SOAP 메시지 전송 최적화 메커니즘(MTOM) 및 XML-2진 최적화 패키징(XOP) 스펙은 MTOM/XOP라고도 하며 SOAP 메시지 내 큰 base64Binary 데이터 오브젝트의 전송을 최적화하는 방법을 정의합니다.

- MTOM 스펙은 2진 데이터를 구분하여(그렇지 않으면 base64로 인코딩됨) SOAP 메시지를 최적화한 후 MIME 멀티파트/관련 메시지를 사용하여 개별 2진 첨부 파일로 보내는 방법을 개념적으로 정의합니다. 이러한 유형의 MIME 메시지를 *MTOM* 메시지라고 합니다. 데이터를 2진 형식으로 보내면 크기가 크게 감소하여 SOAP 메시지 전송이 최적화됩니다.
- XOP 스펙은 MIME 메시지를 포함하되 이에 한정되지 않는 패키징 형식의 2진 첨부 파일을 사용하여 XML 메시지를 최적화하는 구현을 정의합니다.

CICS는 전송 프로토콜이 WebSphere MQ, HTTP 또는 HTTPS인 경우 요청자 및 제공자 파이프라인에서 모두 이 스펙에 대한 지원을 구현합니다. SOAP 메시지에 직접 base64Binary 데이터를 포함하지 않고 웹 서비스 제공자 또는 요청자로 배치된 CICS 애플리케이션이 이 지원을 사용하여 2진 첨부 파일을 포함하는 MTOM 메시지를 송수신할 수도 있습니다.

이 지원은 파이프라인 구성 파일에서 추가 옵션을 사용하여 구성할 수 있습니다.

MTOM/XOP 및 SOAP

MTOM/XOP가 SOAP 메시지를 최적화하기 위해 사용되는 경우 XOP 처리를 사용하여 MIME 멀티파트/관련 메시지로 직렬화됩니다. base64Binary 데이터는 이메일 첨부 파일과 유사한 방식으로 SOAP 메시지에서 추출된 후 MIME 메시지 내 개별 2진 첨부 파일로 패키지화됩니다.

첨부 파일이 2진 형식으로 인코딩되므로 base64Binary 데이터 크기가 크게 감소합니다. SOAP 메시지의 XML은 XOP 형식으로 변환되며 이 때 base64Binary 데이터가 URI를 사용하여 관련 MIME 첨부 파일을 참조하는 특수 <xop:Include> 요소로 바뀝니다.

수정된 SOAP 메시지를 *XOP* 문서라고 하며 메시지 내에서 루트 문서를 구성합니다. *XOP* 문서와 2진 첨부 파일이 함께 *XOP* 패키지를 구성합니다. SOAP MTOM 스펙에 적용되는 경우, *XOP* 패키지는 MTOM 형식의 MIME 메시지입니다.

루트 문서는 MIME 메시지의 전체 content-type 헤더에서 해당 Content-ID를 참조하여 식별됩니다. 다음은 content-type 헤더의 예입니다.

```
Content-Type: Multipart/Related; boundary=MIME_boundary;
type="application/soap+xml"; start="<claim@insurance.com>"
```

start 매개변수에는 *XOP* 문서의 Content-ID가 포함됩니다. 이 매개변수가 content-type 헤더에 포함되지 않으면 MIME 메시지의 첫 번째 파트를 *XOP* 문서로 가정합니다.

MIME 메시지에서 첨부 파일의 순서는 중요하지 않습니다. 예를 들어, 일부 메시지에서 2진 첨부 파일이 *XOP* 문서 앞에 나타날 수 있습니다. MIME 메시지를 처리하는 애플리케이션은 특정 순서로 나타나는 첨부 파일에 의존해서는 안 됩니다. 자세한 정보는 MTOM/XOP 스펙을 읽으십시오.

다음 예제는 *XOP* 처리를 사용하여 JPEG 이미지를 포함하는 단순 SOAP 메시지가 최적화되는 방법을 보여줍니다. SOAP 메시지는 다음과 같습니다.

```
<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>
```

이 SOAP 메시지의 MTOM/XOP 버전은 다음과 같습니다.

```
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** 매개변수는 MIME 메시지에서 루트 XOP 문서 파트를 나타냅니다.
2. Content-ID 값은 MIME 메시지의 일부를 식별합니다. 이 경우 루트 XOP 문서입니다.
3. <xop:Include> 요소는 JPEG 2진 첨부 파일을 참조합니다.
4. Content-ID는 2진 첨부 파일에서 JPEG를 식별합니다.

CICS의 MTOM 메시지 및 2진 첨부 파일

CICS는 웹 서비스 제공자 및 요청자 파이프라인에서 모두 MTOM 핸들러 프로그램과 XOP 처리를 사용하여 MTOM 메시지 핸들링을 지원하고 제어합니다.

파이프라인 구성 파일을 사용하여 MTOM 지원을 구성하고 사용하도록 설정할 수 있습니다. 파이프라인에 MTOM 지원을 사용하도록 설정되면 CICS가 인바운드 MTOM 메시지를 자동으로 풀고 아웃바운드 메시지를 패키징화합니다. 파이프라인에 MTOM 지원을 사용하도록 설정되지 않고 CICS가 MTOM 메시지를 수신하면 Java 기반 파이프라인이 인바운드 MTOM 메시지를 승인하지만 다른 SOAP 파이프라인은 SOAP 결함과 함께 인바운드 MTOM 메시지를 거부합니다.

Java 기반 파이프라인에 대한 구성 옵션

제공자 파이프라인이 다음 태스크를 수행하도록 구성할 수 있습니다.

- MTOM 메시지를 승인하지만 MTOM 응답 메시지는 보내지 않습니다.
- MTOM 메시지를 승인하고 MTOM 응답 메시지를 항상 보냅니다.
- XOP 문서와 2진 첨부 파일을 Axis2 모드로 처리합니다.

요청자 파이프라인이 다음 태스크를 수행하도록 구성할 수 있습니다.

- MTOM 메시지를 보내지 않고 MTOM 응답 메시지는 승인합니다.
- MTOM 메시지를 항상 보내고 MTOM 응답 메시지를 승인합니다.
- XOP 문서와 2진 첨부 파일을 Axis2 모드로 처리합니다.

Java를 지원하지 않는 파이프라인에 대한 구성 옵션

제공자 파이프라인이 다음 태스크를 수행하도록 구성할 수 있습니다.

- MTOM 메시지를 승인하지만 MTOM 응답 메시지는 보내지 않습니다.
- MTOM 메시지를 승인하고 동일한 유형의 응답 메시지를 보냅니다.
- MTOM 메시지를 승인하지만 2진 첨부 파일이 있는 경우에만 MTOM 메시지를 보냅니다.
- MTOM 메시지를 승인하고 MTOM 응답 메시지를 항상 보냅니다.

- XOP 문서와 2진 첨부 파일을 직접 또는 호환 모드로 처리합니다.

요청자 파이프라인이 다음 태스크를 수행하도록 구성할 수 있습니다.

- MTOM 메시지를 보내지 않고 MTOM 응답 메시지는 승인합니다.
- 2진 첨부 파일이 있는 경우에만 MTOM 메시지를 보내고 MTOM 응답 메시지를 승인합니다.
- MTOM 메시지를 항상 보내고 MTOM 응답 메시지를 승인합니다.
- XOP 문서와 2진 첨부 파일을 직접 또는 호환 모드로 처리합니다.

지원 모드

파이프라인에는 XOP 문서와 연관된 2진 첨부 파일을 처리하기 위한 세 가지 지원 모드가 제공됩니다.

Axis2 모드

Axis2 모드는 웹 서비스 파이프라인의 터미널 핸들러가

<cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 메시지 핸들러인 경우 사용됩니다.

직접 모드

직접 모드에서는 인바운드 또는 아웃바운드 MTOM 메시지와 연관된 2진 첨부 파일이 파이프라인을 통해 컨테이너에서 전달되며 데이터 변환을 수행할 필요 없이 애플리케이션으로 직접 처리됩니다.

호환 모드

호환 모드는 파이프라인 처리 시 메시지가 표준 XML 형식이어야 하는 경우 사용되며 2진 데이터가 메시지 내에서 base64Binary 필드로 저장됩니다. 인바운드 메시지의 경우, XOP 문서와 2진 첨부 파일이 파이프라인 시작(웹 서비스 보안을 사용하는 경우) 또는 파이프라인 끝(웹 서비스 유효성 검증을 사용하는 경우)에서 표준 XML 메시지로 재구성됩니다. 아웃바운드 메시지의 경우에는 표준 XML 메시지가 작성되어 파이프라인을 따라 전달됩니다. 이 XML 메시지는 CICS가 보내기 바로 전에 MTOM 핸들러에 의해 XOP 형식으로 변환됩니다.

호환 모드는 2진 데이터가 base64 형식으로 변환되었다가 다시 2진 형식으로 변환되므로 직접 모드보다 훨씬 비효율적입니다. 그러나 애플리케이션을 변경하지 않고 웹 서비스가 다른 MTOM/XOP 웹 서비스 요청자 및 제공자와 상호 운용될 수 있습니다.

Java를 지원하지 않는 파이프라인에 대한 인바운드 MTOM 메시지 처리

MTOM 핸들러를 Java를 지원하지 않는 파이프라인에서 사용하는 경우, DFHREQUEST 또는 DFHRESPONSE 컨테이너에서 인바운드 메시지의 헤더를 확인하여 전송 핸들링 처리 중에 메시지 형식을 판별합니다.

MIME Multipart/Related 메시지가 수신되면 MTOM 핸들러가 다음과 같이 메시지 패키지를 해제합니다.

1. 각 2진 첨부 파일의 헤더와 2진 데이터는 개별 컨테이너에 둡니다.
2. 컨테이너 목록은 DFHWS-XOP-IN 컨테이너에 둡니다.
3. 메시지 루트를 구성한 XOP 문서를 다시 DFHREQUEST 또는 DFHRESPONSE 컨테이너에 두어 원래 메시지를 바꿉니다.

2진 첨부 파일이 없는 경우에는 XOP 문서가 보통 XML 메시지로 처리되며 XOP 처리가 필요하지 않습니다. 2진 첨부 파일이 있는 경우에는 메시지에 XOP 처리를 사용합니다.

XOP 처리를 사용하는 경우, MTOM 핸들러가 파이프라인 특성을 확인하여 현재 메시지가 직접 또는 호환 모드에서 처리되어야 하는지 여부를 판별하고 이 정보를 DFHWS-MTOM-IN 컨테이너에 둡니다.

제공자 모드의 경우 MTOM 핸들러가 또한 DFHWS-MTOM-OUT 컨테이너를 작성하여 아웃바운드 응답 메시지 처리 방법을 판별합니다.

직접 모드

CICS 웹 서비스 지원을 사용하는 경우, 즉 서비스 제공자 파이프라인이 DFHPITP 애플리케이션 핸들러를 사용하거나 서비스 요청자 파이프라인이 **INVOKE WEBSERVICE**를 사용하여 호출되는 경우에는 파이프라인이 XOP 문서와 2진 첨부 파일을 직접 모드로 처리할 수 있습니다.

이 모드에서는 XOP 문서와 연관된 컨테이너가 MTOM 핸들러에 의해 파이프라인 내 다음 메시지 핸들러로 전달되어 처리됩니다. CICS 웹 서비스 지원은 <xop:Include> 요소를 해석합니다. base64Binary 필드가 애플리케이션 데이터 구조에서 컨테이너로 표시되는 경우에는 첨부 파일 컨테이너 이름이 해당 구조에 저장됩니다. 필드가 가변 또는 고정 길이 문자열로 표시되는 경우에는 컨테이너의 콘텐츠가 관련 애플리케이션 데이터 구조 필드에 복사됩니다. 데이터 구조는 그런 다음 애플리케이션 프로그램으로 전달됩니다.

호환 모드

파이프라인이 사용자 정의 애플리케이션 핸들러를 사용하도록 구성되거나 웹 서비스 보안 또한 사용하도록 설정되는 경우, 메시지가 호환 모드에서 처리됩니다. 이 모드에서는 XOP 처리를 사용하여 XOP 문서와 2진 첨부 파일이 즉시 SOAP 메시지로 재구성되어 파이프라인에서 콘텐츠를 처리할 수 있습니다. XOP 처리는 다음 태스크를 수행합니다.

1. <xop:Include> 요소의 XOP 문서를 스캔하고 각 발생을 base64 인코딩 형식으로 참조되는 첨부 파일의 2진 데이터로 바꿉니다.

2. DFHWS-XOP-IN 컨테이너와 모든 첨부 컨테이너를 버립니다.

재구성된 SOAP 메시지는 파이프라인의 다음 핸들러로 전달되어 정상적으로 처리됩니다.

웹 서비스 유효성 검증을 사용하도록 설정된 경우에는 메시지가 애플리케이션 핸들러에 도달하면 파이프라인이 호환 모드로 전환됩니다. 메시지는 SOAP 메시지로 재구성되고 유효성을 검증한 후 애플리케이션으로 전달됩니다.

Java를 지원하지 않는 파이프라인에 대한 아웃바운드 MTOM 메시지 처리

Java를 지원하지 않는 파이프라인이 아웃바운드 MTOM 메시지를 보내도록 구성되면 웹 서비스 및 파이프라인 특성을 확인하여 메시지를 처리, 전송하는 방법을 판별합니다.

이 특성은 두 컨테이너(DFHWS-MTOM-OUT 및 DFHWS-XOP-OUT)에 저장됩니다. 요청자 모드 파이프라인의 경우, 이 컨테이너는 애플리케이션이 **EXEC CICS INVOKE WEBSERVICE** 명령을 실행할 때 CICS로 작성됩니다. 제공자 모드 파이프라인의 경우에는 DFHWS-MTOM-OUT 컨테이너가 인바운드 메시지가 수신될 때 판별된 옵션으로 이미 초기화되어 있습니다.

아웃바운드 메시지가 직접 모드에서 처리될 수 있으면 메시지 최적화가 즉시 수행됩니다. 아웃바운드 메시지를 호환 모드로 처리해야 하는 경우, 파이프라인 처리 맨 끝에서 최적화가 수행됩니다.

CICS 웹 서비스 지원을 사용하여 웹 서비스 제공자 또는 요청자 애플리케이션을 배치하지 않았거나 파이프라인에서 웹 서비스 보안 또는 웹 서비스 유효성 검증을 사용하도록 설정한 경우에는 아웃바운드 메시지가 호환 모드로 처리됩니다.

직접 모드

직접 모드에서는 다음 처리가 수행됩니다.

1. 컨테이너 DFHWS-DATA의 애플리케이션 데이터 구조에서 XOP 문서가 구성됩니다. 크기가 1500바이트 이상인 2진 필드가 식별되며 2진 첨부 파일을 설명하는 MIME 헤더와 2진 데이터는 개별 컨테이너에 포함됩니다. 2진 데이터가 이미 컨테이너에 있는 경우에는 해당 컨테이너가 직접 첨부 파일로 사용됩니다. 그런 다음 `<xop:Include>` 요소가 생성된 Content-ID를 사용하여 일반 base64 인코딩 2진 데이터 대신 XML에 삽입됩니다. 예를 들어, 다음과 같습니다.

```
<xop:Include href="cid:generated-content-ID-value"
  xmlns:xop="http://www.w3.org/2004/08/xop/include">
```

2. 모든 컨테이너가 DFHWS-XOP-OUT 컨테이너의 첨부 파일 목록에 추가됩니다.
3. SOAP 핸들러가 DFHWS-DATA를 처리하면 XOP 문서와 SOAP 엔벨로프가 DFHREQUEST 또는 DFHRESPONSE 컨테이너에 저장되며 파이프라인을 통해 처리됩니다.

4. 마지막 메시지 핸들러가 완료되면 MTOM 핸들러가 XOP 문서와 2진 첨부 파일을 MIME 멀티파트/관련 메시지로 패키징하여 웹 서비스 요청자 또는 제공자로 보냅니다. DFHWS-XOP-OUT 컨테이너와 연관된 컨테이너는 버려집니다.

호환 모드

파이프라인이 XOP 문서를 직접 핸들링할 수 없는 경우에는 다음과 같은 처리가 수행됩니다.

1. SOAP 본문이 애플리케이션 데이터 구조에서 DFHWS-DATA에 구성되고 파이프라인에서 정상적으로 처리됩니다.
2. 최종 핸들러가 메시지 처리를 완료하면 MTOM 핸들러가 DFHWS-MTOM-OUT 컨테이너의 옵션을 확인하고 선택적으로 2진 첨부 파일이 있는지 여부를 고려하여 MTOM을 사용해야 하는지 여부를 판별합니다. MTOM 핸들러가 MTOM이 필요하지 않은 것으로 판별하면 XOP 처리가 수행되지 않고 CICS가 SOAP 메시지를 정상적으로 보냅니다.
3. MTOM 핸들러가 아웃바운드 메시지를 MTOM 형식으로 보내야 하는 것으로 판별하면 XOP 처리 시 올바른 필드에 대한 메시지를 스캔하여 데이터를 2진 첨부 파일로 분할합니다. 올바른 필드는 요소에 대해 MIME **contentType** 속성이 지정되어야 하며 연관된 2진 값이 표준 양식의 올바른 base64Binary 데이터로 구성되어야 합니다. 데이터 크기는 1500바이트보다 커야 합니다. XOP 처리는 2진 첨부 파일과 첨부 파일 목록을 작성한 다음 필드를 <xop:Include> 요소로 바꿉니다.
4. MTOM 핸들러는 XOP 문서와 2진 첨부 파일을 MIME 멀티파트/관련 메시지로 패키징하고 CICS는 해당 패키지를 웹 서비스 요청자 또는 제공자로 보냅니다.

MTOM/XOP 사용 시 제한사항

MTOM/XOP를 지원하기 위해 파이프라인 구성 파일에서 <mtom> 요소를 지정하거나 파이프라인에서 MTOM 핸들러를 사용하도록 설정할 수 있습니다. 그러나 각 방법과 연관된 제한사항이 있습니다.

Java 기반 파이프라인에 대한 제한사항

파이프라인 구성 파일에서 <mtom> 요소를 지정하면 Java 기반 파이프라인에 MTOM/XOP 지원을 사용할 수 있습니다. 그러나 이 MTOM/XOP 구현에는 제한사항이 있습니다.

DFHPITP 애플리케이션 핸들러

MTOM/XOP 지원의 Axis2 모드는 DFHPITP를 애플리케이션 핸들러로 지정하는 파이프라인에 사용할 수 없습니다.

WS-보안

MTOM/XOP 지원의 Axis2 모드는 XML 서명이 필요한 WS-보안 구성을 사용하는 파이프라인에 사용할 수 없습니다.

INQUIRE PIPELINE 명령 사용

MTOM/XOP 지원의 Axis2 모드를 사용하는 Java 기반 파이프라인에 대해 **INQUIRE PIPELINE** 명령이 실행되는 경우 **Mtomst**, **Sendmtomst**, **Mtomnoxopst**, **Xopsupportst**, **Xopdirectst** 속성은 **Nomtom**으로 보고됩니다. 자세한 정보는 **INQUIRE PIPELINE**을 참조하십시오.

기타 SOAP 파이프라인에 대한 제한사항

파이프라인에서 MTOM 핸들러를 사용한다는 것은 MTOM/XOP 최적화를 사용하는 웹 서비스 구현을 지원할 수 있음을 의미합니다. 호환 모드 옵션은 웹 서비스 애플리케이션을 변경하지 않고 이 웹 서비스를 상호 운용할 수 있음을 의미합니다. 그러나 MTOM/XOP를 사용할 수 없거나 사용이 제한되는 특정 상황이 있습니다.

CICS 웹 서비스 지원 사용

MTOM/XOP에 대한 직접 모드 최적화는 DFHWS2LS를 맵핑 레벨 1.2 이상에서 사용하고 WSDL 문서에 xsd:base64Binary 유형의 필드가 최소 하나 포함되어 있는 경우에만 사용할 수 있습니다. DFHLS2WS를 사용하여 사용할 수 있는 웹 서비스는 XOP 최적화에 적합하지 않습니다.

CHAR-VARYING=BINARY를 지정한 DFHLS2WS를 사용하여 생성된 웹 서비스는 MTOM/XOP 최적화에 적합할 수도 있습니다. DFHLS2WS를 사용하여 생성된 기타 웹 서비스에는 2진 데이터가 포함되지 않고 MTOM/XOP 최적화에 적합하지 않지만 MTOM/XOP를 지원하는 PIPELINE에서 정상적으로 작동합니다.

제공자 파이프라인

CICS는 제공자 파이프라인에서 구성될 수 있는 기본 애플리케이션 핸들러 DFHPITP를 제공합니다. 이 애플리케이션 핸들러는 XOP 문서를 핸들링할 수 있으며 파이프라인 처리를 직접 및 호환 모드에서 모두 지원하는 데 필요한 컨테이너를 작성할 수 있습니다. 제공자 파이프라인에서 고유한 애플리케이션 핸들러를 사용하고 MTOM/XOP를 사용하도록 설정하려는 경우에는 파이프라인에 호환 모드에서 실행되도록 구성해야 합니다.

요청자 파이프라인

애플리케이션이 **INVOKE WEBSERVICE** 명령을 사용하는 경우 CICS가 직접 및 호환 모드에서 둘 다 SOAP 메시지 최적화를 처리합니다. 프로그램 DFHPIRT를 사용하여 파이프라인을 시작하는 경우에는 호환 모드에서만 MIME 멀티파트/관련 메시지를 송수신할 수 있습니다.

웹 서비스 보안

파이프라인 구성 파일의 MTOM 핸들러가 직접 모드에서 실행되도록 설정하고 웹 서비스 보안 메시지 핸들러도 사용하도록 설정하려는 경우에는 파이프라인이 호환 모드에서의 MTOM 메시지 핸들러만 지원합니다.

2진 데이터 핸들링

대용량 2진 데이터(예를 들어, JPEG와 같은 그래픽 파일)를 웹 서비스에 포함하는 경우 MTOM/XOP를 사용하여 서비스 제공자 또는 요청자로 보내는 메시지의 크기를 최적화할 수 있습니다. MTOM/XOP를 사용하여 최적화될 수 있는 2진 데이터의 최소 크기는 1500바이트입니다. 필드의 2진 데이터가 1500바이트 미만이면 CICS가 필드를 최적화하지 않습니다.

XOP 스펙에 명시된 대로 base64Binary 데이터에는 공백이 없어야 합니다. base64Binary 데이터를 생성하는 애플리케이션 프로그램은 표준 양식을 사용해야 합니다. 아웃바운드 메시지의 base64Binary 데이터에 공백이 포함되면 CICS가 데이터를 2진 첨부 파일로 변환하지 않습니다. base64Binary 데이터가 CICS로 생성되면 필드가 표준 양식으로 제공되므로 공백이 포함되지 않습니다.

아웃바운드 메시지에 대해 XOP 처리가 호환 모드로 발생하려면 base64Binary 필드에 **contentType** 속성이 있어야 합니다. **contentType** 속성이 hexBinary 필드에 있어서는 안 됩니다.

웹 서비스 유효성 검증

웹 서비스 유효성 검증을 활성화하면 다음 파이프라인 처리가 실행됩니다.

- 직접 모드로 파이프라인을 통해 인바운드 XOP 문서가 전달되면 CICS가 자동으로 호환 모드로 전환되며 CICS 웹 서비스 지원이 문서 유효성을 검증하려고 할 때 표준 XML로 다시 변환합니다.
- 아웃바운드 SOAP 메시지가 표준 XML로 생성되고 호환 모드로 처리됩니다.

유효성 검증 처리는 XOP 문서의 콘텐츠를 처리할 수 없으므로 추가 파이프라인 처리가 필요합니다.

CICS에서 MTOM/XOP를 지원하도록 구성

CICS에서 MTOM 메시지를 지원하려면 파이프라인 구성 파일에서 파이프라인 유형에 올바른 MTOM/XOP 지원을 지정해야 합니다.

Java 기반 파이프라인에 대한 MTOM/XOP 지원 구성

Java 기반 파이프라인에 대한 MTOM/XOP 지원을 구성하려면 파이프라인 구성 파일에 <mtom> 요소를 추가해야 합니다.

시작하기 전에

이 작업을 수행하기 전에 MTOM/XOP의 구성 정보를 추가할 파이프라인 구성 파일을 식별하거나 작성해야 합니다.

이 태스크 정보

<mtom> 요소가 파이프라인 구성 파일에 정의된 경우 모든 인바운드 및 아웃바운드 메시지에 MTOM 지원을 사용할 수 있습니다. 그러나 이 요소가 파이프라인 구성 파일에 지정되지 않으면 인바운드 메시지에만 MTOM 지원을 사용할 수 있습니다.

프로시저

파이프라인 구성 파일에 <mtom> 요소를 추가하십시오. 이 요소는 선택적 <addressing> 요소 뒤와 선택적 <headerprogram> 요소 앞에 정의되어야 합니다.

예제

제공자 또는 요청자 모드 파이프라인의 경우 다음을 지정할 수 있습니다.

```
<cics_soap_1.2_handler_java>
  <jvmserver>JVMSERV1</jvmserver>
  <addressing></addressing>
  <mtom></mtom>
  <headerprogram>
    <program_name>HDRPROG4</program_name>
    <namespace>http://mynamespace</namespace>
    <localname>myheaderblock</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler_java>
```

다른 SOAP 파이프라인에 대한 MTOM/XOP 구성

<cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 핸들러를 사용하지 않는 파이프라인에 대한 MTOM/XOP 지원을 구성하려면 파이프라인 구성 파일에 MTOM 핸들러를 추가해야 합니다.

시작하기 전에

이 태스크를 수행하기 전에 MTOM/XOP의 구성 정보를 추가할 파이프라인 구성 파일을 식별하거나 작성해야 합니다.

프로시저

1. 파이프라인 구성 파일에 <cics_mtom_handler> 요소를 추가하십시오. 이 요소는 <provider_pipeline> 요소의 첫 번째 요소이자 <requester_pipeline> 요소에서 <service_parameter_list> 앞의 마지막 요소여야 합니다. 다음 요소를 코드화하십시오.

```
<cics_mtom_handler>
  <dfhmtom_configuration version="1">
  </dfhmtom_configuration>
</cics_mtom_handler>
```

<dfhmtom_configuration> 요소는 구성 내 다른 요소의 컨테이너입니다. MTOM/XOP 처리의 기본 설정을 승인하려는 경우 다음과 같이 비어 있는 요소를 지정할 수 있습니다.

```
<cics_mtom_handler/>
```

2. 옵션: <mtom_options> 요소를 코드화하십시오. 서비스 제공자 및 서비스 요청자 파이프라인에서 모두 이 요소는 아웃바운드 메시지를 MTOM 메시지로 패키지화해야 하는지 여부를 지정합니다.
 - a. 아웃바운드 메시지가 MTOM 메시지로 전송되어야 하는지 여부를 정의하도록 **send_mtom** 속성을 코드화하십시오. 이 속성에 대한 세부사항은 309 페이지의 『<mtom_options> 요소』를 참조하십시오.
 - b. 2진 첨부 파일이 없는 경우 아웃바운드 메시지가 MTOM 메시지로 전송되어야 하는지 여부를 정의하도록 **send_when_no_xop** 속성을 코드화하십시오. 이 속성에 대한 세부사항은 309 페이지의 『<mtom_options> 요소』를 참조하십시오.
3. 옵션: <xop_options> 요소를 **apphandler_supports_xop** 속성으로 코드화하십시오. 애플리케이션 핸들러가 XOP 문서를 직접 핸들링할 수 있는지 여부를 지정합니다. 이 속성을 포함하지 않는 경우 기본값은 <apphandler> 요소가 DFHPITP 또는 다른 프로그램을 지정하는지 여부에 따라 다릅니다. 이 속성에 대한 세부사항은 310 페이지의 『<xop_options> 요소』를 참조하십시오.
4. 옵션: <mime_options> 요소를 **content_id_domain** 속성으로 코드화하십시오. 2진 첨부 파일을 식별하는 데 사용되는 MIME content-ID 값을 생성할 때 사용되어야 하는 도메인 이름을 지정합니다. 이 속성에 대한 세부사항은 312 페이지의 『<mime_options> 요소』를 참조하십시오.

예제

다음 예제는 선택적 요소가 모두 있는 완료된 <cics_mtom_handler> 요소를 보여줍니다.

```
<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>
```

WS-Addressing에 대한 지원

CICS는 W3C(Worldwide Web Consortium) 웹 서비스 주소 지정(WS-Addressing) 스펙을 사용하는 서비스를 지원합니다. 이 스펙의 계열은 웹 서비스의 주소를 지정하고 엔드 투 엔드 주소 지정을 용이하게 하는 전송 독립 메커니즘을 제공합니다.

CICS는 기존 웹 서비스 애플리케이션이 WS-Addressing을 사용하는 웹 서비스의 요청을 승인할 수 있도록 보장합니다. 또한 엔드포인트 참조 및 SOAP 메시지의 메시지 주소 지정 특성을 사용하는 새 웹 서비스를 작성할 수 있습니다.

WS-Addressing은 MAP(Message Addressing Properties) 양식의 주소 지정 정보를 SOAP 메시지 헤더에 추가합니다. MAP은 메시징 정보(예: 고유 메시지 ID 및 메시지 생성 위치, 메시지를 보낼 대상 및 응답 또는 결함 메시지를 보낼 대상에 대해 자세히 설명하는 엔드포인트 참조)를 포함합니다. 엔드포인트 참조(EPR)는 특정 유형의 MAP이며 메시지의 대상 주소, 애플리케이션에서 사용할 선택적 참조 매개변수 및 선택적 메타데이터를 포함합니다.

WS-Addressing 지원의 기능

CICS에는 다음과 같은 WS-Addressing 지원 기능이 있습니다.

- 웹 서비스 요청자 및 제공자 애플리케이션이 사용자에게 재배치를 요구하지 않고도 WS-Addressing을 사용 중인 다른 서비스와 상호 작용할 수 있습니다. 파이프라인의 새 메시지 핸들러인 주소 지정 메시지 핸들러 DFHWSADH는 WS-Addressing 정보를 포함하는 메시지를 지정된 웹 서비스로 라우팅합니다.
- WS-Addressing API 명령을 사용하는 애플리케이션을 작성하여 엔드포인트 참조를 작성하고 주소 지정 컨텍스트를 작성, 갱신, 삭제 및 조회할 수 있습니다.
- 응답 메시지를 요청자 엔드포인트 외의 엔드포인트로 라우팅할 수 있습니다. 예를 들어, 결함 메시지를 전용 결함 핸들러로 라우팅할 수 있습니다.
- 애플리케이션에 참조 매개변수를 SOAP 헤더의 MAP 일부로 전달할 수 있습니다.

WS-Addressing 스펙 및 상호 운용성에 대한 지원

기본적으로 CICS는 권장 스펙을 지원합니다.

- W3C WS-Addressing 1.0 - 코어
- W3C WS-Addressing 1.0 - SOAP 바인딩
- W3C WS-Addressing 1.0 - 메타데이터

이러한 스펙은 <http://www.w3.org/2005/08/addressing> 네임스페이스로 식별합니다. 별도로 명시되지 않는 한 이 문서에 설명된 WS-Addressing 시맨틱은 권장 스펙을 참조합니다.

상호 운용성의 경우 CICS는 제출 스펙도 지원합니다.

- W3C WS-Addressing - 제출

이 스펙은 <http://schemas.xmlsoap.org/ws/2004/08/addressing> 네임스페이스로 식별합니다. 제출 스펙은 제출 스펙을 구현하는 웹 서비스 제공자 또는 클라이언트와 상호 운용해야 하는 경우에만 사용하십시오.

WS-Addressing 개요

웹 서비스 주소 지정(WS-Addressing)은 SOAP 메시지의 엔드포인트를 지정하기 위한 표준 프레임워크를 제공합니다. 이 프레임워크는 전송 중립이고 다른 전송 메커니즘을 사용하는 웹 서비스의 상호 운용성을 개선합니다. WS-Addressing 스펙은 메시지 주소 지정 특성 및 엔드포인트 참조를 소개합니다.

웹 서비스 주소 지정(WS-Addressing)은 웹 서비스의 주소를 지정하고 SOAP 메시지에 주소 지정 정보를 제공하는 표준 방식을 정의하여 웹 서비스 간 상호 운용성을 향상시키는 W3C(Worldwide Web Consortium) 스펙입니다. SOAP 메시지는 각각 서로 다른 방식으로 메시지의 대상 정보를 저장하는 HTTP 및 WebSphere MQ를 포함하여 다양한 전송 메커니즘을 통해 보낼 수 있습니다.

WS-Addressing을 사용하도록 구성되는 파이프라인에 배치되는 기존 CICS 웹 서비스는 변경 없이 기본 WS-Addressing 설정을 사용할 수 있습니다. WS-Addressing 기능을 최대한 활용하려면 WS-Addressing API 명령을 사용하십시오. WS-Addressing 구현은 각 WSDL 조작에 대해 한 개의 SOAP 결함을 지원합니다.

메시지 주소 지정 특성

메시지 주소 지정 특성(MAP)은 일련의 잘 정의된 WS-Addressing 특성이며 SOAP 헤더의 요소로 나타낼 수 있습니다. MAP는 메시지 응답이 경로 지정되는 엔드포인트 또는 메시지와 다른 메시지 간 관계 정보와 같은 표준 정보 전달 방식을 제공합니다. WS-Addressing 스펙에 정의된 MAP는 다음 테이블에 요약됩니다.

표 12. WS-Addressing 스펙에 의해 정의되는 메시지 주소 지정 특성

추상 WS-Addressing MAP 이름	SOAP WS-Addressing MAP 이름	MAP 콘텐츠 유형	다중성	설명
[action]	<wsa:Action>	xs:anyURI	1..1	메시지의 시맨틱을 고유하게 식별하는 절대 URI입니다. 이 값은 필수입니다.
[destination]	<wsa:To>	SOAP 메시지의 xs:anyURI 주소 지정 컨텍스트에서 EndpointReference	0..1	메시지의 계획된 수신인 주소를 지정하는 절대 URI입니다. 이 값을 지정하지 않으면 스펙에서 정의되는 익명 URI(http://www.w3.org/2005/08/addressing/anonymous)로 기본값 설정됩니다. 주소 지정 컨텍스트에서 <wsa:To> MAP는 EPR로 표시됩니다. <wsa:To>가 SOAP 메시지의 일부로 보내지는 경우 스키마에서 정의된 대로 해당 주소 및 해당 참조 매개변수로 분할됩니다.
[reference parameters] *	[참조 매개변수]*	xs:any	0..무제한	메시지가 주소 지정되는 엔드포인트 참조의 <wsa:ReferenceParameters> 특성에 해당하는 매개변수입니다. 이 값은 선택사항입니다.
[source endpoint]	<wsa:From>	EndpointReference	0..1	메시지가 시작되는 엔드포인트에 대한 참조입니다. 이 값은 선택사항입니다.

표 12. WS-Addressing 스펙에 의해 정의되는 메시지 주소 지정 특성 (계속)

추상 WS-Addressing MAP 이름	SOAP WS-Addressing MAP 이름	MAP 콘텐츠 유형	다중성	설명
[reply endpoint]	<wsa:ReplyTo>	EndpointReference	0..1	이 메시지에 대한 응답의 계획된 수신인에 대한 엔드포인트 참조입니다. 이 값은 선택사항입니다. 이 값을 지정하지 않으면 http://www.w3.org/2005/08/addressing/anonymous로 기본값 설정됩니다.
[fault endpoint]	<wsa:FaultTo>	EndpointReference	0..1	이 메시지에 관련된 결함의 계획된 수신인에 대한 엔드포인트 참조입니다. 이 값은 선택사항이고 <wsa:ReplyTo> MAP의 값으로 기본값 설정됩니다.
[relationship] *	<wsa:RelatesTo>	xs:anyURI에 xs:anyURI 유형의 선택적 속성 추가	0..무제한	이 메시지가 다른 메시지에 관련된 방식을 나타내는 값 쌍입니다. 이 요소의 콘텐츠는 관련 메시지의 <wsa:MessageID>를 전달합니다. 선택적 속성은 관계 유형을 전달합니다. 이 값은 선택사항입니다. 이 값을 지정하지 않으면 http://www.w3.org/2005/08/addressing/reply로 기본값 설정됩니다.
[message id]	<wsa:MessageID>	xs:anyURI		메시지를 고유하게 식별하는 절대 URI입니다. 이 값은 선택사항입니다. 값을 제공하지 않으면 CICS가 아웃바운드 요청 및 응답에 대한 값을 생성합니다.

다음 예제 SOAP 메시지는 WS-Addressing MAP를 포함합니다.

```
<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>
```

엔드포인트 참조

엔드포인트 참조는 특정 엔드포인트에 대한 정보를 캡슐화하기 위한 표준 메커니즘을 제공하는 특정 유형의 MAP입니다. 엔드포인트 참조는 다른 당사자에게 보낼 수 있으며 엔드포인트 참조가 나타내는 웹 서비스 엔드포인트를 대상화하는 데 사용할 수 있습니다. 다음 테이블은 엔드포인트 참조에 대한 정보 모델을 요약합니다.

표 13. 엔드포인트 참조에 대한 정보 모델

추상 특성 이름	특성 유형	다중성	설명
[address]	xs:anyURI	1..1	엔드포인트의 주소를 지정하는 절대 URI입니다.
[reference parameters] *	xs:any	0..무제한	엔드포인트와 인터페이스하는 데 필요한 네임스페이스로 규정된 요소 정보 항목입니다.
[metadata]	xs:any	0..무제한	엔드포인트의 동작, 정책 및 기능에 대한 설명입니다.

다음 XML 단편은 엔드포인트 참조를 나타냅니다. <wsa:EndpointReference> 요소는 URI `http://example.ibm.com/enquiry`에서 엔드포인트를 참조하고 엔드포인트 참조가 참조하는 인터페이스를 지정하는 메타데이터 및 일부 애플리케이션별 참조 매개변수를 포함합니다.

```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
  xmlns:example="http://example.ibm.com/namespace">
  <wsa:Address>http://example.ibm.com/enquiry</wsa:Address>
  <wsa:Metadata
    xmlns:wsdli="http://www.w3.org/ns/wsdli-instance"
    wsdl:wsdlLocation="http://example.ibm.com/wsdli/wsdli-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>
```

wsa:EndpointReferenceType 유형의 WS-Addressing MAP는 <wsa:From>, <wsa:ReplyTo> 및 <wsa:FaultTo>입니다. 그러나 <wsa:To> MAP는 유형이 xs:anyURI인 것으로 WS-Addressing 1.0 표준에서 정의됩니다. 단순성을 위해 CICS는 주소 지정 컨텍스트에서 <wsa:To> MAP를 EPR로 다룹니다. <wsa:To> MAP가 SOAP 메시지의 일부로 전송되면 CICS는 표준에서 요구하는 대로 이를 해당 주소 및 참조 매개변수로 분할합니다.

기본 네임스페이스

다음 접두부 및 해당 네임스페이스는 WS-Addressing 문서 전체에서 참조됩니다.

표 14. 접두부 및 해당 네임스페이스

기본 접두부	네임스페이스
xs	<code>http://www.w3.org/2001/XMLSchema</code>
wsa	<code>http://www.w3.org/2005/08/addressing</code> (권장 스키마) <code>http://schemas.xmlsoap.org/ws/2004/08/addressing</code> (제출 스키마)
wsam	<code>http://www.w3.org/2007/05/addressing/metadata</code>

관련 참조:

361 페이지의 『DFHWS-URI 컨테이너』

DFHWS-URI는 서비스 URI를 포함하는 DATATYPE(CHAR) 컨테이너입니다.

WS-Addressing을 위한 요청자 파이프라인 구성

웹 서비스 주소 지정(WS-Addressing)을 지원하도록 요청자 파이프라인을 구성하려면 파이프라인 구성 파일에 주소 지정 핸들러를 추가해야 합니다.

시작하기 전에

WS-Addressing에 대한 구성 정보를 추가하려면 파이프라인 구성 파일을 식별하거나 작성해야 합니다. 사용할 WS-Addressing 스펙도 결정해야 합니다. 가능한 경우 W3C *WS-Addressing 1.0* 코어 스펙을 사용하십시오.

이 태스크 정보

다음 두 가지 방식 중 하나로 WS-Addressing에 대한 지원을 추가할 수 있습니다.

- SOAP 파이프라인이 Java를 사용하는 경우 SOAP 처리는 Axis2에 의해 처리되고 이 기술에서 제공하는 지원을 사용하여 WS-Addressing을 사용하는 요청을 처리할 수 있습니다. 모든 헤더 핸들링은 Axis2에 의해 처리되며 파이프라인에 DFHWSADH 헤더 처리 프로그램을 추가하지 않는 것이 중요합니다. 자체 헤더 처리 프로그램을 사용할 수 있습니다. SOAP 헤더를 처리하려는 경우 더 나은 성과를 얻으려면 Java로 Axis2 핸들러를 작성하십시오.
- SOAP 파이프라인이 Java를 사용하지 않는 경우 CICS 제공 헤더 처리 프로그램인 DFHWSADH를 추가하여 요청을 처리해야 합니다.

프로시저

- SOAP 파이프라인이 <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소를 사용하는 경우 파이프라인 구성 파일에 <addressing> 요소를 추가하십시오. 요청 메시지에 사용할 스펙을 포함하는 한 개의 <namespace> 요소를 포함시키십시오. 이 스펙은 응답 메시지에 대한 스펙과 다를 수 있습니다. 예를 들어, 응답 메시지가 제출 스펙을 사용한다고 해도 W3C 코어 스펙을 준수하는 요청을 항상 보낼 수 있습니다. Axis2는 인바운드 메시지에 대해 두 WS-Addressing 스펙 모두 지원합니다.

다음 예제는 요청자 파이프라인을 구성하는 방법을 보여줍니다.

```
<requester_pipeline>
  <service>
    <service_handler_list>
      <cics_soap_1.1_handler_java>
        <jvmserver>JVMSERV1</jvmserver>
      <addressing>
        <namespace>http://www.w3.org/2005/08/addressing</namespace>
      </addressing>
```

```

        </cics_soap_1.1_handler_java>
    </service_handler_list>
</service>
</requester_pipeline>

```

<jvmserver> 요소는 Axis2를 지원하는 JVMSERVER 자원의 이름을 포함합니다.

- SOAP 파이프라인이 Java를 사용하지 않는 경우 <cics_soap_1.1_handler> 또는 <cics_soap_1.2_handler>에서 CICS 주소 지정 헤더 프로그램을 파이프라인 구성 파일에 추가하십시오. 다음 예제는 요청자 파이프라인을 구성하는 방법을 보여줍니다.

```

<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>

```

<program_name>, <localname> 및 <mandatory> 요소를 표시된 그대로 코드화하십시오. <namespace>를 <http://www.w3.org/2005/08/addressing>으로 설정하여 W3C WS-Addressing 1.0 코어 스펙을 사용하거나 <http://schemas.xmlsoap.org/ws/2004/08/addressing>으로 설정하여 W3C WS-Addressing 제출 스펙을 사용하십시오.

헤더 처리 프로그램의 순서는 보장되지 않습니다. 다른 헤더 처리 프로그램을 정의하는 경우 정의한 프로그램을 <service_handler_list> 요소에 있는 후속 CICS SOAP 핸들러 요소에서 추가하십시오. DFHWSADH 헤더 핸들러는 첫 번째 SOAP 핸들러 요소에 있어야 합니다.

결과

요청자 파이프라인이 WS-Addressing을 지원하도록 구성되었습니다.

다음에 수행할 작업

구성 파일을 가리키는 PIPELINE 자원을 작성하십시오. Java 기반 SOAP 파이프라인을 사용하는 경우 JVMSERVER 자원이 Axis2 처리를 처리할 수 있는지 확인하십시오.

WS-Addressing을 위한 제공자 파이프라인 구성

웹 서비스 주소 지정(WS-Addressing)을 지원하도록 제공자 파이프라인을 구성하려면 파이프라인 구성 파일에 주소 지정 핸들러를 추가해야 합니다.

시작하기 전에

WS-Addressing에 대한 구성 정보를 추가하려면 파이프라인 구성 파일을 식별하거나 작성해야 합니다. 사용할 WS-Addressing 스펙도 결정해야 합니다. 가능한 경우 W3C *WS-Addressing 1.0* 코어 스펙을 사용하십시오.

이 태스크 정보

다음 두 가지 방식 중 하나로 WS-Addressing에 대한 지원을 추가할 수 있습니다.

- SOAP 파이프라인이 Java를 사용하는 경우 SOAP 처리는 Axis2에 의해 처리되고 이 기술에서 제공하는 지원을 사용하여 WS-Addressing을 사용하는 요청을 처리할 수 있습니다. 모든 헤더 핸들링은 Axis2에 의해 처리되며 파이프라인에 DFHWSADH 헤더 처리 프로그램을 추가하지 않는 것이 중요합니다. 자체 헤더 처리 프로그램을 사용할 수 있습니다. SOAP 헤더를 처리하려는 경우 더 나은 성과를 얻으려면 Java로 Axis2 핸들러를 작성하십시오.
- SOAP 파이프라인이 Java를 사용하지 않는 경우 CICS 제공 헤더 처리 프로그램인 DFHWSADH를 추가하여 요청을 처리해야 합니다.

프로시저

- SOAP 파이프라인이 <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소를 사용하는 경우 파이프라인 구성 파일에 <addressing> 요소를 추가하십시오. 선택적으로 한 개 이상의 <namespace> 요소를 포함시킬 수 있습니다. 이 요소에는 아웃바운드 메시지에 사용할 스펙이 포함되며 이 스펙은 인바운드 메시지에 대한 스펙과 다를 수 있습니다. 예를 들어, 인바운드 메시지가 제출 스펙을 사용한다고 해도 W3C 코어 스펙을 준수하는 아웃바운드 응답을 항상 보낼 수 있습니다. 이 요소를 제외시키면 Axis2는 인바운드 메시지와 같은 스펙을 아웃바운드 메시지에 사용합니다. Axis2는 인바운드 메시지에 대해 두 WS-Addressing 스펙 모두 지원합니다.

다음 예제는 제공자 파이프라인을 구성하는 방법을 보여줍니다.

```
<provider_pipeline>
  <terminal_handler>
    <cics_soap_1.1_handler_java>
      <jvmserver>JVMSEV1</jvmserver>
      <addressing>
        <namespace>http://www.w3.org/2005/08/addressing</namespace>
      </addressing>
    </cics_soap_1.1_handler_java>
  </terminal_handler>
</provider_pipeline>
```

<jvmserver> 요소는 Axis2를 지원하는 JVMSERVER 자원의 이름을 포함합니다.

- SOAP 파이프라인이 Java를 사용하지 않는 경우 CICS 주소 지정 헤더 프로그램인 DFHWSADH를 파이프라인 구성 파일의 SOAP 핸들러에 추가하십시오. 다음 예제는 제공자 파이프라인을 구성하는 방법을 보여줍니다.

```
<provider_pipeline>
  <terminal_handler>
    <cics_soap_1.1_handler>
      <headerprogram>
        <program_name>DFHWSADH</program_name>
        <namespace>http://www.w3.org/2005/08/addressing</namespace>
        <localname>*</localname>
        <mandatory>true</mandatory>
      </headerprogram>
    </cics_soap_1.1_handler>
  </terminal_handler>
</provider_pipeline>
```

<program_name>, <localname> 및 <mandatory> 요소를 표시된 그대로 코드화하십시오. <namespace>를 http://www.w3.org/2005/08/addressing으로 설정하여 W3C WS-Addressing 1.0 코어 스펙을 사용하거나 http://schemas.xmlsoap.org/ws/2004/08/addressing으로 설정하여 W3C WS-Addressing 제출 스펙을 사용하십시오.

헤더 처리 프로그램의 순서는 보장되지 않습니다. 다른 헤더 처리 프로그램을 정의하는 경우 정의한 프로그램을 <service_handler_list> 요소에 있는 다른 CICS SOAP 핸들러 요소에서 추가하십시오. DFHWSADH 헤더 핸들러는 마지막 SOAP 핸들러 요소에 있어야 합니다.

결과

제공자 파이프라인이 WS-Addressing을 지원하도록 구성되었습니다.

다음에 수행할 작업

구성 파일을 가리키는 PIPELINE 자원을 작성하십시오. Java 기반 SOAP 파이프라인을 사용하는 경우 JVMSERVER 자원이 Axis2 처리를 처리할 수 있는지 확인하십시오.

WS-Addressing을 사용하는 웹 서비스 작성

WSDL에서 웹 서비스 주소 지정(WS-Addressing)을 사용하는 웹 서비스를 작성하려면 웹 서비스 지원에서 매개변수를 사용하여 XML에서 언어 구조로의 변환을 처리하십시오.

이 태스크 정보

웹 서비스 지원 작업 DFHWS2LS를 사용하여 엔드포인트 참조(EPR)가 WSDL 문서에서 처리되는 방식을 제어하고 CICS가 기본 입력, 출력 및 결합 조치를 구성하는지 여부를 판별할 수 있습니다.

프로시저

1. 웹 서비스 지원 DFHWS2LS에서 **MINIMUM-RUNTIME** 매개변수를 3.0 이상으로 설정하십시오. 최소 3.0 이상의 런타임 레벨은 생성된 웹 서비스 바인딩이 WS-Addressing을 지원하고 다른 웹 서비스 플랫폼과 상호 운용할 수 있도록 합니다.
2. 웹 서비스 지원 DFHWS2LS에서 **MAPPING-LEVEL** 매개변수를 3.0 이상으로 설정하십시오.
3. 요청 또는 응답 메시지에서 `wsa:EndpointReferenceType` 유형 요소를 사용하려면 **WSADDR-EPR-ANY** 매개변수를 TRUE로 설정하십시오. 엔드포인트 참조는 애플리케이션 데이터에 포함될 수 있고 **WSACONTEXT BUILD**와 같은 API 명령에서 EPR을 사용하는 옵션이 있습니다. **WSADDR-EPR-ANY** 매개변수를 TRUE로 설정하는 것은 CICS가 런타임 시 EPR을 언어 구조로 변환하지 말아야 하며 대신 CICS가 EPR 데이터를 `<xsd:any>` 요소로 처리한 후 이름 지정된 컨테이너에 저장해야 함을 나타냅니다.

이 예제 WSDL 단편은 `wsa:EndpointReferenceType` 유형의 요소로 전달되고 있는 `<wsa:To>` MAP를 보여줍니다.

```
<?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>
```

`<xs:element name="myEpr" type="wsa:EndpointReferenceType"/>` **1** 요소가 **WSADDR-EPR-ANY** 매개변수가 TRUE로 설정된 DFHWS2LS에 의해 처리될 때 myEpr 요소 데이터는 `<xsd:any>` 요소이자 생성된 언어 구조에 추가된 컨테이너에 대한 포인터로 이름 지정된 컨테이너에 저장됩니다.

예를 들어, myEpr 요소에 대해 DFHWS2LS에서 생성되는 COBOL 언어 구조는 다음과 같습니다.

```
09 myEpr.
   12 myEpr-xml-cont          PIC X(16).
   12 myEpr-xmlns-cont        PIC X(16).
```

myEpr-xml-cont 컨테이너는 myEpr 데이터를 포함하는 컨테이너의 이름을 저장합니다. myEpr-xmlns-cont는 범위 내 XML 네임스페이스 선언으로 채워지는 선택적 컨테이너입니다.

4. DFHWS2LS 작업을 저장하고 제출하십시오.

결과

CICS는 웹 서비스 바인딩을 작성하여 서비스 요청자 또는 제공자 애플리케이션을 작성하는 데 사용할 수 있는 언어 구조 및 데이터 변환을 처리합니다.

다음에 수행할 작업

웹 서비스를 사용으로 설정하려면 파이프라인 스캔을 수행하여 필수 CICS 자원을 작성하십시오.

기본 엔드포인트 참조

대부분의 WSDL 문서에는 웹 서비스가 호스팅되는 주소가 포함되어 있습니다. WS-Addressing에서 WSDL 문서는 웹 서비스의 엔드포인트 참조(EPR)도 포함할 수 있습니다. 이 EPR은 요청자와 제공자 애플리케이션 사이의 통신을 용이하게 하는 추가 메타데이터도 포함할 수 있습니다.

DFHWS2LS를 사용하여 WSDL을 처리하는 경우 EPR은 웹 서비스 바인딩에 저장되어 요청 및 응답 메시지를 보내기 위해 CICS에서 사용됩니다. EPR에서 정의되는 모든 참조 매개변수 `<wsa:ReferenceParameters>`는 SOAP 메시지에 포함됩니다. 이 EPR은 애플리케이션에 의해 대체될 수 있기 때문에 기본 EPR이라고 합니다. 애플리케이션이 명시적 EPR을 제공하지 않는 경우 WSDL의 기본 EPR이 사용됩니다.

다음 WSDL 1.1 단편에는 기본 EPR(`<soap:address location="http://example.ibm.com:12345/exampleTest" />`)이 포함됩니다. `<port>` 요소는 하위 요소인 `<wsa:EndpointReference>`를 포함하여 하위 요소 **2**에 의해 지정되는 주소는 상위 요소 **1**에서 지정된 주소와 일치해야 합니다.


```

<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>

```

명시적 조치

WSDL 문서는 <wsa:Action> 특성의 값을 명시적으로 정의할 수 있습니다. WSDL 문서에 명시적으로 정의된 <wsa:Action> 특성이 없을 경우 CICS는 WSDL이 DFHWS2LS에 의해 처리될 때 기본 조치를 빌드합니다.

WSDL 1.1

다음 WSDL 1.1 단편은 명시적으로 정의된 <wsa:Action> 특성을 포함하는 예약 시스템을 나타냅니다.

```

<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>

```

이 예제에서 makeBooking 조작의 입력 조치는 http://example.ibm.com/namespace/makeBooking으로 명시적으로 정의되고 출력 조치는 http://example.ibm.com/namespace/bookingResponse로 명시적으로 정의됩니다.

WSDL 2.0

다음 WSDL 2.0 단편은 명시적으로 정의된 <wsa:Action> 특성을 포함하는 예약 시스템을 나타냅니다.

```

<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>

```

이 예제에서 makeBooking 조작의 입력 조치는 http://example.ibm.com/namespace/makeBooking으로 명시적으로 정의되고 출력 조치는 http://example.ibm.com/namespace/makeBookingResponse로 정의됩니다.

자세한 정보는 W3C WS-Addressing 1.0 메타데이터 스펙을 참조하십시오.

WSDL 1.1의 기본 조치

WSDL 1.1 문서에 명시적으로 지정된 <wsa:Action> 특성이 포함되지 않은 경우 CICS는 WSDL이 DFHWS2LS에 의해 처리될 때 기본 입력, 출력 및 기본 조치를 빌드합니다.

WSDL 1.1의 기본 입력 및 출력 조치

다음 패턴은 권장 스키마 또는 제출 스키마에 따라 기본 입력 또는 출력 조치를 구성하는 WSDL 1.1 문서에서 CICS에 의해 사용됩니다.

```
[target namespace]/[port type name]/[input|output name]
```

WSDL 1.1의 기본 결함 조치

권장 스키마를 따르는 경우 CICS가 기본 결함 조치를 빌드하는 방식은 스키마에서 설명한 동작과는 다릅니다. 다음 패턴은 권장 스키마에 따라 기본 결함 메시지를 구성하는 WSDL 1.1 문서에서 CICS에 의해 사용됩니다. 결함 이름은 생략됨에 유의하십시오.

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

제출 스키마를 따르는 경우 CICS가 기본 결함 조치를 빌드하는 방식은 스키마에서 설명한 동작을 따릅니다. 다음 패턴은 제출 스키마에 따라 기본 결함 메시지를 구성하는 WSDL 1.1 문서에서 CICS에 의해 사용됩니다.

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

CICS에 의해 생성되는 WSDL 1.1 문서의 기본 조치 예제

이 예약 시스템 예제는 CICS가 WSDL 1.1 문서에서 기본 조치를 구성하는 방법을 보여줍니다.

```

<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"/>
    </operation>
  </portType>
</description>

```

```

        <fault message="tns:InvalidBooking" name="InvalidBooking"/>
    </operation>
</interface>
...
</definitions>

```

WSDL 단편은 다음과 같은 주소 지정 특성을 가집니다.

특성 이름	값
[대상 네임스페이스]	http://example.ibm.com/namespace
[portType name]	bookingInterface
[조작 이름]	makeBooking
[입력 이름]	MakeBooking
[출력 이름]	BookingResponse
[결함 이름]	InvalidBooking

이러한 값에서 다음 조치가 작성됩니다.

조치	값
입력 조치	http://example.ibm.com/namespace/bookingInterface/MakeBooking [입력 이름]을 지정하지 않으면 "요청"이 추가된 [조작 이름]의 값이 대신 사용됩니다. 예를 들어, 이 경우에 입력 조치는 http://example.ibm.com/namespace/bookingInterface/makeBookingRequest입니다. http://example.ibm.com/namespace/bookingInterface/BookingResponse
출력 조치	[출력 이름]을 지정하지 않으면 "응답"이 추가된 [조작 이름]의 값이 대신 사용됩니다. 예를 들어, 이 경우에 출력 조치는 http://example.ibm.com/namespace/bookingInterface/makeBookingResponse입니다.
결함 조치 (권장 스키마)	http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/ [결함 이름]은 생략됨에 유의하십시오.
결함 조치 (제출 스키마)	http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/ InvalidBooking

자세한 정보는 W3C WS-Addressing 1.0 메타데이터 스펙을 참조하십시오.

WSDL 2.0의 기본 조치

WSDL 2.0 문서에 명시적으로 지정된 <wsa:Action> 특성이 포함되지 않은 경우 CICS는 WSDL이 DFHWS2LS에 의해 처리될 때 기본 입력, 출력 및 기본 조치를 빌드합니다.

WSDL 2.0의 기본 입력 및 출력 조치

다음 패턴은 권장 스키마에 따라 입력 및 출력에 대한 기본 조치를 구성하는 WSDL 2.0 문서에서 CICS에 의해 사용됩니다.

[target namespace]/[interface name]/[operation name][direction token]

WSDL 2.0의 기본 결함 조치

권장 스키마를 따르는 경우 CICS가 WS-Addressing 결함에 대한 기본 조치를 빌드하는 방식은 스키마에서 설명한 동작과는 다릅니다. 제출 스키마를 따르는 경우 CICS가 WS-Addressing 결함에 대한 기본 조치를 빌드하는 방식은 스키마에서 설명한 동작을 따릅니다.

다음 패턴은 권장 스키마에 따라 결함에 대한 기본 조치를 구성하는 WSDL 2.0 문서에서 CICS에 의해 사용됩니다. 결함 이름은 생략됨에 유의하십시오.

```
[target namespace]/[interface name]/
```

다음 패턴은 제출 스키마에 따라 결함에 대한 기본 조치를 구성하는 WSDL 2.0 문서에서 CICS에 의해 사용됩니다.

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

CICS에 의해 생성되는 WSDL 2.0 문서의 기본 조치 예제

이 예제는 CICS가 권장 스키마에 따라 WSDL 2.0 문서의 기본 조치를 구성하는 방법을 보여줍니다.

```
<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"/>
    <output element="tns:bookingResponse" messageLabel="Out"/>
  </operation>
</interface>
...
</definitions>
```

WSDL 단편은 다음과 같은 주소 지정 특성을 가집니다.

특성 이름	값
[대상 네임스페이스]	http://example.ibm.com/namespace
[인터페이스 이름]	bookingInterface
[조작 이름]	makeBooking
[방향 토큰]	Request 또는 Response.

이러한 값에서 다음 입력 및 출력조치가 작성됩니다.

조치	값
입력 조치	http://example.ibm.com/namespace/bookingInterface/makeBookingRequest
출력 조치	http://example.ibm.com/namespace/bookingInterface/makeBookingResponse

자세한 정보는 W3C WS-Addressing 1.0 메타데이터 스펙을 참조하십시오.

메시지 교환

웹 서비스 주소 지정(WS-Addressing)은 다음의 메시지 교환을 지원합니다. 단방향, 양방향요청- 응답, 동기 요청-응답 및 비동기 요청-응답.

WS-Addressing 메시지 교환에는 메시지 주소 지정 특성(MAP) 및 엔드포인트 참조(EPR)가 포함됩니다.

런타임 시 CICS는 요청 메시지의 SOAP 헤더에 관련 WS-Addressing 메시지 정보가 포함되는지 확인하고 요청자 애플리케이션은 WS-Addressing 헤더를 설정하지 않아도 되며 WS-Addressing을 사용 중인지 인식하지 못할 수도 있습니다.

단방향

이 단순한 단방향 메시지는 입력 전용 조작으로 정의됩니다. 이 조작에 대한 WSDL(Web Services Description Language)의 양식은 다음과 같습니다.

```
<operation name="myOperation">
  <input message="tns:myInputMessage"/>
</operation>
```

WS-Addressing을 사용 중인 경우 CICS는 런타임 시 <wsa:Action> MAP 및 <wsa:MessageID> MAP를 WS-Addressing 요청 메시지의 SOAP 메시지 헤더에 추가하여 WS-Addressing 스펙 준수를 보장합니다.

<wsa:MessageID> MAP는 고유 ID이며 지정하지 않은 경우 CICS가 이 ID를 자동으로 생성합니다.

<wsa:Action> MAP는 WSDL에서 파생되고 WSBind 파일로 저장됩니다.

CICS WS-Addressing API 명령을 사용하여 이러한 MAP의 값을 대체할 수 있습니다.

양방향 요청-응답

이 양방향 교환에는 요청 메시지 및 응답 메시지가 포함됩니다. 조작의 응답 파트는 출력 메시지, 결함 메시지 또는 둘 다로 정의할 수 있습니다. 요청-응답 조작에 대한 WSDL 정의의 양식은 다음과 같습니다.

```
<operation name="myOperation">
  <input message="tns:myInputMessage"/>
  <output message="tns:myOutputMessage"/>
  <fault="tns:myFaultMessage"/>
</operation>
```

응답 대상 또는 결함 생성 위치, 엔드포인트로 경로 지정된 요청은 응답 유형이 보통인지 또는 결함인지에 따라 <wsa:ReplyTo> MAP 또는 the <wsa:FaultTo> MAP를 대상으로 합니다.

요청 메시지에서 <wsa:ReplyTo> 또는 <wsa:FaultTo> MAP를 지정하여 응답을 전송해야 하는 위치를 표시하십시오.

권장 스펙을 사용하고 <wsa:ReplyTo> MAP의 값을 지정하지 않으면 <wsa:ReplyTo> MAP는 익명 URI(<http://www.w3.org/2005/08/addressing/anonymous>)를 포함하는 엔드포인트 참조로 기본값 설정되고 이로 인해 CICS는 요청자에게 응답을 다시 보냅니다.

권장 스펙을 사용하고 <wsa:FaultTo> MAP의 값을 지정하지 않으면 <wsa:FaultTo> MAP가 <wsa:ReplyTo> MAP의 값으로 기본값 설정됩니다.

요청자가 올바르지 않은 MAP를 빌드하고 이로 인해 유효성 검증이 실패하면 CICS는 <wsa:FaultTo> MAP에 의해 지정된 주소 대신 요청자에게 다시 결합 메시지를 보냅니다.

동기 요청-응답

기본적으로 양방향 메시지의 응답 파트는 사용 중인 기본 프로토콜에 따라 리턴됩니다. HTTP 요청의 경우 응답은 HTTP 응답에서 동시에 리턴됩니다.

비동기 요청-응답

비동기 응답은 다른 웹 서비스를 대상으로 하고 원래 요청자 애플리케이션에 다시 돌아오지 않습니다. HTTP 요청의 경우 요청 클라이언트와의 연결은 HTTP 202 응답과 함께 닫힙니다. 웹 서비스 제공자가 CICS 시스템을 실행 중인 경우 요청자 애플리케이션은 비어 있는 응답 메시지를 수신합니다. 웹 서비스 제공자가 WebSphere MQ 시스템에서 실행되고 있는 경우 요청자 애플리케이션은 어떠한 응답도 수신하지 않습니다.

양방향 메시지 응답 파트의 대상을 변경하려면 <wsa:ReplyTo> MAP 또는 <wsa:ReplyTo> 및 <wsa:FaultTo>, MAP에 적절한 주소를 지정해야 합니다.

WSDL 1.1과 WSDL 2.0에서 필수인 MAP의 전체 목록은 428 페이지의 『WS-Addressing의 필수 메시지 주소 지정 특성』의 내용을 참조하십시오.

관련 개념:

36 페이지의 『WSDL 및 메시지 교환 패턴』

WSDL 2.0 문서에는 웹 서비스 요청자와 웹 서비스 제공자 간에 SOAP 1.2 메시지가 교환되는 방식을 정의하는 메시지 교환 패턴(MEP)이 포함됩니다.

관련 참조:

428 페이지의 『WS-Addressing의 필수 메시지 주소 지정 특성』

WS-Addressing 1.0 메타데이터 스펙은 WSDL 1.1 및 WSDL 2.0 문서에 포함시켜야 하는 메시지 주소 지정 특성(MAP)을 명시합니다. WS-Addressing의 CICS 구현은 이러한 필수 MAP에 대한 값을 자동으로 제공하여 WS-Addressing 스펙을 준수할 수 있도록 돕습니다.

WS-Addressing의 필수 메시지 주소 지정 특성

WS-Addressing 1.0 메타데이터 스펙은 WSDL 1.1 및 WSDL 2.0 문서에 포함시켜야 하는 메시지 주소 지정 특성(MAP)을 명시합니다. WS-Addressing의 CICS 구현은 이러한 필수 MAP에 대한 값을 자동으로 제공하여 WS-Addressing 스펙을 준수할 수 있도록 돕습니다.

제공하는 WSDL에서 MAP에 사용자 고유 값을 지정하고 CICS WS-Addressing API 명령을 사용하여 이러한 값을 주소 지정 컨텍스트에서 갱신할 수 있습니다. 필수 MAP에 대한 값을 제공하지 않으면 CICS가 값을 생성하게 됩니다.

다음 테이블은 WSDL 1.1 및 WSDL 2에서 지원되는 여러 메시지 교환 패턴(MEP)에 대해 필수인 MAP를 나열합니다.

표 15. WS-Addressing의 필수 메시지 주소 지정 특성

WS-Addressing MAP 이름	설명	WSDL 1.1에서 필수	WSDL 2.0에서 필수
<wsa:To>	메시지의 계획된 수신인 주소입니다.	아니오	아니오
<wsa:Action>	WS-Addressing 조치: 입력, 출력 또는 결합	다음 MEP에 대해 필수임: 단방향 양방향(요청) 양방향(응답)	다음 MEP에 대해 필수임: In-only Robust In-only(In) Robust In-only(결합) In-out(In) In-out(Out) In-optional-out(In) In-optional-out(Out)
<wsa:From>	메시지가 시작되는 엔드포인트입니다.	아니오	아니오
이 값은 필수가 아님			
<wsa:ReplyTo>	메시지에 대한 응답의 계획된 수신인의 엔드포인트입니다.	아니오	아니오
<wsa:FaultTo>	메시지에 관련된 결합에 대한 계획된 수신인의 엔드포인트입니다.	아니오	아니오
<wsa:MessageID>	고유 메시지 ID입니다.	다음 MEP에 대해 필수임: 양방향(요청)	다음 MEP에 대해 필수임: Robust In-only(In) In-out(In) In-optional-out(In)
<wsa:RelatesTo>	이 메시지가 다른 메시지에 관련된 방식을 나타내는 값 쌍입니다. 이 요소에는 관련된 메시지의 <wsa:MessageID>가 포함되고 선택적 속성은 관계 유형을 전달합니다.	다음 MEP에 대해 필수임: 양방향(응답)	다음 MEP에 대해 필수임: Robust In-only(결합) In-out(Out) In-optional-out(Out)

자세한 정보는 W3C WS-Addressing 1.0 메타데이터 스펙(<http://www.w3.org/TR/ws-addr-metadata>)을 참조하십시오.

참고:

- <wsa:ReplyTo> MAP의 주소 요소에 대해 값이 설정되지 않은 경우 주소는 익명 URI <http://www.w3.org/2005/08/addressing/anonymous>로 설정됩니다. 익명 URI는 응답이 요청자에게로 다시 전송되는 것을 나타냅니다.
- <wsa:FaultTo> MAP의 주소 요소에 대해 값을 지정하지 않으면 CICS는 이 주소를 <wsa:ReplyTo> MAP의 주소 요소와 같은 값으로 설정합니다.

요청자가 올바르지 않은 MAP를 빌드하고 이로 인해 유효성 검증이 실패하면 CICS는 <wsa:FaultTo> MAP에 의해 지정된 주소 대신 요청자에게 다시 결합 메시지를 보내는 점에 유의하십시오.

- <wsa:To> MAP의 값을 지정하지 않으면 CICS가 주소를 익명 URI <http://www.w3.org/2005/08/addressing/anonymous>로 설정합니다. 익명 URI는 요청이 DFHWS-URI 컨테이너에서 지정된 주소로 전송됨을 나타내며 자세한 정보는 361 페이지의 『DFHWS-URI 컨테이너』의 내용을 참조하십시오.
- <wsa:Action> MAP를 WSDL 문서에서 명시적으로 정의하거나 CICS가 자동으로 생성하게 할 수 있습니다.
- CICS는 런타임 시 응답을 예상하는 요청 메시지 및 응답 메시지에 대해 <wsa:MessageID> MAP에 자동으로 고유 값을 설정합니다.
- <wsa:RelatesTo> MAP는 응답 메시지에 대해 필수입니다. 메시지의 관계 유형은 선택적이고 <http://www.w3.org/2005/08/addressing/reply>로 기본값 설정됩니다.

관련 개념:

426 페이지의 『메시지 교환』

웹 서비스 주소 지정(WS-Addressing)은 다음의 메시지 교환을 지원합니다. 단방향, 양방향요청-응답, 동기 요청-응답 및 비동기 요청-응답.

관련 참조:

361 페이지의 『DFHWS-URI 컨테이너』

DFHWS-URI는 서비스 URI를 포함하는 DATATYPE(CHAR) 컨테이너입니다.

WS-Addressing 보안

웹 서비스 주소 지정(WS-Addressing)을 사용하여 공용 네트워크에서 이동하는 통신은 적절하게 보호되어야 하고 통신 당사자 간 충분한 수준의 신뢰가 확립되어야 합니다. 통신 보안을 위해 SSL 또는 HTTPS와 같은 전송 레벨 보안을 사용하도록 권장됩니다.

SSL 또는 HTTPS와 같은 전송 레벨 보안은 WS-Addressing 통신 보안을 보장하는 가장 단순한 방법입니다. 전송 레벨 보안을 사용할 수 없는 경우에는 WS-Addressing 메시지 주소 지정 특성에 서명하거나 엔드포인트 참조를 암호화하여 메시지를 보호할 수 있습니다.

CICS는 WS-Addressing 메시지 주소 지정 특성을 포함하는 헤더에 서명할 수 없으며 엔드포인트 참조를 암호화할 수 없습니다. 그러나 CICS는 수신 메시지에 대한 서명을 확인하고 암호화된 헤더를 복호화할 수 있습니다. 서명 및 암호화를 사용하여 통신을 보호하려는 경우 IBM WebSphere DataPower® XML Security Gateway와 같은 외부 보안 게이트웨이를 사용해야 합니다. 자세한 정보는 IBM WebSphere DataPower XML Security Gateway를 참조하십시오.

WS-Addressing 예제

이 예제는 WS-Addressing을 사용하여 메시지를 보내는 회사에 고객이 주문을 할 때 발생하는 프로세스에 대한 상위 레벨 개요를 제공합니다.

전자 컴포넌트를 판매하는 국제 기업은 비즈니스에 WS-Addressing을 사용합니다. 이러한 회사의 인프라는 주문 클라이언트, 분배 서비스 그룹, 이행 서비스 및 구성 서비스로 구성됩니다.

WS-Addressing 사용은 회사에 다음과 같은 이점을 제공합니다.

- WS-Addressing은 메시지 전송을 위한 전송 독립 메커니즘을 제공하고 이 메커니즘은 여러 플랫폼에서 실행되는 웹 서비스 간 상호 운용성을 향상시킵니다. 이 예제에서 회사가 소유하는 분배 서비스는 다양한 플랫폼에서 실행됩니다. WS-Addressing은 간단하게 여러 플랫폼 간의 상호 운용성을 제공합니다. 웹 서비스 요청자 및 제공자는 메시지를 교환하고 있는 서비스가 실행되고 있는 플랫폼에 대해서 알지 못해도 됩니다.
- WS-Addressing은 <wsa:ReplyTo> MAP에서 EPR을 갱신하여 응답 메시지의 대상을 변경하는 데 사용할 수 있습니다. 이 예제에서 이행 서비스는 메시지가 우회되는 분배 서비스를 선택할 때 응답 메시지의 대상을 수정합니다.

회사는 다수의 국가에 여러 분배 센터를 가지고 있으며 이 예제에서 각 분배 센터는 분배 서비스로 표시되고 구성 서비스에 등록됩니다.

이행 서비스는 요청된 항목의 가용성 및 고객과 분배 센터와의 거리를 포함한 다양한 요소를 기준으로 하여 주문을 처리하기에 가장 적합한 분배 서비스를 선택합니다.

주소 지정 정보는 구성 서비스에(서) 전달됩니다. 구성 서비스는 사용 가능한 서비스의 주소를 엔드포인트 참조 양식으로 저장합니다. **WSAEPR CREATE** 명령을 사용하여 EPR을 작성하고 이 EPR을 구성 서비스에 전송하여 새 서비스를 구성 서비스에 등록합니다. 구성 서비스에는 XML의 블록으로 EPR이 필요하기 때문에 **DFHWS2LS의 WSADDR-EPR-ANY** 매개변수를 TRUE로 설정해야 합니다. **WSADDR-EPR-ANY=TRUE** 옵션

을 사용하여 CICS에 EPR을 <xsd:any> 요소로 처리하도록 지시합니다. CICS는 런타임 시 이 EPR을 언어 구조로 변환하는 대신 컨테이너에 배치해야 합니다.

이러한 서비스가 상호 작용하는 방식은 다음 다이어그램에 표시되어 있습니다. 이 다이어그램은 비즈니스 애플리케이션에 관련되었을 수 있지만 테스트에서는 제외된 다른 서비스를 보여줍니다.

- 주문의 상태로 기타 서비스 각각에 의해 갱신할 수 있는 추적 서비스
- 발생하는 결함 메시지를 처리하기 위한 문제점 해결 서비스
- 주문 클라이언트에게 향하는 응답 메시지를 처리하는 주문 클라이언트 콜백 서비스

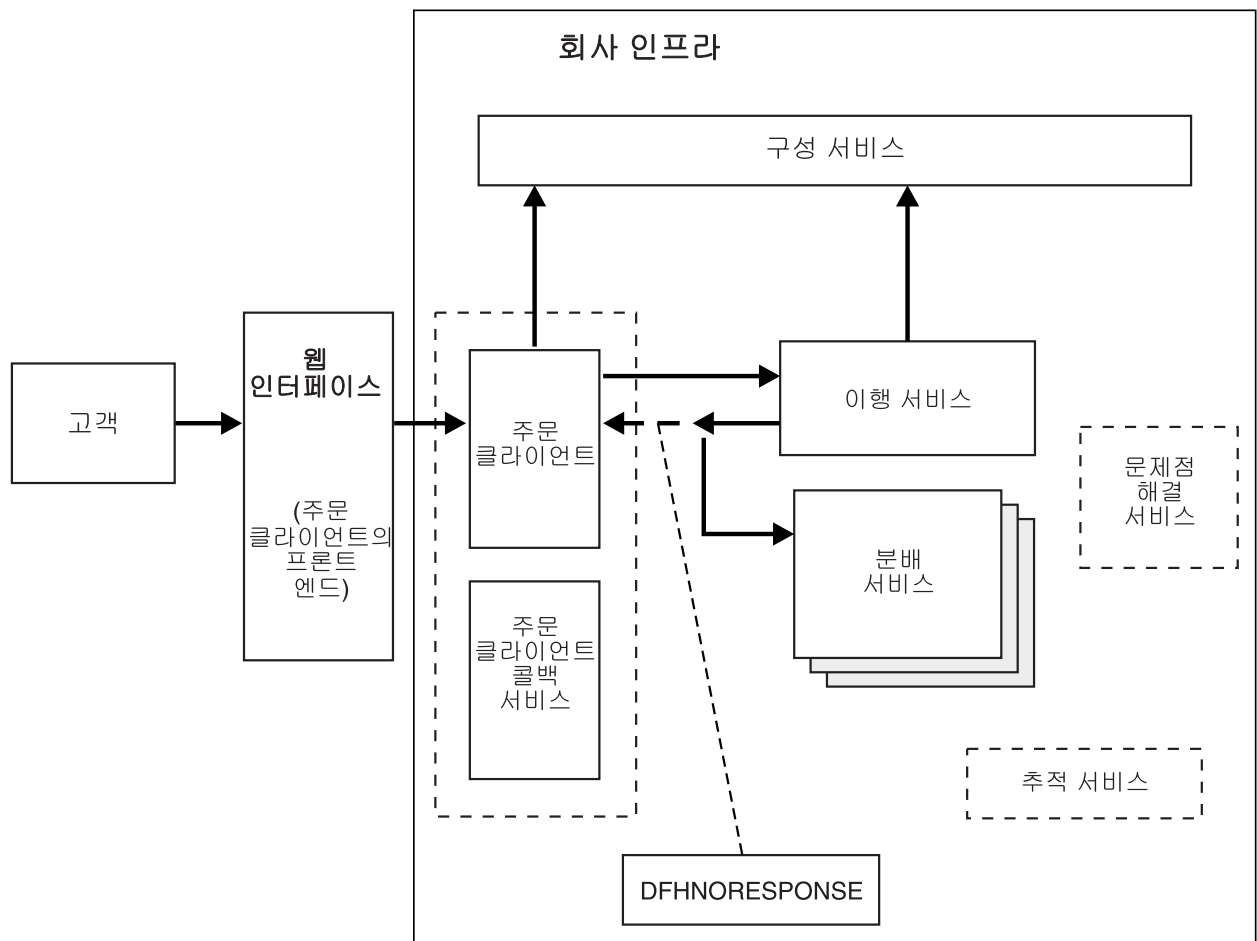


그림 31. 회사 인프라

다음 단계는 고객이 주문을 하는 순간부터 주문이 처리되는 시점까지 발생하는 프로세스를 설명합니다.

1. 고객이 회사에 주문합니다.
 - a. 고객이 주문 클라이언트의 프론트 엔드인 회사 웹 사이트에서 주문합니다.
 - b. 주문 클라이언트가 고객의 접속 세부사항을 주문의 일부로 가져옵니다.

- c. 주문 클라이언트는 웹 인터페이스를 통해 고객에게 확인 및 고유 주문 참조를 리턴합니다.
2. 주문 클라이언트는 이행 서비스에 주문 요청을 보냅니다.
 - a. 이행 서비스의 EPR을 이미 알고 있지 않으면 주문 클라이언트는 구성 서비스에 이를 요청합니다. 주문 클라이언트가 구성 서비스에 이행 서비스의 EPR을 요청할 때 포함되는 프로세스는 <wsa:To>의 예제 섹션에 자세히 설명되어 있습니다.
 - b. 주문 클라이언트는 이행 서비스에 대해 **INVOKE SERVICE** 명령을 발행합니다. WS-Addressing은 메시지를 TO EPR에서 요청 주소 지정 컨텍스트로 지정된 주소로 라우팅합니다.
3. 이행 서비스는 주문을 처리할 분배 서비스를 선택하고 응답 메시지를 해당 서비스로 경로 재지정합니다.
 - a. 이행 서비스는 **WSACONTEXT GET** 명령을 사용하여 주소 지정 컨텍스트로부터 주문 참조 및 기타 주소 지정 특성을 추출합니다.
 - b. 이행 서비스는 구성 서비스에서 가장 적합한 분배 서비스를 선택합니다.
 - c. <wsa:ReplyTo> EPR이 주소 지정 컨텍스트에 추가됩니다.


```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Address>http://www.example.ibm.com/DistributionService</wsa:Address>
</wsa:EndpointReference>
```
 - d. 이행 서비스는 **WSACONTEXT BUILD** 명령을 사용하여 선택한 분배 서비스의 ReplyTo EPR을 요청 주소 지정 컨텍스트에 추가합니다.
 - e. DFHNORESPONSE 컨테이너는 주문 클라이언트 파이프라인에 추가되어 응답을 수신하지 않으며 응답 메시지가 요청 메시지의 양식으로 분배 서비스에 경로 재지정됨을 주문 클라이언트에 표시합니다.
4. 분배 서비스는 경로 재지정된 응답 메시지를 수신하여 주문을 처리합니다.
 - a. 분배 서비스는 **WSACONTEXT GET** 명령을 사용하여 요청 주소 지정 컨텍스트로부터 주문 참조 및 주소 지정 세부사항을 추출합니다.
 - b. 분배 서비스가 주문을 처리합니다.

<wsa:To>의 예제

1. 주문 클라이언트가 메시지를 보낼 서비스의 EPR을 구성 서비스에 요청합니다. 이 예제에서 주문 클라이언트는 이행 서비스의 EPR을 요청합니다.
2. 구성 서비스는 다음과 같이 응답 메시지를 작성하고 보냅니다.
 - a. 구성 서비스는 이행 서비스의 요청된 <wsa:To> EPR을 **WSAEPR CREATE API** 명령 **EXEC CICS WSAEPR CREATE**를 사용하여 작성합니다.

- b. 구성 서비스는 **WSAEPR CREATE** 명령의 출력을 컨테이너 EXEC CICS PUT CONTAINER(work-cont)에 기록합니다.
- c. 구성 서비스는 컨테이너 이름을 myEpr-xml-cont 요소인 MOVE work-cont TO myEpr-xml-cont에 복사합니다.
- d. 구성 서비스는 주문 클라이언트로 응답 메시지를 보내고 이 메시지에는 myEpr-xml-cont 컨테이너로 이름 지정된 컨테이너의 콘텐츠가 포함됩니다. 이 예제에서 work-cont 컨테이너의 콘텐츠는 <wsa:myEpr> 요소 내 주문 클라이언트에게로 전송됩니다.

```
<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>
```

434 페이지의 그림 32에서는 주문 클라이언트와 구성 서비스 간의 요청-응답 메시지 교환을 보여줍니다. 이 메시지 교환에는 두 개의 일반적인 웹 서비스 파이프라인이 포함됩니다.

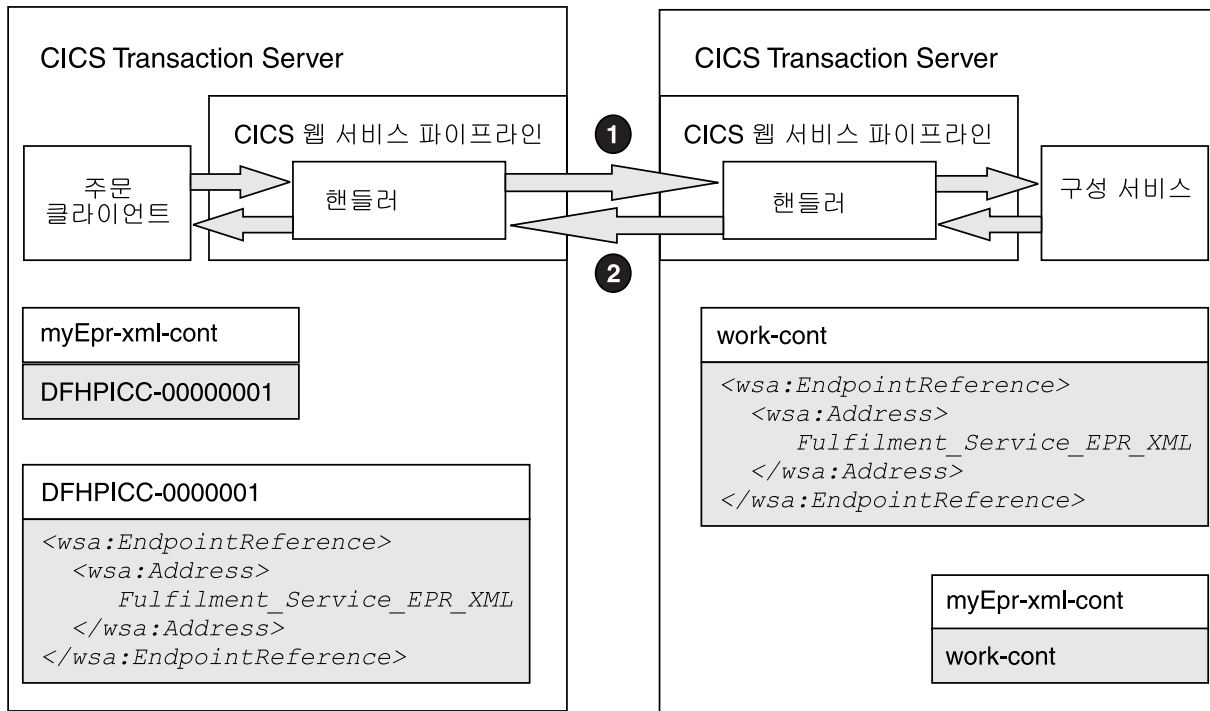


그림 32. 주문 클라이언트와 구성 서비스 간 요청-응답 메시지 교환

3. 주문 클라이언트는 응답 메시지를 수신하고 `<wsa:To>` EPR을 빌드한 다음 이행 서비스에 요청을 보냅니다.
 - a. 주문 클라이언트는 응답 메시지로부터 `<wsa:To>` EPR 데이터를 추출합니다.
 - b. CICS는 고유 컨테이너(이 예제에서는 DFHPICC-000000001 컨테이너임)를 `<wsa:To>` EPR 데이터로 채웁니다.
 - c. CICS는 컨테이너의 이름(이 예제에서는 DFHPICC-000000001임)을 `myEpr-xml-cont` 요소에 복사합니다.
 - d. 주문 클라이언트는 `myEpr-xml-cont` 요소로 지정된 컨테이너의 콘텐츠를 읽고 이 콘텐츠를 **WSACONTEXT BUILD** API 명령에 대한 입력으로 제공합니다. **WSACONTEXT BUILD** 명령은 이 입력을 사용하여 이행 서비스의 `<wsa:To>` EPR을 빌드합니다.
 - e. 주문 클라이언트는 파이프라인 처리를 시작하는 **INVOKE SERVICE** 명령을 발행합니다.
 - f. CICS WS-Addressing 핸들러인 DFHWSADH는 아웃바운드 파이프라인에서 `<wsa:To>` EPR을 주소 및 선택적 참조 매개변수 세트로 변환하고 이행 서비스로 보내고 있는 SOAP 요청 메시지의 헤더에 넣습니다.

```

<env:Header>
  <wsa:To>http://example.ibm.com/Fulfilment_Service</wsa:To>
</env:Header>

```

그림 33에서는 주문 클라이언트에서 이행 서비스로 보내는 요청을 보여줍니다. 이 요청에는 CICS WS-Addressing 핸들러인 DFHWSADH를 포함하는 웹 서비스 파이프라인이 포함됩니다.

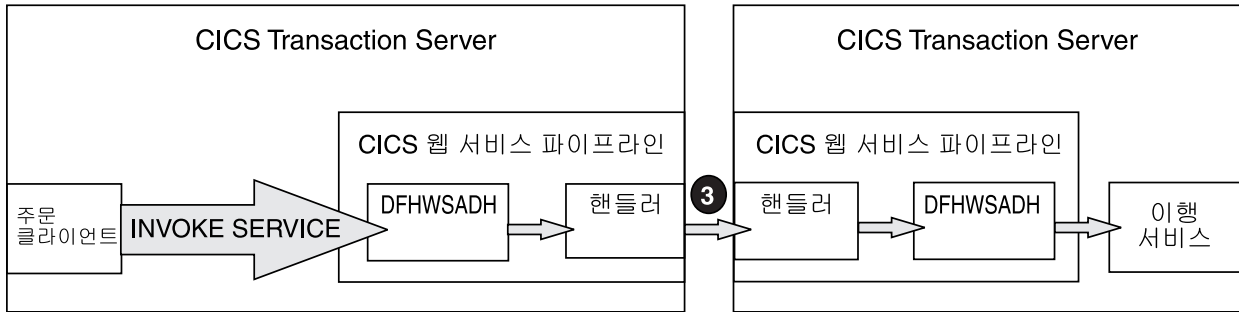


그림 33. 주문 클라이언트에서 이행 서비스로의 요청

관련 참조:

『WS-Addressing 용어』

웹 서비스 주소 지정(WS-Addressing) 지원을 설명하는 데 사용되는 용어입니다.

WS-Addressing 용어

웹 서비스 주소 지정(WS-Addressing) 지원을 설명하는 데 사용되는 용어입니다.

주소 지정 컨텍스트

SOAP 요청 메시지로 보내기 전 및 SOAP 요청 및 응답 메시지에서 수신된 후 WS-Addressing 메시지 주소 지정 특성(MAP)을 저장하는 XML 문서입니다.

엔드포인트 참조(EPR)

메시지를 웹 서비스로 라우팅하는 데 사용되는 주소 지정 정보를 포함하는 XML 구조입니다. 이 주소 지정 정보에는 메시지의 대상 주소, 애플리케이션에서 사용할 선택적 참조 매개변수 및 선택적 메타데이터가 포함됩니다.

메시지 주소 지정 특성(MAP)

고유 메시지 ID, 메시지 대상 및 메시지의 엔드포인트 참조와 같이 특정 웹 서비스 메시지에 대한 주소 지정 정보를 전달하는 XML 요소입니다.

SAML에 대한 지원

CICS는 온라인 비즈니스 파트너 간 보안 정보를 설명하고 교환하기 위한 SAML(Security Assertion Markup Language)의 사용을 지원합니다.

CICS는 SAMLCore1.1 및 SAML Core2.0 표준을 지원합니다. 해당 표준에서 설명되는 프로토콜을 지원하지 않습니다.

SAML 토큰을 사용하도록 제공자 및 요청자 파이프라인을 구성할 수 있지만 먼저 CICS STS(Security Token Service)를 배치해야 합니다. SAML을 지원하도록 CICS 설치를 구성하는 방법에 대한 자세한 정보는 SAML에 대해 CICS 구성을 참조하십시오.

Kerberos에 대해 CICS 웹 서비스 구성

Kerberos 인증을 구현하도록 제공자 파이프라인을 구성하려면 파이프라인 구성 파일에 보안 핸들러를 추가해야 합니다.

시작하기 전에

Kerberos에 대한 지원을 사용하려면 RACF와 같은 외부 보안 관리자를 구성해야 합니다. 자세한 정보는 Kerberos용 RACF 구성의 내용을 참조하십시오.

파이프라인 구성 파일을 식별하거나 작성하여 Kerberos에 대한 구성 정보를 추가해야 합니다.

프로시저

1. 파이프라인에 <wsse_handler> 요소를 추가하십시오. 핸들러는 <service_handler_list> 요소에 포함되어야 합니다. 다음 요소를 코드화하십시오.

```
<wsse_handler>
  <dfhwsse_configuration version="1">
  </dfhwsse_configuration>
</wsse_handler>
```

<dfhwsse_configuration> 요소는 구성에 있는 다른 요소를 위한 컨테이너입니다.

2. <인증> 요소를 코드화하십시오.
 - a. trust="basic"을 지정하도록 **trust** 속성을 코드화하십시오.
 - b. mode="basic-kerberos"를 지정하도록 **mode** 속성을 코드화하십시오.
 - c. 옵션: 비어 있는 <억제/> 요소를 코드화하십시오. 이 요소를 지정하지 않으면 작업이 토큰과 연관된 사용자 ID에서 실행됩니다.

결과

Kerberos 인증에 대해 CICS 제공자 파이프라인이 구성됩니다. 파이프라인으로부터 수신된 인바운드 웹 서비스 요청은 유효한 Kerberos 토큰을 포함해야 합니다. 그렇지 않은 경우 해당 SOAP 결함이 표시되며 요청이 거부됩니다. 파이프라인 구성 옵션에 따라 대상 애플리케이션은 토큰과 연관된 사용자 ID에서 실행됩니다.

예

다음 예제는 제공자 파이프라인을 구성하는 방법을 보여줍니다.

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

제 7 장 SOAP 웹 서비스 작성

기존 CICS 애플리케이션을 SOAP 웹 서비스로 노출하여 SOAP 웹 서비스 제공자 또는 요청자 역할을 할 새 CICS 애플리케이션을 작성할 수 있습니다.

시작하기 전에

SOAP 웹 서비스 작성을 시작하기 전에 다음 태스크를 수행하십시오.

1. 웹 서비스를 지원하도록 CICS 시스템을 구성합니다. 61 페이지의 『웹 서비스를 위한 CICS 시스템 구성』의 내용을 참조하십시오.
2. 웹 서비스의 배치를 지원하기 위해 필요한 인프라를 작성합니다. 61 페이지의 제 6 장 『웹 서비스 인프라 작성』의 내용을 참조하십시오.
3. 웹 서비스 지원을 사용할지 여부를 결정합니다. 53 페이지의 『SOAP 웹 서비스 사용 계획』의 내용을 참조하십시오.

이 태스크 정보

CICS 웹 서비스 지원은 사용자가 새 SOAP 웹 서비스 제공자 또는 서비스 요청자 애플리케이션에 필요한 아티팩트를 작성하고 기존 애플리케이션을 웹 서비스 제공자로 사용할 수 있도록 돕는 제공 유틸리티입니다.

CICS 웹 서비스 지원은 단순 언어 구조에서 WSDL 문서를 작성하거나 기존 WSDL 문서에서 언어 구조를 작성할 수 있으며 COBOL, C/C++ 및 PL/I를 지원합니다. 또한 SOAP 메시지를 컨테이너 및 COMMAREA로 또는 그 반대로 자동 런타임 변환을 가능하게 하기 위해 사용되는 정보를 생성합니다. 이 정보는 파이프라인 처리 중 CICS 웹 서비스 지원에서 사용됩니다.

다음 프로시저에서 설명하는 대로 웹 서비스를 작성하고 작성한 웹 서비스가 올바르게 작동하는지 유효성 검증하십시오.

프로시저

1. 다음 네 가지 방식 중 하나로 SOAP 웹 서비스를 작성하십시오.
 - 웹 서비스 지원을 사용하여 웹 서비스 설명 또는 언어 구조를 작성하고 작성한 항목을 CICS에 배치합니다. **PIPELINE SCAN** 명령을 사용하여 필수 CICS 자원을 자동 작성합니다.
 - z Systems용 Rational Developer 또는 Java API를 사용하여 웹 서비스 설명 또는 언어 구조를 작성하고 작성한 항목을 CICS에 배치합니다. **PIPELINE SCAN** 명령을 사용하여 필수 CICS 자원을 자동 작성합니다.

- 데이터 변환을 포함하여 인바운드 및 아웃바운드 메시지에서 XML을 처리하기 위해 애플리케이션 프로그램을 작성하거나 변경하고 파이프라인에 올바른 콘텐츠를 채웁니다. 필수 CICS 자원을 수동으로 작성해야 합니다.
 - Axis2 애플리케이션을 웹 서비스로 배치합니다.
2. 웹 서비스를 시작하여 웹 서비스가 의도한 대로 작동하는지 테스트하십시오. 웹 서비스 지원을 사용하여 웹 서비스를 배치하는 경우 **SET WEBSERVICE** 명령을 사용하여 유효성 검증을 결 수 있습니다. 이 유효성 검증에서는 데이터가 올바르게 변환되는지 확인합니다.

다음에 수행할 작업

이러한 단계는 다음 주제에서 보다 자세히 설명됩니다.

CICS 웹 서비스 지원자

CICS 웹 서비스 지원은 사용자가 기존 CICS 애플리케이션을 웹 서비스로 변환시키도록 도움을 줄 수 있고, CICS 애플리케이션이 외부 제공자가 제공하는 웹 서비스를 사용할 수 있도록 도움을 줄 수 있는 일련의 일괄처리 유틸리티입니다. 지원은 최소 프로그래밍 노력으로 서비스 제공자 및 서비스 요청자에서 사용하기 위한 CICS 애플리케이션의 빠른 배치를 지원합니다.

CICS에 웹 서비스 지원을 사용할 때, 인바운드 메시지를 구문 분석하고 아웃바운드 메시지를 구성하기 위한 사용자 자체 코드를 작성할 필요가 없습니다. CICS는 SOAP 메시지와 애플리케이션 프로그램의 데이터 구조 사이의 데이터를 맵핑합니다.

지원은 단순 언어 구조에서 WSDL 문서를 작성하거나 기존 WSDL 문서에서 언어 구조를 작성할 수 있고, COBOL, C/C++ 및 PL/I를 지원합니다. 이는 컨테이너 및 COMMAREA에 대한 SOAP 메시지의 자동화 런타임 변환을 가능하게 하거나 또는 그 반대에 사용되는 정보를 생성합니다.

CICS 웹 서비스 지원은 두 개의 유틸리티 프로그램으로 구성됩니다.

DFHLS2WS

언어 구조에서 웹 서비스 바인딩 파일을 생성합니다. 이 유틸리티는 웹 서비스 설명도 생성합니다.

DFHWS2LS

웹 서비스 설명에서 웹 서비스 바인딩 파일을 생성합니다. 이 유틸리티는 애플리케이션 프로그램에서 사용할 수 있는 언어 구조도 생성합니다.

양쪽 모두의 프로그램을 실행하기 위한 JCL 프로시저는 *hlq.XDFHINST* 라이브러리에 있습니다.

웹 서비스 지원의 유틸리티 프로그램 및 데이터 맵핑에 대한 자세한 정보는 다음 주제를 참조하십시오.

DFHLS2WS: 고급 언어 대 WSDL 변환

DFHLS2WS 프로시저는 고급 언어 데이터 구조에서 웹 서비스 바인딩 파일과 웹 서비스 설명을 생성합니다. DFHLS2WS는 CICS 애플리케이션 프로그램을 서비스 제공자로 표시할 때 사용할 수 있습니다.

DFHLS2WS와 기호 매개변수, 입력 매개변수와 설명, 작업 예제에 대한 작업 제어 명령문은 이 프로시저를 사용하는 데 도움을 줍니다.

DFHLS2WS의 작업 제어 명령문

JOB 작업을 시작합니다.

EXEC

프로시저 이름(DFHLS2WS)을 지정합니다.

INPUT.SYSUT1 DD

입력을 지정합니다. 입력 매개변수는 일반적으로 입력 스트림에 지정됩니다. 그러나 데이터 세트 또는 파티션된 데이터 세트의 멤버에 정의될 수 있습니다.

기호 매개변수

DFHLS2WS에서는 다음 기호 매개변수가 정의됩니다.

JAVADIR=*path*

DFHLS2WS에서 사용되는 Java 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/에 추가되어 /usr/lpp/*path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

PATHPREFIX=*prefix*

다른 매개변수에서 사용되는 z/OS UNIX 디렉토리 경로를 확장하는 선택적 접두부를 지정합니다. 기본값은 비어 있는 문자열입니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

SERVICE=*value*

이 매개변수는 IBM 지원 센터에서 지시하는 경우에만 사용합니다.

TMPDIR=*tmpdir*

DFHLS2WS가 z/OS UNIX에서 임시 작업공간으로 사용하는 디렉토리의 위치를 지정합니다. 작업이 실행되는 사용자 ID는 이 디렉토리에 대한 읽기 및 쓰기 권한이 있어야 합니다.

기본값은 /tmp입니다.

TMPFILE=*tmpprefix*

DFHLS2WS가 임시 작업공간 파일의 이름을 구성하기 위해 사용하는 접두부를 지정합니다.

기본값은 LS2WS입니다.

USSDIR=*path*

UNIX 시스템 서비스 파일 시스템에서 CICS TS 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/cicsts/에 추가되어 /usr/lpp/cicsts/*path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **USSDIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

임시 작업공간

DFHLS2WS는 런타임에 다음 세 개 임시 파일을 작성합니다.

tmpdir/tmpprefix.in

tmpdir/tmpprefix.out

tmpdir/tmpprefix.err

여기서,

*tmpdir*은 **TMPDIR** 매개변수에 지정된 값입니다.

*tmpprefix*는 **TMPFILE** 매개변수에 지정된 값입니다.

TMPDIR 및 **TMPFILE**이 지정되지 않은 경우 파일의 기본 이름은 다음과 같습니다.

/tmp/LS2WS.in

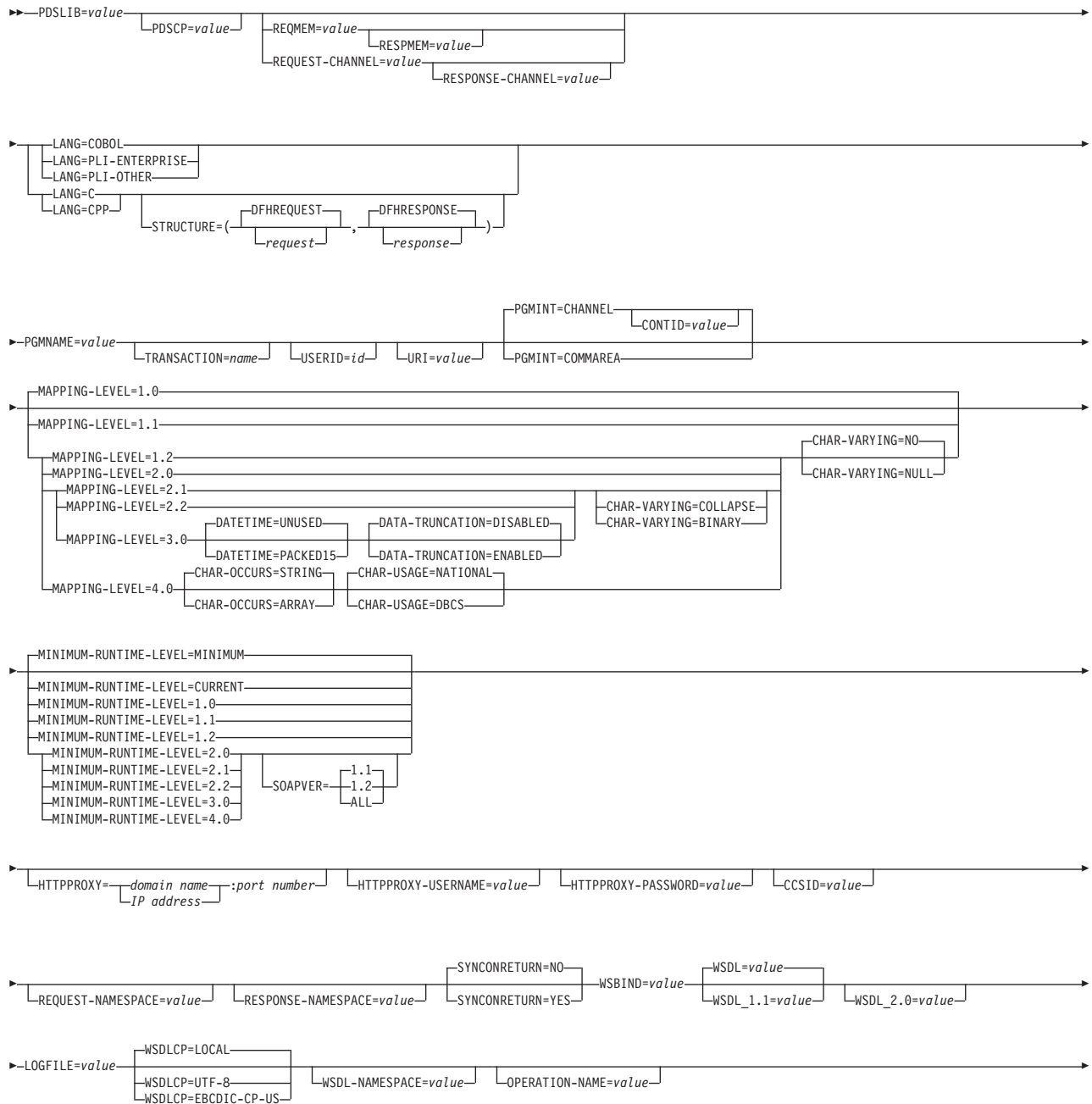
/tmp/LS2WS.out

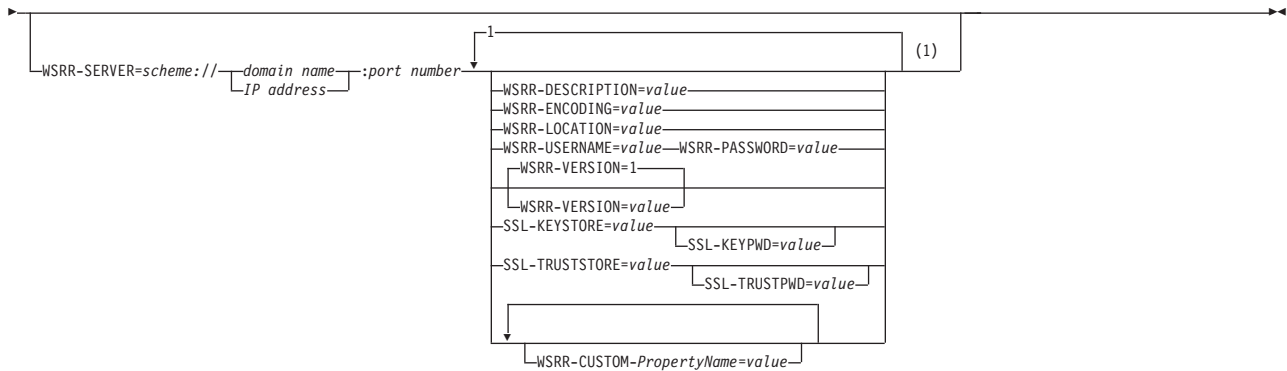
/tmp/LS2WS.err

중요사항: DFHLS2WS는 z/OS UNIX 파일 또는 데이터 세트 멤버에 대한 액세스를 차단하지 않습니다. 따라서 두 개 이상의 DFHLS2WS 인스턴스가 동시에 실행되고 동일한 임시 작업공간 파일을 사용하는 경우, 다른 작업이 작업공간 파일을 사용하는 동안 특정 작업이 해당 파일을 겹쳐쓸 수 있어 예상치 못한 실패를 경험할 수 있습니다.

따라서 이러한 상황을 방지하는 이름 지정 규칙과 작동 프로시저를 고안해야 합니다. 예를 들어, 시스템 기호 매개변수 **SYSUID**를 사용하여 개별 사용자에게 고유한 작업공간 파일 이름을 생성할 수 있습니다. 이러한 임시 파일은 작업이 종료되기 전에 삭제됩니다.

DFHLS2WS의 입력 매개변수





참고:

- 1 **WSRR-SERVER** 매개변수가 설정될 때 지정할 수 있는 각 **WSRR** 매개변수는 한 번만 지정할 수 있습니다. 이 규칙의 예외는 **WSRR-CUSTOM** 매개변수로 최대 255번까지 지정할 수 있습니다.

매개변수 사용

- 입력 매개변수는 순서에 관계 없이 지정할 수 있습니다.
- 각 매개변수는 행을 바꿔 시작해야 합니다.
- 매개변수는 사용 시 해당 연속 문자와 함께 열 72를 넘어가서는 안됩니다. 열 73 - 80은 공백이어야 합니다.
- 매개변수가 너무 길어 한 행에 맞지 않는 경우에는 행 끝에서 별표(*) 문자를 사용하여 매개변수가 다음 행에서 연속됨을 표시해야 합니다. 별표 앞의 모든 내용은 공백을 포함하여 매개변수의 일부로 간주됩니다. 예를 들어, 다음과 같습니다.

```
WSBIND=wsbinddir*  
/app1
```

구문은 다음과 같습니다.

```
WSBIND=wsbinddir/app1
```

- 행에서 첫 번째 문자 위치의 # 문자는 주석 문자입니다. 이 행은 무시됩니다.
- 행에서 마지막 문자 위치의 쉼표는 선택적 선 구분 기호이며 무시됩니다.

매개변수 설명

CCSID=value

런타임에 애플리케이션 데이터 구조에서 문자 데이터를 인코딩하는 데 사용되는 CCSID를 지정합니다. 이 매개변수의 값은 **LOCALCCSID** 시스템 초기화 매개변수의 값을 대체합니다. *value*는 Java 및 z/OS 변환 서비스가 지원하는 EBCDIC CCSID 여야 합니다(z/OS Unicode Services User's Guide and Reference 참조). 이 매개변수를 지정하지 않으면 시스템 초기화 매개변수에 지정된 CCSID를 사용하여 애플리케이션 데이터 구조가 인코딩됩니다.

이 매개변수는 임의 맵핑 레벨에 사용할 수 있습니다. 그러나 생성된 파일을 CICS TS 3.1 리전에 배치하려면 APAR PK23547을 적용하여 웹 서비스 바인딩 파일을 설치할 수 있는 코드의 최소 런타임 레벨을 달성해야 합니다.

CHAR-VARYING={NO|NULL|COLLAPSE|BINARY}

맵핑 레벨이 1.2 이상일 때 언어 구조의 문자 필드가 맵핑되는 방식을 지정합니다. COBOL의 문자 필드가 X 유형의 Picture 절인 경우(예: PIC(X) 10) C/C++의 문자 필드는 문자 배열입니다. 다음 옵션을 선택할 수 있습니다.

NO 문자 필드가 <xsd:string>에 맵핑되고 고정 길이 필드로 처리됩니다. 데이터의 최대 길이는 필드 길이와 같습니다. 맵핑 레벨이 2.0 이하인 COBOL 및 PL/I의 경우 **CHAR-VARYING** 매개변수의 기본값은 NO입니다.

이 값은 엔터프라이즈 및 기타 PL/I 언어 구조에는 적용되지 않습니다.

NULL

문자 필드가 <xsd:string>에 맵핑되고 널 종료 문자열로 처리됩니다. CICS는 SOAP 메시지에서 변환될 때 종료 널 문자를 추가합니다. 문자열의 최대 길이는 언어 구조에 표시된 길이보다 한 문자 적은 길이로 계산됩니다. C/C++의 경우 **CHAR-VARYING** 매개변수의 기본값은 NULL입니다.

이 값은 엔터프라이즈 및 기타 PL/I 언어 구조에는 적용되지 않습니다.

COLLAPSE

문자 필드가 <xsd:string>에 맵핑됩니다. 필드에서 후미 공백은 SOAP 메시지에 포함되지 않습니다. 인바운드 SOAP 메시지는 선두, 후미, 임베디드 공백을 모두 제거하기 위해 구문 분석됩니다. 맵핑 레벨이 2.1 이상인 COBOL 및 PL/I의 경우 **CHAR-VARYING** 매개변수의 기본값은 COLLAPSE입니다.

가변 길이 값 및 공백에 대한 자세한 정보는 가변 길이 값 및 공백에 대한 지원을 참조하십시오.

BINARY

문자 필드가 <xsd:base64binary>에 맵핑되고 고정 길이 필드로 처리됩니다. **CHAR-VARYING** 매개변수의 BINARY 값은 맵핑 레벨이 2.1 이상인 경우에만 사용할 수 있습니다.

CHAR-OCCURS={STRING|ARRAY}

맵핑 레벨이 4.0 이상일 때 언어 구조의 문자 배열이 맵핑되는 방식을 지정합니다 (예: PIC X OCCURS 20). 이 매개변수는 COBOL 언어만 사용할 수 있습니다.

ARRAY

문자 배열이 XML 배열에 맵핑됩니다. 이는 모든 문자가 개별 XML 요소에 맵핑됨을 의미합니다. 맵핑 레벨이 3.0 이하인 경우의 동작이기도 합니다.

STRING

문자 배열이 XML 문자열에 맵핑됩니다. 이는 전체 COBOL 배열이 단일 XML 요소에 맵핑됨을 의미합니다.

CHAR-USAGE={NATIONAL|DBCS}

COBOL에서 국가적 데이터 유형인 PIC N은 UTF-16 또는 DBCS 데이터에 사용될 수 있습니다. 이 설정은 NSYMBOL 컴파일러 옵션에 의해 제어됩니다. 데이터가 적절하게 처리되는지 확인하려면 보조의 **CHAR-USAGE** 매개변수를 NSYMBOL 컴파일러 옵션과 동일한 값으로 설정해야 합니다. 이는 주로 UTF-16을 사용할 때 CHAR-USAGE=NATIONAL로 설정됩니다.

DBCS PIC (*n*) 필드의 데이터는 UTF-16 인코딩 데이터로 처리됩니다.

NATIONAL

PIC (*n*) 필드의 데이터는 DBCS 인코딩 데이터로 처리됩니다.

CONTID=*value*

서비스 제공자에서, SOAP 메시지를 나타내는 데 사용되는 상위 레벨 데이터 구조를 보유하는 컨테이너의 이름을 지정합니다.

CICS가 대상 애플리케이션 프로그램으로 전달하는 컨테이너의 길이는 요청 컨테이너의 길이와 응답 컨테이너의 길이 중 긴 값입니다.

DATA-TRUNCATION={DISABLED|ENABLED}

가변 길이 데이터가 고정 길이 필드 구조에서 용인되는지 여부를 지정합니다.

DISABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 거부하고 오류 메시지를 발행합니다.

ENABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 용인하고 누락된 데이터를 널값으로 처리합니다.

DATETIME={UNUSED|PACKED15}

고급 언어 구조의 잠재적 ABSTIME 필드가 시간소인으로 맵핑되는지 여부를 지정합니다.

PACKED15

길이가 15(8바이트)인 팩형 10진수 필드가 CICS ABSTIME 필드로 처리되고 시간소인으로 맵핑됩니다.

UNUSED

길이가 15(8바이트)인 팩형 10진수 필드가 시간소인으로 처리되지 않습니다.

이 매개변수는 맵핑 레벨 3.0에서 설정할 수 있습니다.

HTTPPROXY={domain name:port number|IP address:port number}

WSDL에 인터넷에 있는 다른 WSDL 파일에 대한 참조를 포함하고 DFHLS2WS를 실행하는 시스템이 인터넷에 액세스하기 위해 프록시 서버를 사용하는 경우, 프록시 서버의 포트 번호와 도메인 이름 또는 IP 주소를 지정하십시오. 예를 들어, 다음과 같습니다.

HTTPPROXY=proxy.example.com:8080

다른 경우에는 이 매개변수가 필요하지 않습니다.

HTTPPROXY-PASSWORD=value

DFHLS2WS를 실행하는 시스템이 인터넷에 액세스하기 위해 HTTP 프록시 서버를 사용하고 HTTP 프록시 서버가 기본 인증을 사용하는 경우 **HTTPPROXY-USERNAME**과 함께 사용해야 하는 HTTP 프록시 비밀번호를 지정합니다. **HTTPPROXY**도 지정하는 경우에만 이 매개변수를 사용할 수 있습니다.

HTTPPROXY-USERNAME=value

DFHLS2WS를 실행하는 시스템이 인터넷에 액세스하기 위해 HTTP 프록시 서버를 사용하고 HTTP 프록시 서버가 기본 인증을 사용하는 경우 **HTTPPROXY-PASSWORD**와 함께 사용해야 하는 HTTP 프록시 사용자 이름을 지정합니다. **HTTPPROXY**도 지정하는 경우에만 이 매개변수를 사용할 수 있습니다.

LANG=COBOL

고급 언어 구조의 프로그래밍 언어가 COBOL인 것으로 지정합니다.

LANG=PLI-ENTERPRISE

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I인 것으로 지정합니다.

LANG=PLI-OTHER

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I 이외의 PL/I 레벨인 것으로 지정합니다.

LANG=C

고급 언어 구조의 프로그래밍 언어가 C인 것으로 지정합니다.

LANG=CPP

고급 언어 구조의 프로그래밍 언어가 C++인 것으로 지정합니다.

LOGFILE=value

DFHLS2WS가 해당 활동 로그 및 추적 정보를 기록하는 파일의 완전한 z/OS UNIX 이름입니다. DFHLS2WS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다.

이 파일은 일반적으로 사용하지 않지만 DFHLS2WS 관련 문제점이 발생하는 경우 IBM 서비스 조직이 요청할 수 있습니다.

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0}

웹 서비스 바인딩 파일과 웹 서비스 설명을 생성할 때 DFHLS2WS가 사용하는 맵핑 레벨을 지정합니다. 다음 옵션을 선택할 수 있습니다.

- 1.0** 이 맵핑 레벨은 기본값입니다. 웹 서비스 바인딩 파일이 CICS TS 3.1 맵핑 레벨을 사용하여 생성됨을 나타냅니다.
- 1.1** 이 특정 레벨에서 바인딩 파일을 다시 생성하려면 이 맵핑을 사용합니다.
- 1.2** 이 맵핑 레벨에서는 **CHAR-VARYING** 매개변수를 사용하여 런타임에 문자 배열이 처리되는 방식을 제어할 수 있습니다. **VARYING** 및 **VARYINGZ** 배열은 **PL/I**에서도 지원됩니다.
- 2.0** 이 맵핑 레벨은 CICS TS 3.2 리전 이상에서 언어 구조와 웹 서비스 바인딩 파일 간 맵핑에 대한 개선사항을 이용하기 위해 사용합니다.
- 2.1** APAR PK59794가 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2보다 상위 리전에는 이 맵핑 레벨을 사용합니다. 이 맵핑 레벨에서는 **CHAR-VARYING** 매개변수에 새 값, **COLLAPSE** 및 **BINARY**를 사용할 수 있습니다. **COBOL**의 **FILLER** 필드와 **PL/I**의 * 필드는 이 맵핑 레벨에서 시스템적으로 무시되며 생성된 **WSDL** 문서에 이 필드가 나타나지 않고 런타임에 데이터 구조에서 적절한 값이 유지됩니다.
- 2.2** APAR PK69738이 적용된 CICS TS 3.2 리전이나 CICS TS 3.2보다 상위 리전에 이 맵핑 레벨을 사용하면 **DFHWS2LS** 사용 시 맵핑 개선사항을 이용할 수 있습니다.
- 3.0** 이 맵핑 레벨은 CICS TS 4.1 리전에 사용합니다. 이 맵핑 레벨에서는 **REQUEST-CHANNEL** 및 **RESPONSE-CHANNEL** 매개변수를 설정하여 해당 인터페이스에서 많은 컨테이너를 사용하는 애플리케이션에서 웹 서비스를 작성할 수 있습니다. **DATETIME** 매개변수를 설정하여 **dateTime** 필드를 **XML** 시간소인에 맵핑할 수도 있습니다.
- 4.0** 이 맵핑 레벨은 CICS TS 5.2 리전에 사용합니다. 이 맵핑 레벨에서는 **COBOL OCCURS DEPENDING ON** 필드와 **CHAR-OCCURS** 매개변수를 사용할 수 있습니다.

맵핑 레벨에 대한 자세한 정보는 CICS 지원에 대한 맵핑 레벨을 참조하십시오.

MINIMUM - RUNTIME -

LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0|CURRENT}

웹 서비스 바인딩 파일이 배치될 수 있는 최소 CICS 런타임 환경을 지정합니다. 지정한 다른 매개변수와 일치하지 않는 레벨을 선택하면 오류 메시지를 받게 됩니다. 다음 옵션을 선택할 수 있습니다.

MINIMUM

지정된 매개변수의 경우 CICS의 런타임 레벨 중 가장 낮은 레벨이 자동으로 할당됩니다.

- 1.0** 생성된 웹 서비스 바인딩 파일이 APAR PK15904 및 PK23547이 적용되지 않은 CICS TS 3.1 리전에 배치됩니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 1.1** 생성된 웹 서비스 바인딩 파일이 최소 APAR PK15904가 적용된 CICS TS 3.1 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 1.1 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 1.2** 생성된 웹 서비스 바인딩 파일이 APAR PK15904 및 PK23547이 모두 적용된 CICS TS 3.1 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 1.2 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 3.2 이상 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 2.0 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.1** 생성된 웹 서비스 바인딩 파일이 APAR PK59794가 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2보다 상위 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 2.1 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.2** 생성된 웹 서비스 바인딩 파일이 APAR PK69738이 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2보다 상위 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 2.2 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 3.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 4.1 이상 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 3.0 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 4.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 5.2 이상 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 4.0 이하 맵핑 레벨을 사용할 수 있습니다. 이 레벨에서는 선택적 매개변수를 사용할 수 있습니다.

CURRENT

생성된 웹 서비스 바인딩 파일이 웹 서비스 바인딩 파일을 생성하기 위해 사용하는 런타임 레벨과 동일한 런타임 레벨에서 CICS 리전에 배치됩니다.

OPERATION-NAME=value

생성된 WSDL 문서에서 사용되는 조작 이름을 지정합니다. 값을 제공하지 않으면 **PGMNAME** 매개변수의 값과 그 뒤의 **operation** 값을 사용하여 기본 이름이 생성됩니다.

PDSLIB=value

처리할 고급 언어 데이터 구조를 포함하는 파티션된 데이터 세트의 이름을 지정합니다. 요청 및 응답에 사용되는 데이터 세트 멤버는 각각 **REQMEM** 및 **RESPMEM** 매개변수에 지정됩니다.

제한사항: 파티션된 데이터 세트 내 레코드는 80바이트의 고정 길이를 갖습니다.

PDSCP=value

REQMEM 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에서 사용되는 코드 페이지를 지정합니다. 여기서 *value*는 CCSID 번호 또는 Java 코드 페이지 번호입니다. 이 매개변수가 지정되지 않으면 z/OS UNIX System Services 코드 페이지를 사용합니다. 예를 들어, **PDSCP=037**을 지정할 수 있습니다.

PGMINT={CHANNEL|COMMAREA}

서비스 제공자의 경우, CICS가 대상 애플리케이션 프로그램으로 데이터를 전달하는 방식을 지정합니다.

CHANNEL

CICS가 채널 인터페이스를 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

- 맵핑 레벨이 3.0 이하인 경우에는 입출력에 모두 사용되는 하나의 컨테이너만 채널에 포함될 수 있습니다. 컨테이너의 이름을 지정하려면 **CONTID** 매개변수를 사용하십시오. 기본 이름은 DFHWS-DATA입니다.
- 맵핑 레벨 3.0에서는 채널에 여러 컨테이너가 포함될 수 있습니다. **REQUEST-CHANNEL** 및 **RESPONSE-CHANNEL** 매개변수를 사용하십시오. **PDSLIB**, **REQMEM** 또는 **RESPMEM**은 지정하지 마십시오.

COMMAREA

CICS가 통신 영역(COMMAREA)을 사용하여 대상 애플리케이션 프로그램으로 데이터를 전달합니다.

대상 애플리케이션 프로그램이 요청을 처리한 경우에는 동일한 메커니즘으로 응답을 리턴해야 합니다. 즉, 요청이 통신 영역에서 수신된 경우에는 응답도 통신 영역에서 리턴되어야 하고 요청이 컨테이너에서 수신된 경우에는 응답도 컨테이너에서 리턴되어야 합니다. CICS가 대상 애플리케이션 프로그램으로 전달하는 통신 영역 또는 컨테이너의 길이는 요청 통신 영역 또는 컨테이너의 길이와 응답 통신 영역 또는 컨테이너의 길이 중 긴 값입니다.

PGMNAME=value

웹 서비스로 노출될 대상 애플리케이션 프로그램에 대한 CICS PROGRAM 자원의 이름을 지정합니다. CICS 웹 서비스 지원은 이 프로그램에 링크됩니다.

REQMEM=value

웹 서비스 요청에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 지정합니다. 서비스 제공자의 경우 웹 서비스 요청은 애플리케이션 프로그램에 입력됩니다.

REQUEST-CHANNEL=value

채널 설명 문서의 이름과 위치를 지정합니다. 채널 설명은 웹 서비스 요청자로부터 SOAP 메시지를 수신할 때 웹 서비스 제공자 애플리케이션이 해당 인터페이스에서 사용할 수 있는 컨테이너를 설명합니다. 채널 설명은 CICS 제공 채널 스키마를 준수해야 하는 XML 문서입니다.

이 매개변수는 맵핑 레벨 3.0에서만 사용할 수 있습니다.

REQUEST-NAMESPACE=value

생성된 웹 서비스 설명에서 요청 메시지에 대한 XML 스키마의 네임스페이스를 지정합니다. 이 매개변수를 지정하지 않으면 CICS가 네임스페이스를 자동으로 생성합니다.

RESPMEM=value

웹 서비스 응답에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 지정합니다. 서비스 제공자의 경우 웹 서비스 응답은 애플리케이션 프로그램에서 출력됩니다.

응답과 관련이 없으면 즉, 단방향 메시지인 경우에는 이 매개변수를 생략하십시오.

RESPONSE-CHANNEL=value

채널 설명 문서의 이름과 위치를 지정합니다. 채널 설명은 웹 서비스 요청자로 SOAP 응답 메시지를 보낼 때 웹 서비스 제공자 애플리케이션이 해당 인터페이스에서 사용할 수 있는 컨테이너를 설명합니다. 채널 설명은 CICS 제공 채널 스키마를 준수해야 하는 XML 문서입니다.

이 매개변수는 맵핑 레벨 3.0에서만 사용할 수 있습니다.

RESPONSE-NAMESPACE=value

생성된 웹 서비스 설명에서 응답 메시지에 대한 XML 스키마의 네임스페이스를 지정합니다. 이 매개변수를 지정하지 않으면 CICS가 네임스페이스를 자동으로 생성합니다.

SOAPVER={1.1|1.2|ALL}

생성된 웹 서비스 설명에서 사용할 SOAP 레벨을 지정합니다. 이 매개변수는 **MINIMUM-RUNTIME-LEVEL**이 2.0 이상으로 설정된 경우에만 사용할 수 있습니다.

1.1 SOAP 1.1 프로토콜이 웹 서비스 설명에 대한 바인딩으로 사용됩니다.

1.2 SOAP 1.2 프로토콜이 웹 서비스 설명에 대한 바인딩으로 사용됩니다.

ALL SOAP 1.1 또는 1.2 프로토콜을 웹 서비스 설명에 대한 바인딩으로 사용할 수 있습니다.

이 매개변수의 값을 지정하지 않는 경우 기본값은 다음과 같이 작성하려는 WSDL 버전에 따라 다릅니다.

- WSDL 1.1만 필요한 경우에는 SOAP 1.1 바인딩이 사용됩니다.
- WSDL 2.0만 필요한 경우에는 SOAP 1.2 바인딩이 사용됩니다.
- WSDL 1.1과 WSDL 2.0이 모두 필요한 경우에는 각 웹 서비스 설명에 SOAP 1.1 및 1.2 바인딩이 모두 사용됩니다.

SSL-KEYSTORE=*value*

이 선택적 매개변수는 키 저장소 파일의 완전한 위치를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-KEYPWD=*value*

이 선택적 매개변수는 키 저장소의 비밀번호를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-TRUSTSTORE=*value*

이 선택적 매개변수는 신뢰 저장소 파일의 완전한 위치를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-TRUSTPWD=*value*

이 선택적 매개변수는 신뢰 저장소의 비밀번호를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

STRUCTURE=(*request,response*)

C 및 C++의 경우에만, **REQMEM** 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에 포함된 상위 레벨 구조의 이름을 지정합니다.

request

REQMEM 매개변수가 지정되는 경우 요청을 포함하는 상위 레벨 구조의 이름을 지정합니다. 기본값은 DFHREQUEST입니다.

파티션된 데이터 세트 멤버에는 사용자가 지정하는 이름을 갖는 상위 레벨 구조 또는 이름을 지정하지 않는 경우 DFHREQUEST 구조가 포함되어야 합니다.

response

RESPMEM 매개변수가 지정되는 경우 응답을 포함하는 상위 레벨 구조의 이름을 지정합니다. 기본값은 DFHRESPONSE입니다.

값을 지정하는 경우, 파티션된 데이터 세트 멤버에는 사용자가 지정하는 이름을 갖는 상위 레벨 구조 또는 DFHRESPONSE 구조(이름을 지정하지 않는 경우)가 포함되어야 합니다.

SYNCONRETURN={NO|YES}

원격 웹 서비스가 동기점을 발행할 수 있는지 여부를 지정합니다.

NO 원격 웹 서비스가 동기점을 발행할 수 없습니다. 이 값은 기본값입니다. 원격 웹 서비스가 동기점을 발행하면 ADPL 이상 종료와 함께 실패합니다.

YES 원격 웹 서비스가 동기점을 발행할 수 있습니다. YES를 선택하면 원격 웹 서비스에서 제어가 리턴되는 경우 원격 태스크가 개별 작업 단위로 커밋됩니다. 원격 웹 서비스가 복구 가능 자원을 갱신하고 리턴 후 실패하면 해당 자원에 대한 갱신을 백아웃할 수 없습니다.

TRANSACTION=name

서비스 제공자에서 이 매개변수는 파이프라인을 시작할 수 있는 별명 트랜잭션의 1 - 4자 이름을 지정합니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 TRANSACTION 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ # _ < >

URI=value

이 매개변수는 클라이언트가 웹 서비스에 액세스하기 위해 사용할 상대 또는 절대 URI를 지정합니다. CICS는 DFHLS2WS로 작성된 웹 서비스 바인딩 파일에서 URIMAP 자원을 생성할 때 지정되는 값을 사용합니다. 이 매개변수는 URIMAP 정의가 적용되는 URI의 경로 컴포넌트를 지정합니다.

USERID=id

서비스 제공자에서 이 매개변수는 웹 클라이언트가 사용할 수 있는 1 - 8자 사용자 ID를 지정합니다. 애플리케이션 생성 응답 또는 웹 서비스의 경우 이 사용자 ID 아래에 별명 트랜잭션이 첨부됩니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 USERID 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ #

WSBIND=*value*

웹 서비스 바인딩 파일의 완전한 z/OS UNIX 이름입니다. DFHLS2WS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 파일 확장자는 .wsbind입니다.

WSDL=*value*

웹 서비스 설명이 작성되는 파일의 완전한 z/OS UNIX 이름입니다. 웹 서비스 설명은 WSDL 1.1 스펙을 준수합니다. DFHLS2WS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 파일 확장자는 .wsdl입니다.

WSDL_1.1=*value*

웹 서비스 설명이 작성되는 파일의 완전한 z/OS UNIX 이름입니다. 웹 서비스 설명은 WSDL 1.1 스펙을 준수합니다. DFHLS2WS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 파일 확장자는 .wsdl입니다. 이 매개변수는 **WSDL** 매개변수와 동일한 결과를 생성하므로 둘 중 하나만 지정할 수 있습니다.

WSDL_2.0=*value*

웹 서비스 설명이 작성되는 파일의 완전한 z/OS UNIX 이름입니다. 웹 서비스 설명은 WSDL 2.0 스펙을 준수합니다. DFHLS2WS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 파일 확장자는 .wsdl입니다. 이 매개변수는 **WSDL** 또는 **WSDL_1.1** 매개변수와 함께 사용할 수 있습니다. 이 매개변수는 **MINIMUM-RUNTIME-LEVEL**이 2.0 이상으로 설정된 경우에만 사용할 수 있습니다.

WSDLCP={LOCAL|UTF-8|EBCDIC-CP-US}

WSDL 문서를 생성하는 데 사용되는 코드 페이지를 지정합니다.

LOCAL

WSDL 문서가 로컬 코드 페이지를 사용하여 생성되며 WSDL 문서에서는 인코딩 태그가 생성되지 않는 것으로 지정합니다.

UTF-8

WSDL 문서가 UTF-8 코드 페이지를 사용하여 생성되는 것으로 지정합니다. WSDL 문서에는 인코딩 태그가 생성됩니다. 이 옵션을 지정하는 경우 다른 플랫폼 간에 WSDL 문서를 복사할 때 인코딩이 올바르게 유지되는지 확인해야 합니다.

EBCDIC-CP-US

이 값은 WSDL 문서가 US EBCDIC 코드 페이지를 사용하여 생성되는 것으로 지정합니다. WSDL 문서에는 인코딩 태그가 생성됩니다.

WSDL-NAMESPACE=value

생성된 WSDL 문서에서 사용할 CICS의 네임스페이스를 지정합니다.

이 매개변수를 지정하지 않으면 CICS가 네임스페이스를 자동으로 생성합니다.

WSRR-CUSTOM-PropertyName=value

WSRR에서 WSDL 문서에 사용자 정의 메타데이터를 추가하려면 이 선택적 매개변수를 사용하십시오. **WSRR-CUSTOM-PropertyName=value** 쌍은 WSDL 문서에 추가되고 WSRR에 WSSR-CUSTOM 접두부 없이 나타납니다.

최대 255개 사용자 정의 **PropertyName=value** 쌍을 지정할 수 있습니다. **PropertyName=value** 쌍은 중복되거나 공백이어서는 안 됩니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-DESCRIPTION=value

공개되는 WSDL 문서를 설명하는 메타데이터를 지정하려면 이 선택적 매개변수를 사용하십시오.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-ENCODING=value

WSDL 문서의 문자 세트 인코딩을 지정하려면 이 선택적 매개변수를 사용하십시오. **WSRR-ENCODING** 매개변수가 지정되지 않으면 WSRR이 WSDL 문서에 지정된 값을 사용합니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-LOCATION=value

WSDL 문서의 위치를 식별하는 URI를 지정하려면 이 선택적 매개변수를 사용하십시오. 이 매개변수가 지정되지 않는 경우 URI의 기본값은 **WSDL** 매개변수에 지정된 파일 이름입니다. 예를 들어, **WSDL** 매개변수의 값이 `wsrr/example.wsdl`인 경우 **WSRR-LOCATION** 매개변수의 기본값은 `example.wsdl`입니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-PASSWORD=value

WSRR에 액세스하기 위해 비밀번호가 필요한 경우에는 이 선택적 매개변수를 사용하십시오.

WSRR-USERNAME 매개변수가 지정되는 경우 이 매개변수도 지정해야 합니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-SERVER={domain name:port number|IP address:port number}

WSRR(IBM WebSphere Service Registry and Repository) 서버의 위치를 지정하려면 이 매개변수를 사용하십시오. 이 매개변수가 지정되면 WSRR 매개변수 유효성 검증이 사용됩니다.

WSRR-USERNAME=value

WSRR에 액세스하기 위해 사용자 이름을 지정해야 하는 경우 이 선택적 매개변수를 사용하십시오. 이 사용자 이름은 WSRR이 소유자 특성을 설정하기 위해 사용합니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용합니다.

WSRR-VERSION={1|value}

WSRR에서 WSDL 문서의 버전 특성을 설정하려면 이 매개변수를 사용하십시오.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용합니다.

기타 정보

- DFHLS2SC를 실행하는 사용자 ID는 UNIX System Services를 사용하도록 구성되어야 합니다. 사용자 ID는 CICS z/OS UNIX 파일 구조 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND**, **WSDL** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다.
- JCL의 최대 매개변수 길이는 100자입니다. 이 길이는 **STDPRM**문을 사용하여 늘릴 수 있습니다. 자세한 정보는 *z/OS UNIX System Services User Guide*를 참조하십시오.

예제

```
//LS2WS JOB 'accounting information',name,MSGCLASS=A
// SET QT=''
//JAVAPROG EXEC DFHLS2WS,
// TMPFILE=&QT.&SYSUID.&QT
//INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
REQMEM=DFH0XCP4
RESPMEM=DFH0XCP4
LANG=COBOL
LOGFILE=/u/exampleapp/wsbinding/example.log
MINIMUM-RUNTIME-LEVEL=2.1
MAPPING-LEVEL=2.1
CHAR-VARYING=COLLAPSE
PGMNAME=DFH0XCMN
URI=http://myserver.example.org:8080/exampleApp/example
PGMINT=COMMAREA
SOAPVER=1.1
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbinding/example.wsbinding
WSDL=/u/exampleapp/wsd1/example.wsd1
WSDL_2.0=/u/exampleapp/wsd1/example_20.wsd1
WSDLCP=LOCAL
WSDL-NAMESPACE=http://mywsdlnamespace
/*
```


DFHWS2LS: WSDL을 고급 언어로 변환

DFHWS2LS 프로시저는 웹 서비스 설명으로부터 고급 언어 데이터 구조 및 웹 서비스 바인딩 파일을 생성합니다. CICS 애플리케이션 프로그램을 서비스 제공자로 노출하거나 서비스 요청자를 구성할 때 DFHWS2LS를 사용할 수 있습니다.

DFHWS2LS에 대한 작업 제어 명령문, 해당 기호 매개변수, 해당 입력 매개변수와 그에 대한 설명 및 예제 작업은 이 프로시저를 사용할 수 있도록 도움을 줍니다.

DFHWS2LS에 대한 작업 제어 명령문

JOB 작업을 시작합니다.

EXEC

프로시저 이름(DFHWS2LS)을 지정합니다.

INPUT.SYSUT1 DD

입력을 지정합니다. 입력 매개변수는 일반적으로 입력 스트림에 지정됩니다. 그러나 데이터 세트 또는 파티션된 데이터 세트의 멤버에 정의될 수 있습니다.

기호 매개변수

다음 기호 매개변수가 DFHWS2LS에서 정의됩니다.

JAVADIR=*path*

DFHWS2LS에서 사용되는 Java 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/에 추가되어 /usr/lpp/*path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

PATHPREFIX=*prefix*

다른 매개변수에서 사용되는 z/OS UNIX 디렉토리 경로를 확장하는 선택적 접두부를 지정합니다. 기본값은 비어 있는 문자열입니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

TMPDIR=*tmpdir*

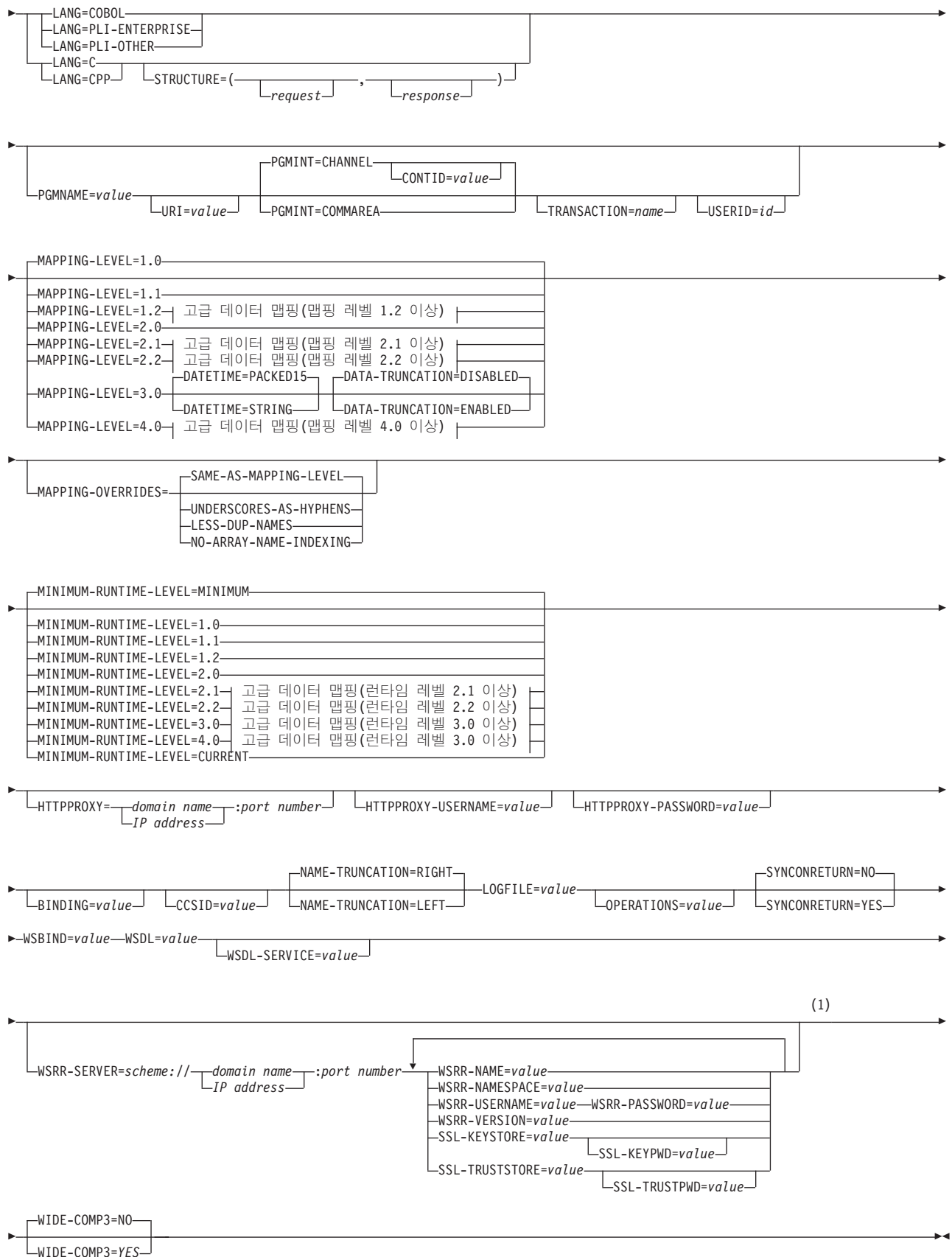
DFHWS2LS가 임시 작업공간으로 사용하는 z/OS UNIX의 디렉토리 위치를 지정합니다. 작업이 실행되는 사용자 ID는 이 디렉토리에 대한 읽기 및 쓰기 권한이 있어야 합니다.

기본값은 /tmp입니다.

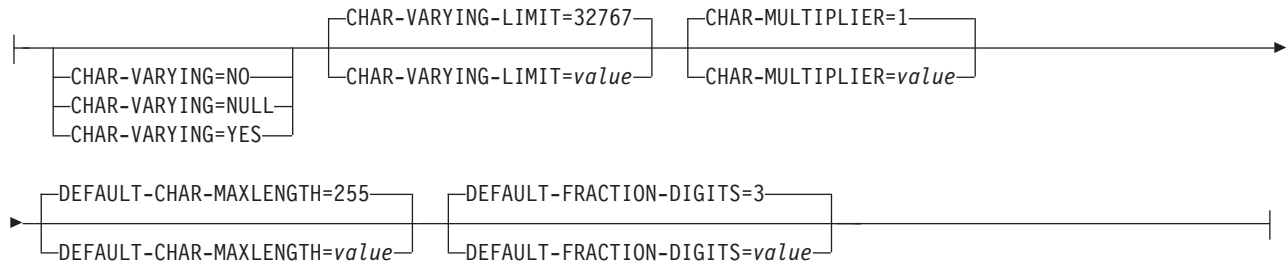
TMPPFILE=*tmpprefix*

DFHWS2LS가 임시 작업공간 파일의 이름을 구성하는 데 사용하는 접두부를 지정합니다.

기본값은 WS2LS입니다.



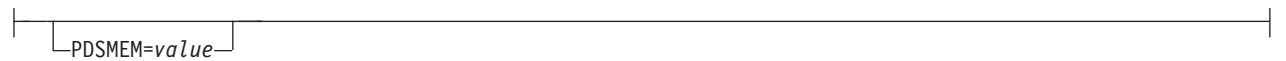
고급 데이터 맵핑(맵핑 레벨 1.2 이상):



고급 데이터 맵핑(맵핑 레벨 2.1 이상):



고급 데이터 맵핑(맵핑 레벨 2.2 이상):



고급 데이터 맵핑(런타임 레벨 2.1 이상):



고급 데이터 맵핑(런타임 레벨 3.0 이상):



참고:

- 1 **WSRR-SERVER** 매개변수가 설정될 때 지정할 수 있는 각 **WSRR** 매개변수는 한 번만 지정할 수 있습니다.

매개변수 사용

- 입력 매개변수는 순서에 관계 없이 지정할 수 있습니다.
- 각 매개변수는 행을 바꿔 시작해야 합니다.
- 매개변수는 사용 시 해당 연속 문자와 함께 열 72를 넘어가서는 안됩니다. 열 73 - 80은 공백이어야 합니다.

- 매개변수가 너무 길어 한 행에 맞지 않는 경우에는 행 끝에서 별표(*) 문자를 사용하여 매개변수가 다음 행에서 연속됨을 표시해야 합니다. 별표 앞의 모든 내용은 공백을 포함하여 매개변수의 일부로 간주됩니다. 예를 들어, 다음과 같습니다.

```
WSBIND=wsbinddir*  
/app1
```

구문은 다음과 같습니다.

```
WSBIND=wsbinddir/app1
```

- 행에서 첫 번째 문자 위치의 # 문자는 주석 문자입니다. 이 행은 무시됩니다.
- 행에서 마지막 문자 위치의 쉼표는 선택적 선 구분 기호이며 무시됩니다.

매개변수 설명

BINDING=*value*

웹 서비스 설명에 한 개 이상의 <wsdl:바인딩> 요소가 포함된 경우, 이 매개변수를 사용하여 언어 구조 및 웹 서비스 바인딩 파일을 생성하는 데 사용할 요소를 지정하십시오. 웹 서비스 설명의 <wsdl:바인딩> 요소에 대해 사용되는 name 속성의 값을 지정하십시오.

CCSID=*value*

런타임에 애플리케이션 데이터 구조에서 문자 데이터를 인코딩하는 데 사용되는 CCSID를 지정합니다. 이 매개변수의 값은 **LOCALCCSID** 시스템 초기화 매개변수의 값을 대체합니다. *value*는 Java 및 z/OS 변환 서비스가 지원하는 EBCDIC CCSID 여야 합니다(z/OS Unicode Services User's Guide and Reference 참조). 이 매개변수를 지정하지 않으면 시스템 초기화 매개변수에 지정된 CCSID를 사용하여 애플리케이션 데이터 구조가 인코딩됩니다.

이 매개변수는 임의 맵핑 레벨에 사용할 수 있습니다. 그러나 생성된 파일을 CICS TS 3.1 리전에 배치하려면 APAR PK23547을 적용하여 웹 서비스 바인딩 파일을 설치할 수 있는 코드의 최소 런타임 레벨을 달성해야 합니다.

CHAR-MULTIPLIER={1**|***value***}**

맵핑 레벨이 1.2 이상인 경우 각 문자에 허용되는 바이트 수를 지정합니다. 이 매개변수의 *value*는 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다. 모든 비숫자 문자 기반 맵핑은 이 승수가 적용됩니다. 2진, 숫자, 존형 및 팩형 10진수 필드는 이 승수의 대상이 아닙니다.

이 매개변수는 예를 들어, DBCS 문자를 사용하려는 경우에 유용할 수 있습니다. 이 경우 런타임 시 모든 2바이트 문자 앞뒤로 잠재적인 SO 및 SI 문자를 위한 공간을 허용하기 위해 승수 3을 선택할 수 있습니다.

CCSID=1200(UTF-16 표시)을 설정하면 **CHAR-MULTIPLIER**에 대한 유일한 올바른 값은 2 또는 4입니다. UTF-16을 사용할 때 기본값은 2입니다. 애플리케이션 데이터가 1개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우

CHAR-MULTIPLIER=2를 사용하십시오. 애플리케이션 데이터가 2개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우 **CHAR-MULTIPLIER=4**를 사용하십시오.

참고: **CHAR-MULTIPLIER**를 1로 설정하면 DBCS 문자의 사용이 배제되지 않으며 이를 2로 설정하면 UTF-16 대리 쌍의 사용을 배제하지 않습니다. 그러나 와이드 문자가 주기적으로 사용되면 일부 올바른 값이 할당된 필드에 맞지 않게 됩니다. **CHAR-MULTIPLIER**의 값으로 더 큰 값을 사용할 수록 XML에서 올바른 문자 수보다 더 많은 문자를 할당된 필드에 저장할 수 있습니다. 해당 범위 제한을 주의하여 따라야 합니다.

CHAR-VARYING={NO|NULL|YES}

맵핑 레벨이 1.2 이상인 경우 가변 길이 문자 데이터가 맵핑되는 방식을 지정합니다. 가변 길이 2진 데이터 유형은 항상 컨테이너 또는 가변 구조에 맵핑됩니다. 이 매개변수를 지정하지 않은 경우 지정한 언어에 따라 기본 맵핑이 달라집니다. 다음 옵션을 선택할 수 있습니다.

아니오 가변 길이 문자 데이터가 고정 길이 문자열로 맵핑됩니다.

널 가변 길이 문자 데이터가 널 종료 문자열로 맵핑됩니다.

예 PL/I의 경우 가변 길이 문자 데이터가 CHAR VARYING 데이터 유형에 맵핑됩니다. COBOL, C, C++ 언어에서는 가변 길이 문자 데이터가 두 가지 관련 요소(데이터-길이 및 데이터)로 구성되는 해당 표시에 맵핑됩니다.

CHAR-VARYING-LIMIT={32767|value}

맵핑 레벨이 1.2 이상인 경우 언어 구조에 맵핑되는 가변 길이 문자 데이터와 2진 데이터의 최대 크기를 지정합니다. 문자 또는 2진 데이터가 이 매개변수에 지정된 값보다 큰 경우에는 컨테이너에 맵핑되고 생성된 언어 구조에서 컨테이너 이름이 사용됩니다. 값의 범위는 0 - 32,767바이트(기본값)입니다.

CONTID=value

서비스 제공자에서, SOAP 메시지를 나타내는 데 사용되는 상위 레벨 데이터 구조를 보유하는 컨테이너의 이름을 지정합니다.

CICS가 대상 애플리케이션 프로그램으로 전달하는 컨테이너의 길이는 요청 컨테이너의 길이와 응답 컨테이너의 길이 중 긴 값입니다.

DATA-TRUNCATION={DISABLED|ENABLED}

가변 길이 데이터가 고정 길이 필드 구조에서 용인되는지 여부를 지정합니다.

DISABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 거부하고 오류 메시지를 발행합니다.

ENABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 용인하고 누락된 데이터를 널값으로 처리합니다.

DATETIME={PACKED15|STRING}

<xsd:dateTime> 요소가 언어 구조로 매핑되는 방식을 지정합니다.

PACKED15

기본값은 모든 <xsd:dateTime> 요소가 시간소인으로 처리되고 CICS ABSTIME 형식으로 매핑되는 것입니다.

STRING

<xsd:dateTime> 요소가 텍스트로 처리됩니다.

DEFAULT-CHAR-MAXLENGTH={255|value}

매핑 레벨이 1.2 이상이고 웹 서비스 설명 문서에 길이가 나타나지 않는 경우 매핑 문자 내 문자 데이터의 기본 배열 길이를 지정합니다. 이 매개변수의 값은 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다.

DEFAULT-FRACTION-DIGITS={3|value}

XML 10진수 스키마 유형에 대해 사용할 기본 소수 자리수를 지정합니다. 기본값은 3입니다. COBOL의 경우 유효한 값은 0 - 17이고 매개변수 **WIDE-COMP3**를 사용 중인 경우 0 - 30입니다. C 또는 PLI의 경우 유효 범위는 0 - 30입니다.

HTTPPROXY={domain name:port number|IP address:port number}

인터넷에 있는 다른 WSDL 파일에 대한 참조가 WSDL에 포함되고 DFHWS2LS를 실행하고 있는 시스템이 프록시 서버를 사용하여 인터넷에 액세스하는 경우 도메인 이름 또는 IP 주소 및 프록시 서버의 포트 번호를 지정합니다. 예를 들어, 다음과 같습니다.

HTTPPROXY=proxy.example.com:8080

다른 경우에는 이 매개변수가 필요하지 않습니다.

HTTPPROXY-PASSWORD=value

DFHWS2LS를 실행하고 있는 시스템이 HTTP 프록시 서버를 사용하여 인터넷에 액세스하고 이 HTTP 프록시 서버가 기본 인증을 사용하는 경우에는 **HTTPPROXY-USERNAME**에 사용해야 하는 HTTP 프록시 비밀번호를 지정합니다. **HTTPPROXY**도 지정하는 경우에만 이 매개변수를 사용할 수 있습니다.

HTTPPROXY-USERNAME=value

DFHWS2LS를 실행하고 있는 시스템이 HTTP 프록시 서버를 사용하여 인터넷에 액세스하고 이 HTTP 프록시 서버가 기본 인증을 사용하는 경우에는 **HTTPPROXY-PASSWORD**에 사용해야 하는 HTTP 프록시 사용자 이름을 지정합니다. **HTTPPROXY**도 지정하는 경우에만 이 매개변수를 사용할 수 있습니다.

INLINE-MAXOCCURS-LIMIT={1|value}

maxOccurs 속성에 따라 인라인 가변 반복 콘텐츠가 사용되는지 여부를 지정합니다. 인라인으로 맵핑되는 가변 반복 콘텐츠는 생성된 언어 구조의 나머지와 함께 현재 컨테이너에 배치됩니다. 가변 반복 콘텐츠는 두 파트로, 즉 데이터 발생 횟수를 저장하는 카운터와 데이터의 각 발생을 저장하는 배열로 저장됩니다. 가변 반복 콘텐츠의 대체 맵핑은 컨테이너 기반 맵핑으로, 데이터 발생 횟수와 데이터가 배치되는 컨테이너의 이름을 저장합니다. 개별 컨테이너에 데이터를 저장하는 경우 성능에 영향을 주어 인라인 맵핑이 선호될 수 있습니다.

INLINE-MAXOCCURS-LIMIT 매개변수는 맵핑 레벨 2.1 이상에서만 사용할 수 있습니다. **INLINE-MAXOCCURS-LIMIT** 값은 0 - 32,767 범위의 양의 정수일 수 있습니다. 값이 0이면 인라인 맵핑이 사용되지 않음을 나타냅니다. 값이 1이면 선택적 요소가 인라인으로 맵핑됩니다. maxOccurs 속성의 *value*가

INLINE-MAXOCCURS-LIMIT의 *value*보다 크면 컨테이너 기반 맵핑을 사용하고 그렇지 않으면 인라인 맵핑을 사용합니다.

가변적으로 반복되는 목록이 인라인 맵핑되도록 할지 선택할 때 반복 데이터의 단일 항목 길이를 고려하십시오. 길이가 긴 인스턴스가 거의 없는 경우에는 컨테이너 기반 맵핑이 선호됩니다. 길이가 짧은 인스턴스가 많은 경우에는 인라인 맵핑이 선호됩니다.

LANG=COBOL

고급 언어 구조의 프로그래밍 언어가 COBOL인 것으로 지정합니다.

LANG=PLI-ENTERPRISE

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I인 것으로 지정합니다.

LANG=PLI-OTHER

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I 이외의 PL/I 레벨인 것으로 지정합니다.

LANG=C

고급 언어 구조의 프로그래밍 언어가 C인 것으로 지정합니다.

LANG=CPP

고급 언어 구조의 프로그래밍 언어가 C++인 것으로 지정합니다.

LOGFILE=value

DFHWS2LS가 그 활동 로그 및 추적 정보를 쓰는 파일의 완전한 z/OS UNIX 이름입니다. DFHWS2LS는 파일을 작성하지만 이미 존재하지 않는 경우 디렉토리 구조는 작성하지 않습니다.

일반적으로 이 파일을 사용하지는 않지만 DFHWS2LS와 관련된 문제점이 발생하는 경우 IBM 서비스 조직에서 이 파일을 요청할 수 있습니다.

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0}

웹 서비스 바인딩 파일 및 언어 구조를 생성할 때 DFHWS2LS가 사용하는 맵핑 레벨을 지정합니다. 다음 옵션을 선택할 수 있습니다.

- 1.0** CICS TS 3.1 맵핑 레벨을 사용하여 웹 서비스 바인딩 파일 및 언어 구조가 생성됩니다.
- 1.1** XML 속성 및 <목록>과 <유니온> 데이터 유형이 언어 구조로 맵핑됩니다. 최대 길이가 32,767바이트를 초과하는 문자 및 2진 데이터는 컨테이너로 맵핑됩니다. 컨테이너 이름이 언어 구조에서 작성됩니다.
- 1.2** **CHAR-VARYING** 및 **CHAR-VARYING-LIMIT** 매개변수를 사용하여 런타임 시 문자 데이터가 맵핑되고 처리되는 방식을 제어합니다. 이들 매개변수 중 어느 매개변수도 지정하지 않으면 최대 길이가 32,768바이트 미만인 2진 및 문자 데이터는 C++을 제외한 모든 언어에 대해 **VARYING** 구조로 맵핑됩니다. C++에서는 문자 데이터가 널 종료 문자열로 맵핑됩니다.
- 2.0** 이 맵핑 레벨은 CICS TS 3.2 리전 이상에서 언어 구조와 웹 서비스 바인딩 파일 간 맵핑에 대한 개선사항을 이용하기 위해 사용합니다.
- 2.1** 이 맵핑 레벨은 <xsd:any> 및 xsd:anyType 지원, 가변적으로 반복되는 컨테츠를 **INLINE-MAXOCCURS-LIMIT** 매개변수로 인라인 맵핑하는 옵션 및 <xsd:sequence>, <xsd:choice> 및 <xsd:all>에서 minOccurs="0"에 대한 지원을 얻기 위해 APAR PK59794가 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2 이후의 모든 리전에서 사용합니다.
- 2.2** 이 맵핑 레벨은 APAR PK69738이 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2 이후의 모든 리전에서 사용합니다. 이 맵핑 레벨에서는 다음 지원을 제공합니다.
 - 고정값을 가지는 요소
 - <xsd:choice> 요소에 대한 개선된 지원
 - 추상 데이터 유형
 - 추상 요소
 - 대체 그룹
- 3.0** 이 맵핑 레벨은 CICS TS 4.1 또는 이후 리전에서 사용합니다. 이 맵핑 레벨에서 시간소인을 CICS ABSTIME 형식으로 변환할 수 있습니다.
- 4.0** UTF-16을 사용하려는 경우 CICS TS 5.2 리전에 이 맵핑 레벨을 사용하십시오.

맵핑 레벨에 대한 자세한 정보는 CICS 지원을 위한 맵핑 레벨의 내용을 참조하십시오.

MAPPING-OVERRIDES={SAME-AS-MAPPING-LEVEL|UNDERSCORES-AS-HYPHENS|LESS-DUP-NAMES|NO-ARRAY-NAME-INDEXING}

언어 구조를 생성할 때 지정된 맵핑 레벨에 대해 기본 동작이 대체되는지 여부를 지정합니다.

참고: 모든 하위 옵션은 쉼표로 구분된 목록으로 사용할 수 있습니다. 이러한 옵션은 상호 배타적이지 않고 조합되며 순서가 지정되지 않습니다.

LESS-DUP-NAMES

이 매개변수는 이름 끝에 `_value`가 있는 비구조적 구조 필드 이름을 생성하여 필드를 직접 참조할 수 있습니다. 예를 들어, 다음 PLI 언어 구조에서 **MAPPING-OVERRIDES=LESS-DUP-NAMES**가 지정되는 경우 레벨 12 필드 `streetName`에는 접미부 `_value`가 추가됩니다.

```
09 streetName,
   12 streetName                                CHAR(255) VARYING
   UNALIGNED,
   12 filler                                    BIT (7),
   12 attr_nil_streetName_value                BIT (1),
```

결과적으로 구조는 다음과 같습니다.

```
09 streetName,
   12 streetName_value                          CHAR(255) VARYING
   UNALIGNED,
   12 filler                                    BIT (7),
   12 attr_nil_streetName_value                BIT (1),
```

SAME-AS-MAPPING-LEVEL

이 매개변수는 맵핑 레벨과 동일한 스타일의 언어 구조를 생성합니다. 이는 기본값입니다.

UNDERSCORES-AS-HYPHENS

COBOL에서만 사용됩니다. 이 매개변수는 WSDL 문서의 밑줄을 문자 X가 아닌 하이픈으로 변환하여 생성된 COBOL 언어 구조의 가독성을 향상시킵니다. 필드 이름 충돌이 발생하면 필드가 고유할 수 있도록 필드에 번호가 지정됩니다. 자세한 정보는 애플리케이션 개발에서 XML 스키마 대 COBOL 맵핑의 내용을 참조하십시오.

NO-ARRAY-NAME-INDEXING

엔터프라이즈 PL/I에서만 사용됩니다. 배열 내의 필드 이름은 상위 레벨 구조의 범위 내에서만 고유합니다.

MINIMUM - RUNTIME -

LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0|CURRENT}

웹 서비스 바인딩 파일이 배치될 수 있는 최소 CICS 런타임 환경을 지정합니다. 지정한 다른 매개변수와 일치하지 않는 레벨을 선택하면 오류 메시지를 받게 됩니다. 다음 옵션을 선택할 수 있습니다.

MINIMUM

지정된 매개변수의 경우 CICS의 런타임 레벨 중 가장 낮은 레벨이 자동으로 할당됩니다.

- 1.0** 생성된 웹 서비스 바인딩 파일이 APAR PK15904 및 PK23547이 적용되지 않은 CICS TS 3.1 리전에 배치됩니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 1.1** 생성된 웹 서비스 바인딩 파일이 최소 APAR PK15904가 적용된 CICS TS 3.1 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 1.1 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 1.2** 생성된 웹 서비스 바인딩 파일이 APAR PK15904 및 PK23547이 모두 적용된 CICS TS 3.1 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 1.2 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 3.2 이상 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 2.0 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.1** 생성된 웹 서비스 바인딩 파일이 APAR PK59794가 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2보다 상위 리전에 배치됩니다. MAPPING-LEVEL 매개변수에는 2.1 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 2.2** 생성된 웹 서비스 바인딩 파일이 APAR PK69738이 적용된 CICS TS 3.2 리전 또는 CICS TS 3.2보다 상위 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 2.2 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 3.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 4.1 이상 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 3.0 이하 맵핑 레벨을 사용할 수 있습니다. 일부 매개변수는 이 런타임 레벨에서 사용할 수 없습니다.
- 4.0** 생성된 웹 서비스 바인딩 파일이 CICS TS 5.2 이상 리전에 배치됩니다. 이 런타임 레벨에서는 MAPPING-LEVEL 매개변수에 4.0 이하 맵핑 레벨을 사용할 수 있습니다. 이 레벨에서는 선택적 매개변수를 사용할 수 있습니다.

CURRENT

생성된 웹 서비스 바인딩 파일이 웹 서비스 바인딩 파일을 생성하기 위해 사용하는 런타임 레벨과 동일한 런타임 레벨에서 CICS 리전에 배치됩니다.

NAME-TRUNCATION={LEFT|RIGHT}

XML 요소의 이름을 왼쪽부터 자를지 또는 오른쪽부터 자를지 지정합니다. CICS 웹 서비스 지원은 XML 요소 이름을 지정된 고급 언어에 맞는 길이로 자르며 기본적으로 오른쪽부터 자릅니다.

OPERATIONS=value

웹 서비스 요청자 애플리케이션의 경우 웹 서비스 바인딩 파일을 생성하는 데 사용되는 웹 서비스 설명에서 유효한 <wsdl:Operation> 요소의 서브세트를 지정합니다. 각 <wsdl:Operation> 요소는 간격으로 분리되며 필요한 경우 목록이 둘 이상의 행에 걸쳐 이어질 수 있습니다. WSDL 1.1 및 WSDL 2.0 문서 둘 다에서 이 매개변수를 사용할 수 있습니다.

PDSCP=value

REQMEM 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에서 사용되는 코드 페이지를 지정합니다. 여기서 *value*는 CCSID 번호 또는 Java 코드 페이지 번호입니다. 이 매개변수가 지정되지 않으면 z/OS UNIX System Services 코드 페이지를 사용합니다. 예를 들어, **PDSCP=037**을 지정할 수 있습니다.

PDSLIB=value

생성된 고급 언어를 포함하는 파티션된 데이터 세트의 이름을 지정합니다. 요청 및 응답에 사용되는 데이터 세트 멤버는 각각 **REQMEM** 및 **RESPMEM** 매개변수에 지정됩니다.

PDSMEM=value

추상 데이터 유형에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 DFHWS2LS가 사용하는 1-에서 6-문자 접두부를 지정합니다. 접두부에 2자리 숫자를 추가하여 멤버 이름을 생성합니다.

추상 데이터 유형과 연관된 언어 구조의 이름을 지정하려면 맵핑 레벨 2.2 이상에서 이 매개변수를 사용하십시오. **PDSMEM** 매개변수가 생략된 경우 추상 데이터 유형에 대한 언어 구조는 **REQMEM** 매개변수의 값을 사용하여 이름 지정됩니다.

PGMINT={CHANNEL|COMMAREA}

서비스 제공자의 경우, CICS가 대상 애플리케이션 프로그램으로 데이터를 전달하는 방식을 지정합니다.

CHANNEL

CICS가 채널 인터페이스를 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

COMMAREA

CICS가 통신 영역을 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

이 매개변수는 DFHWS2LS의 출력이 서비스 요청자에서 사용될 때 무시됩니다.

대상 애플리케이션 프로그램이 요청을 처리한 경우에는 동일한 메커니즘으로 응답을 리턴해야 합니다. 통신 영역에서 요청을 수신한 경우 통신 영역에서 응답이 리턴되어야 합니다. 컨테이너에서 요청을 수신한 경우에는 컨테이너에서 응답이 리턴되어야 합니다. CICS가 대상 애플리케이션 프로그램으로 전달하는 통신 영역 또는 컨테이너의 길이는 요청 통신 영역 또는 컨테이너의 길이와 응답 통신 영역 또는 컨테이너의 길이 중 긴 값입니다.

PGMNAME=value

CICS PROGRAM 자원의 이름을 지정합니다.

DFHWS2LS가 서비스 제공자에서 사용될 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우 이 매개변수를 제공해야 합니다. 이 매개변수는 웹 서비스로 노출되는 애플리케이션 프로그램의 자원 이름을 지정합니다.

DFHWS2LS가 서비스 요청자에서 사용될 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우 이 매개변수를 생략하십시오.

REQMEM=value

웹 서비스 요청에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 DFHWS2LS가 사용하는 1-에서 6-문자 접두부를 지정합니다.

- 서비스 제공자의 경우 웹 서비스 요청은 애플리케이션 프로그램에 입력됩니다.
- 서비스 요청자에서 웹 서비스 요청은 애플리케이션 프로그램으로부터의 출력입니다.

DFHWS2LS는 각 조작에 대해 파티션된 데이터 세트 멤버를 생성합니다. 접두부에 2자리 숫자를 추가하여 멤버 이름을 생성합니다.

이 매개변수는 선택사항이기는 하지만 웹 서비스 설명에 요청의 정의가 포함되어 있는 경우에는 이 매개변수를 지정해야 합니다.

RESPMEM=value

웹 서비스 응답에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 DFHWS2LS가 사용하는 1-에서 6-문자 접두부를 지정합니다.

- 서비스 제공자의 경우 웹 서비스 응답은 애플리케이션 프로그램에서 출력됩니다.
- 서비스 요청자에서 웹 서비스 응답은 애플리케이션 프로그램에 대한 입력입니다.

DFHWS2LS는 각 조작에 대해 파티션된 데이터 세트 멤버를 생성합니다. 접두부에 2자리 숫자를 추가하여 멤버 이름을 생성합니다.

응답과 관련이 없으면 즉, 단방향 메시지인 경우에는 이 매개변수를 생략하십시오.

SSL-KEYSTORE=value

이 선택적 매개변수는 키 저장소 파일의 완전한 위치를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-KEYPWD=*value*

이 선택적 매개변수는 키 저장소의 비밀번호를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-TRUSTSTORE=*value*

이 선택적 매개변수는 신뢰 저장소 파일의 완전한 위치를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

SSL-TRUSTPWD=*value*

이 선택적 매개변수는 신뢰 저장소의 비밀번호를 지정합니다.

웹 서비스 지원이 네트워크를 통해 WSRR(IBM WebSphere Service Registry and Repository)과 통신하기 위해 보안 소켓 레이어(SSL) 암호화를 사용하도록 하려면 이 매개변수를 사용하십시오.

STRUCTURE=(*request,response*)

C 및 C++의 경우에만, 요청 및 응답 구조의 이름이 생성되는 방식을 지정합니다.

생성된 요청 및 응답 구조는 *requestnn* 및 *responsenn* 형식의 이름이며 여기에서 *nn*은 각 조작에 대해 구조를 구별하기 위해 생성되는 숫자 접미부입니다.

하나 또는 두 이름이 모두 생략되면 구조는 사용자가 지정한 **REQMEM** 및 **RESPMEM** 매개변수에서 생성된 파티션된 데이터 세트 멤버 이름과 같은 이름을 가집니다.

SYNCONRETURN={NO|YES}

원격 웹 서비스가 동기점을 발행할 수 있는지 여부를 지정합니다.

NO 원격 웹 서비스가 동기점을 발행할 수 없습니다. 이 값은 기본값입니다. 원격 웹 서비스가 동기점을 발행하면 ADPL 이상 종료와 함께 실패합니다.

YES 원격 웹 서비스가 동기점을 발행할 수 있습니다. YES를 선택하면 원격 웹 서비스에서 제어가 리턴되는 경우 원격 태스크가 개별 작업 단위로 커밋됩니다. 원격 웹 서비스가 복구 가능 자원을 갱신하고 리턴 후 실패하면 해당 자원에 대한 갱신을 백아웃할 수 없습니다.

TRANSACTION=*name*

서비스 제공자에서 이 매개변수는 파이프라인을 시작할 수 있는 별명 트랜잭션의 1 - 4자 이름을 지정합니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 TRANSACTION 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ # _ < >

URI=value

서비스 제공자에서, 이 매개변수는 클라이언트가 웹 서비스에 액세스하기 위해 사용하는 상대 URI를 지정합니다. CICS는 DFHWS2LS에 의해 작성된 웹 서비스 바인딩 파일에서 URIMAP 자원을 생성할 때 지정한 값을 사용합니다. 이 매개변수는 URIMAP 정의가 적용되는 URI의 경로 컴포넌트를 지정합니다.

서비스 요청자에서 대상 웹 서비스의 URI는 이 매개변수로 지정되지 않습니다. CICS는 서비스 요청자에 대해 URIMAP 자원을 생성하지 않습니다. 서비스 요청자가 대상 웹 서비스의 URI에 대해 클라이언트 요청을 수행할 때 사용할 서비스 요청자에 대한 자체 URIMAP 자원을 정의할 수 있습니다. 서비스 요청자가 **INVOKE SERVICE** 명령을 발행하면 CICS는 웹 서비스 설명에서 지정한 `wsdl:port`에서(있는 경우) `soap:address` 위치를 사용합니다. 이를 대체하고 **INVOKE SERVICE** 명령에 대해 URIMAP 또는 URI 옵션을 사용하여 다른 URI를 지정할 수 있습니다.

USERID=id

서비스 제공자에서 이 매개변수는 웹 클라이언트가 사용할 수 있는 1 - 8자 사용자 ID를 지정합니다. 애플리케이션 생성 응답 또는 웹 서비스의 경우 이 사용자 ID 아래에 별명 트랜잭션이 첨부됩니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 USERID 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3={NO|YES}

COBOL에서만 사용됩니다. COBOL 언어 구조에서 팩형 10진수 가변 길이의 최대 크기를 제어합니다.

아니오 DFHWS2LS는 COBOL 언어 구조 유형 COMP-3을 생성할 때 팩형 10진수 가변 길이를 18로 제한합니다. 팩형 10진수 크기가 18보다 크면 지정된 유형은 총 18숫자로 제한됨을 나타내는 메시지 DFHPI9022W가 발행됩니다.

예 DFHWS2LS는 COBOL 언어 구조 유형 COMP-3을 생성할 때 최대 크기 31을 지원합니다.

WSADDR-EPR-ANY={TRUE|FALSE}

CICS가 WS-Addressing 엔드포인트 참조(EPR)를 언어 구조의 해당 컴포넌트 파

트로 변환하는지 또는 EPR을 <xsd:any> 유형으로 처리하는지 지정합니다. EPR을 <xsd:any> 유형으로 처리하는 것은 **WSACONTEXT BUILD** API가 EPR XML을 직접 사용할 수 있음을 의미합니다.

FALSE

DFHWS2LS가 일반적으로 작동하며 XML을 고급 언어 구조로 변환합니다.

TRUE

이 옵션을 TRUE로 설정하는 것은 런타임 시 CICS가 전체 EPR을 <xsd:any> 유형으로 처리하고 EPR XML을 애플리케이션이 참조할 수 있는 컨테이너에 배치함을 의미합니다. 애플리케이션은 **WSACONTEXT BUILD** API와 함께 EPR XML을 사용하여 주소 지정 컨텍스트에서 EPR을 구성할 수 있습니다.

이 매개변수는 런타임 레벨 3.0 이상에서만 사용할 수 있습니다.

WSBIND=*value*

웹 서비스 바인딩 파일의 완전한 z/OS UNIX 이름입니다. DFHWS2LS는 파일을 작성하지만 이미 존재하지 않는 경우 디렉토리 구조는 작성하지 않습니다. 기본 파일 확장자는 .wsbind입니다.

WSDL=*value*

웹 서비스 설명을 포함하는 파일의 완전한 z/OS UNIX 이름입니다. WSRR을 사용하여 WSDL 문서를 검색 중인 경우 이 매개변수는 파일 시스템에서 WSDL 문서의 로컬 사본이 작성될 위치를 지정합니다.

WSDL-SERVICE=*value*

한 개의 바인딩 요소에 대해 둘 이상의 서비스 요소가 웹 서비스 설명에 포함되어 있는 경우 사용되는 wsdl:Service 요소를 지정합니다. **BINDING** 매개변수에 대한 값을 지정하는 경우 이 매개변수에 대해 지정하는 서비스 요소는 지정된 바인딩 요소와 일치해야 합니다. WSDL 1.1 또는 WSDL 2.0 문서 둘 다에서 이 매개변수를 사용할 수 있습니다.

WSRR-NAME=*value*

WSRR에서 검색할 WSDL 문서의 이름을 지정합니다. 이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용합니다.

WSRR-NAMESPACE=*value*

WSRR에서 검색할 WSDL 문서의 네임스페이스를 지정합니다. **WSRR-SERVER** 매개변수가 지정되는 경우에 선택적으로 이 매개변수를 사용하여 **WSRR-NAME** 매개변수에 지정된 WSDL 문서 이름을 완전하게 확인할 수 있습니다.

WSRR-PASSWORD=*value*

WSRR에 액세스하기 위해 비밀번호가 필요한 경우에는 이 선택적 매개변수를 사용하십시오.

WSRR-USERNAME 매개변수가 지정되는 경우 이 매개변수도 지정해야 합니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-SERVER=*{domain name:port number|IP address:port number}*

WSRR(IBM WebSphere Service Registry and Repository) 서버의 위치를 지정하려면 이 매개변수를 사용하십시오. 이 매개변수가 지정되면 **WSRR** 매개변수 유효성 검증이 사용됩니다.

WSRR-USERNAME=*value*

WSRR에 액세스하기 위해 사용자 이름을 지정해야 하는 경우 이 선택적 매개변수를 사용하십시오. 이 사용자 이름은 **WSRR**이 소유자 특성을 설정하기 위해 사용됩니다.

이 매개변수는 **WSRR-SERVER** 매개변수가 지정된 경우에만 사용됩니다.

WSRR-VERSION=*value*

WSRR에서 검색할 WSDL 문서의 버전을 지정합니다. 이 매개변수는 **WSRR-SERVER** 매개변수가 지정되는 경우에만 사용할 수 있습니다.

XML-ONLY=*{TRUE|FALSE}*

CICS가 SOAP 메시지의 XML을 애플리케이션 데이터로 변환하는지 여부를 지정합니다. **XML-ONLY** 매개변수를 사용하여 XML 자체를 처리하는 웹 서비스 애플리케이션을 작성하십시오.

TRUE

CICS는 XML으로의 어떠한 변환도 수행하지 않습니다. 이 서비스 요청자 또는 제공자 애플리케이션은 XML과 고급 언어 사이에서 데이터를 맵핑하기 위해 DFHWS-BODY 컨테이너의 콘텐츠를 사용하여 직접 작업해야 합니다.

FALSE

CICS가 XML을 고급 언어로 변환합니다.

이 매개변수는 런타임 레벨 2.1 이상에서만 사용할 수 있습니다.

기타 정보

- DFHLS2SC를 실행하는 사용자 ID는 UNIX System Services를 사용하도록 구성되어야 합니다. 사용자 ID는 CICS z/OS UNIX 파일 구조 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND**, **WSDL** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다.
- JCL의 최대 매개변수 길이는 100자입니다. 이 길이는 **STDPARM** 명령문을 사용하여 늘릴 수 있습니다. 자세한 정보는 z/OS UNIX System Services 사용자 안내서의 내용을 참조하십시오.

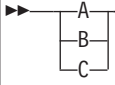
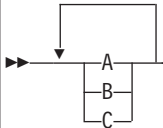
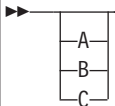
예제

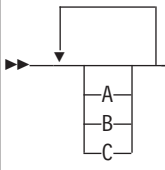
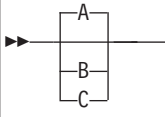
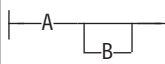
```
//WS2LS JOB  'accounting information',name,MSGCLASS=A
//  SET QT='''
//JAVAPROG EXEC DFHWS2LS,
// TMPFILE=&QT.&SYSUID.&QT
//INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
REQMEM=CPYBK1
RESPMEM=CPYBK2
LANG=COBOL
LOGFILE=/u/exampleapp/wsbind/example.log
MAPPING-LEVEL=3.0
MAPPING-OVERRIDES=UNDERSCORES-AS-HYPHENS
CHAR-VARYING=NULL
INLINE-MAXOCCURS-LIMIT=2
PGMNAME=DFH0XCMN
URI=exampleApp/example
PGMINT=COMMAREA
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbind/example.wsbind
WSDL=/u/exampleapp/wsd1/example.wsd1
/*
```

구문 표기법

구문 표기법은 CICS 명령, 자원 정의 및 기타 여러 항목에 지정할 수 있는 속성 또는 옵션의 사용 가능한 조합을 지정합니다.

구문 표기법에서 사용되는 규칙:

표기법	설명
	필수 대체사항 세트를 선언합니다. 반드시 표시되는 값 중 하나를(하나만) 지정해야 합니다.
	필수 대체사항 세트를 선언합니다. 표시되는 값 중 최소한 하나를 지정해야 합니다. 하나 이상의 값을 어떤 순서로든 지정할 수 있습니다.
	선택적 대체사항 세트를 선언합니다. 표시되는 값 중 하나를 지정하거나 아무 것도 지정하지 않을 수 있습니다.

표기법	설명
	선택적 대체사항 세트를 선언합니다. 아무 값도 지정하지 않거나 표시되는 값 중 하나 또는 하나 이상을 어떤 순서로든 지정할 수 있습니다.
	선택적 대체사항 세트를 선언합니다. 표시되는 값 중 하나를 지정하거나 아무 것도 지정하지 않을 수 있습니다. A 는 아무 값도 지정하지 않은 경우에 사용되는 기본값입니다.
이름 이름: 	구문 표기법의 이름 지정된 섹션에 대한 참조입니다.
$A=value$	A는 표시된 대로 정확히 입력해야 하는 문자를 선언합니다. <i>value</i>는 적절한 값을 지정해야 하는 변수를 선언합니다.

CICS 지원에 대한 �핑 레벨

�핑은 언어 구조와 XML 스키마 간에 정보가 변환되는 방법을 지정하는 규칙 세트입니다. 가장 정교한 �핑을 활용하려면 CICS 지원에서 **MAPPING-LEVEL** 매개변수를 최신 레벨로 설정하는 것이 좋습니다.

각 �핑 레벨은 이전 �핑의 기능을 상속하며 가장 높은 �핑 레벨이 가장 우수한 기능을 제공합니다. 가장 높은 �핑 레벨이 런타임에 데이터 변환을 보다 강력하게 제어하며 특정 데이터 유형 및 XML 요소 지원에 대한 제한사항을 제거합니다.

이전에 해당 레벨에서 사용한 애플리케이션을 다시 배치하려는 경우 **MAPPING-LEVEL** 매개변수를 이전 레벨로 설정할 수 있습니다.

�핑 레벨 3.0

�핑 레벨 3.0은 CICS TS 4.1 리전과 호환됩니다.

이 �핑 레벨은 다음 지원을 제공합니다.

- DFHSC2LS 및 DFHWS2LS가 `xsd:dateTime` 데이터 유형을 CICS ASKTIME 형식에 맵핑합니다.

- DFHLS2WS가 하나의 컨테이너가 아닌 많은 컨테이너를 사용하는 애플리케이션에서 WSDL 문서와 웹 서비스 바인딩을 생성할 수 있습니다.
- 고정 길이 데이터 구조로 설명하는 잘린 데이터 허용. 이 동작은 CICS 지원에 **DATA-TRUNCATION** 매개변수를 사용하여 설정할 수 있습니다.

맵핑 레벨 2.2 이상

맵핑 레벨 2.2는 APAR PK69738 이상이 적용된 CICS TS 3.2 리전과 호환됩니다.

맵핑 레벨이 2.2 이상인 경우 DFHSC2LS 및 DFHWS2LS는 다음 XML 맵핑을 지원합니다.

- 요소의 고정 값
- 대체 그룹
- 추상 데이터 유형
- XML 스키마 <sequence> 요소는 <choice> 요소 내부에 중첩될 수 있습니다.

DFHSC2LS 및 DFHWS2LS는 다음 XML 맵핑에 대한 강화된 지원을 제공합니다.

- 추상 요소
- XML 스키마 <choice> 요소

맵핑 레벨 2.1 이상

맵핑 레벨 2.1은 APAR PK59794 이상이 적용된 CICS TS 3.2 리전과 호환됩니다.

이 맵핑 레벨에는 새 **INLINE-MAXOCCURS-LIMIT** 매개변수와 **CHAR-VARYING** 매개변수의 새 값으로 변수 콘텐츠를 처리하는 방법에 대한 강화된 제어가 포함됩니다.

맵핑 레벨 2.1 이상의 경우 DFHSC2LS 및 DFHWS2LS는 XML 맵핑에 대해 다음과 같은 새로운 또는 향상된 지원을 제공합니다.

- XML 스키마 <any> 요소
- xsd:anyType 유형
- 추상 요소 허용
- **INLINE-MAXOCCURS-LIMIT** 매개변수
- minOccurs 속성

INLINE-MAXOCCURS-LIMIT 매개변수는 가변 반복 목록이 인라인으로 맵핑되는지 여부를 지정합니다. 가변 반복 콘텐츠 인라인 맵핑에 대한 자세한 정보는 가변 요소 배열을 참조하십시오.

XML 스키마 <sequence>, <choice>, <all> 요소에서 minOccurs 속성에 대한 지원이 강화되었습니다. minOccurs="0"인 경우 CICS 지원은 이들 요소를 minOccurs="0" 속성이 모든 하위 요소의 속성인 것처럼 처리합니다.

맵핑 레벨이 2.1 이상인 경우 DFHLS2SC 및 DFHLS2WS는 다음 XML 맵핑을 지원합니다.

- COBOL 및 PL/I의 FILLER 필드는 무시됨
- **CHAR-VARYING** 매개변수에 대한 COLLAPSE 값
- **CHAR-VARYING** 매개변수에 대한 BINARY 값

COBOL 및 PL/I의 FILLER 필드는 무시됩니다. 즉, 생성된 XML 스키마에 나타나지 않으며 런타임에 데이터 구조에서 적절한 값이 유지됩니다.

COLLAPSE로 인해 CICS가 텍스트 필드에서 후속 공백을 무시합니다.

BINARY는 2진 필드에 대한 지원을 제공합니다. 이 값은 COBOL을 XML 스키마로 변환할 때 유용합니다. 이 옵션은 SBCS 문자 배열에서만 사용할 수 있으며 `xsd:string` 필드가 아닌 고정 길이 `xsd:base64Binary` 필드로 배열이 맵핑될 수 있습니다.

맵핑 레벨 1.2 이상

맵핑 레벨 1.2는 CICS TS 3.1 이상 리전과 호환됩니다.

일괄처리 도구의 이 추가 매개변수로 런타임에 문자 및 2진 데이터가 변환되는 방식을 보다 강력하게 제어할 수 있습니다.

- **CHAR-VARYING**
- **CHAR-VARYING-LIMIT**
- **CHAR-MULTIPLIER**
- **DEFAULT-CHAR-MAXLENGTH**

DFHSC2LS 또는 DFHWS2LS에서 **CHAR-MULTIPLIER** 매개변수를 사용하려는 경우, 문자 데이터에 필요한 공백을 계산하기 위해 이 매개변수의 값을 사용한 후 다음 규칙이 적용됨에 유의하십시오.

- DFHSC2LS 및 DFHWS2LS는 다음 맵핑을 제공합니다.
 - 최대 길이가 32 767바이트보다 긴 가변 길이 문자 데이터 유형은 컨테이너에 맵핑됩니다. **CHAR-VARYING-LIMIT** 매개변수를 사용하여 낮은 한계를 설정할 수 있습니다. 언어 구조에서 컨테이너 이름을 저장하기 위한 16바이트 필드가 작성됩니다. 런타임에는 문자 데이터가 컨테이너에 저장되고 컨테이너 이름은 언어 구조에 배치됩니다.
 - 최대 길이가 32 768바이트 미만인 가변 길이 문자 데이터 유형은 C/C++ 및 엔터프라이즈 PL/I를 제외한 모든 언어의 경우 **VARYING** 구조에 맵핑됩니다. C/C++의 경우에는 이러한 데이터 유형이 널 종료 문자열에 맵핑되고 엔터프라이즈 PL/I의 경우에는 이러한 데이터 유형이 **VARYINGZ** 구조에 맵핑됩니다. **CHAR-VARYING** 매개변수를 사용하여 가변 길이 문자 데이터가 맵핑되는 방식을 선택할 수 있습니다.

- 최대 길이가 32 768바이트 미만인 가변 길이 2진 데이터가 모든 언어의 VARYING 구조에 맵핑됩니다. 최대 길이가 32 768바이트 이상인 경우에는 데이터가 컨테이너에 맵핑됩니다. 언어 구조에서 컨테이너 이름을 저장하기 위한 16 바이트 필드가 작성됩니다. 런타임에는 2진 데이터가 컨테이너에 저장되고 컨테이너 이름은 언어 구조에 배치됩니다.

XML 스키마에 연관된 길이가 없는 문자 데이터 유형이 있는 경우에는 DFHWS2LS 또는 DFHSC2LS에서 **DEFAULT-CHAR-MAXLENGTH** 매개변수를 사용하여 기본 길이를 지정할 수 있습니다.

DFHLS2SC 및 DFHLS2WS는 다음 맵핑을 제공합니다.

- 문자 필드는 xsd:string 데이터 유형에 맵핑되며 런타임에 고정 길이 필드 또는 널 종료 문자열로 처리될 수 있습니다. **CHAR-VARYING** 매개변수를 사용하여 PL/I를 제외한 모든 언어에 대해 런타임에 가변 길이 문자 데이터가 처리되는 방식을 선택할 수 있습니다.
- 데이터의 최대 길이가 32 767바이트보다 크거나 길이가 정의되지 않은 경우 Base64Binary 데이터 유형이 컨테이너에 맵핑됩니다. 데이터 길이가 32 767 이하인 경우에는 base64Binary 데이터 유형이 모든 언어의 VARYING 구조에 맵핑됩니다.

맵핑 레벨 1.1 이상:

맵핑 레벨 1.1은 CICS TS 3.1 이상 리전과 호환됩니다.

이 맵핑 레벨은 특히 maxLength 및 minLength 속성이 XML 스키마에서 다른 값으로 정의된 가변 길이의 데이터를 맵핑하는 경우 XML 문자 및 2진 데이터 유형의 향상된 맵핑을 제공합니다. 데이터는 다음과 같이 처리됩니다.

- 16 MB보다 큰 고정 길이의 문자 및 2진 데이터 유형은 PL/I를 제외한 모든 언어의 경우 컨테이너에 맵핑됩니다. PL/I의 경우에는 32 767바이트보다 큰 고정 길이 문자 및 2진 데이터 유형이 컨테이너에 맵핑됩니다. 언어 구조에서 컨테이너 이름을 저장하기 위한 16바이트 필드가 작성됩니다. 런타임에는 고정 길이 데이터가 컨테이너에 저장되고 컨테이너 이름은 언어 구조에 배치됩니다.

컨테이너는 길이가 가변적이므로 컨테이너에 맵핑되는 고정 길이 데이터가 XML 스키마 또는 웹 서비스 설명에 지정된 고정 길이와 일치시키기 위해 잘리거나 공백 또는 널로 채워지지 않습니다. 데이터 길이가 중요한 경우 길이를 확인하는 애플리케이션을 작성하거나 CICS 리전에서 유효성 검증을 활성화할 수 있습니다. SOAP 및 XML 유효성 검증 모두 성능에 중요한 영향을 미칩니다.

- XML 스키마 <list> 및 <union> 데이터 유형은 문자 필드에 맵핑됩니다.
- 스키마 정의 XML 속성은 무시되지 않고 맵핑됩니다. 각 XML 요소에 최대 255개 속성이 허용됩니다. 자세한 정보는 XML 속성에 대한 지원을 참조하십시오.

- xsi:nil 속성은 지원됩니다. 자세한 정보는 XML 속성에 대한 지원을 참조하십시오.

맵핑 레벨 1.1 전용

맵핑 레벨 1.1은 CICS TS 3.1 이상 리전과 호환됩니다.

이 맵핑 레벨은 특히 maxLength 및 minLength 속성이 XML 스키마에서 다른 값으로 정의된 가변 길이의 데이터를 맵핑하는 경우 XML 문자 및 2진 데이터 유형의 향상된 맵핑을 제공합니다. 데이터는 다음과 같이 처리됩니다.

- 가변 길이 2진 데이터 유형은 컨테이너에 맵핑됩니다. 언어 구조에서 컨테이너 이름을 저장하기 위한 16바이트 필드가 작성됩니다. 런타임에는 2진 데이터가 컨테이너에 저장되고 컨테이너 이름은 언어 구조에 배치됩니다.
- 최대 길이가 32 767바이트보다 긴 가변 길이 문자 데이터 유형은 컨테이너에 맵핑됩니다. 언어 구조에서 컨테이너 이름을 저장하기 위한 16바이트 필드가 작성됩니다. 런타임에는 문자 데이터가 컨테이너에 저장되고 컨테이너 이름은 언어 구조에 배치됩니다.
- 16MB 미만의 고정 길이를 갖는 문자 및 2진 데이터 유형은 PL/I를 제외한 모든 언어의 경우 고정 길이 필드에 맵핑됩니다. PL/I의 경우 32 767바이트 이하의 고정 길이 문자 및 2진 데이터 유형은 고정 길이 필드에 맵핑됩니다.
- CICS는 base64Binary 형식이 아닌 hexBinary 형식으로 데이터를 인코딩하고 디코딩합니다. XML 스키마의 Base64Binary 데이터 유형은 언어 구조에서 필드에 맵핑됩니다. 필드의 크기는 다음 공식을 사용하여 계산됩니다. $4 \times (\text{ceil}(z/3))$ 여기서,
 - z 는 XML 스키마에서 데이터 유형의 길이입니다.
 - $\text{ceil}(x)$ 은 x 보다 크거나 같은 가장 작은 정수입니다.

z 의 길이가 24 566바이트보다 큰 경우에는 결과 언어 구조가 컴파일되지 않습니다. 24 566바이트보다 큰 base64Binary 데이터가 있는 경우에는 맵핑 레벨 1.2를 사용하는 것이 좋습니다. 맵핑 레벨이 1.2인 경우 언어 구조에서 필드를 사용하지 않고 base64Binary 데이터를 컨테이너에 맵핑할 수 있습니다.

맵핑 레벨 1.0 전용

맵핑 레벨 1.0은 CICS TS 3.1 이상 리전과 호환됩니다.

다음 제한사항이 적용됩니다. 이 제한사항은 이후 맵핑 레벨에서 수정되었습니다.

- DFHSC2LS 및 DFHWS2LS는 XML 스키마의 문자 및 2진 데이터 유형을 언어 구조의 고정 길이 필드에 맵핑합니다. 다음 부분 XML 스키마를 예로 들 수 있습니다.

```
<xsd:element name="example">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
```

```

        <xsd:maxLength value="33000"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:element>

```

해당 부분 XML 스키마는 COBOL 언어 구조에 다음과 같이 나타납니다.

```
15 example      PIC X(33000)
```

- CICS는 base64Binary 형식이 아닌 hexBinary 형식으로 데이터를 인코딩하고 디코딩합니다. DFHSC2LS 및 DFHWS2LS는 Base64Binary 데이터를 고정 길이 문자 필드에 맵핑하며 해당 콘텐츠는 애플리케이션 프로그램이 인코딩 또는 디코딩해야 합니다.
- DFHSC2LS 및 DFHWS2LS는 처리 중에 XML 속성을 무시합니다.
- DFHLS2SC 및 DFHLS2WS는 언어 구조에서 문자 및 2진 필드를 고정 길이 필드로 해석하며 해당 필드를 maxLength 속성을 갖는 XML 요소에 맵핑합니다. 사용할 수 있는 데이터가 충분하지 않은 경우 런타임에 언어 구조의 필드가 공백 또는 널로 채워집니다.

고급 언어 및 XML 스키마 맵핑

CICS 지원을 사용하여 고급 언어 구조와 XML 스키마 또는 WSDL 문서 사이에 맵핑을 생성하십시오. CICS 보조는 고급 언어 데이터 구조 또는 그 반대에서 XML 스키마 또는 WSDL 문서도 생성합니다.

DFHSC2LS 및 DFHLS2SC 유틸리티 프로그램은 총괄적으로 CICS XML 보조로 알려집니다. DFHWS2LS 및 DFHLS2WS 유틸리티 프로그램은 총괄적으로 CICS 웹 서비스 보조로 알려집니다.

- DFHLS2SC 및 DFHLS2WS는 고급 언어 구조를 XML 스키마 및 WSDL 문서에 각각 맵핑합니다.
- DFHSC2LS 및 DFHWS2LS는 XML 스키마 및 WSDL 문서를 고급 언어 구조에 맵핑합니다.

두 개의 맵핑이 대칭적이지 않습니다.

- DFHLS2SC 또는 DFHLS2WS를 가진 언어 데이터 구조를 처리하고 보완적 유틸리티 프로그램을 가진 결과로 생기는 XML 스키마 또는 WSDL 문서(각각 DFHSC2LS 또는 DFHWS2LS)를 처리하는 경우, 최종 데이터 구조가 원래의 구조와 같은 것으로 예상하지 마십시오. 그러나, 최종 데이터 구조는 논리적으로 원래의 구조에 해당합니다.
- DFHSC2LS 또는 DFHWS2LS를 가진 XML 스키마 또는 WSDL 문서를 처리하고 보완적 유틸리티 프로그램을 가진 결과로 생기는 언어 구조(각각 DFHLS2SC 또는 DFHLS2WS)를 처리하는 경우, 최종 XML 스키마 또는 WSDL 문서의 XML 스키마가 원래와 같은 것으로 예상하지 마십시오.

- 일부 경우에, DFHSC2LS 및 DFHWS2LS는 DFHLS2SC 및 DFHLS2WS가 지원하지 않는 언어 구조를 생성합니다.

CICS가 지원하는 언어 컴파일러에서 구현되는 것처럼 언어의 규칙에 따라 DFHLS2SC 및 DFHLS2WS가 처리하는 언어 구조를 코드화해야 합니다.

CICS 지원 사용 시 데이터 맵핑 제한사항

CICS는 WSDL 버전 1.1 또는 2.0을 준수하는 XML 스키마 또는 WSDL 문서와 고급 언어 구조 간의 양방향 데이터 맵핑을 지원합니다. 그러나 여기에는 특정 제한사항이 적용됩니다. 이러한 제한사항은 DFHWS2LS 및 DFHSC2LS 도구에만 적용되며 맵핑 레벨에 따라 다릅니다.

모든 맵핑 레벨의 제한사항

- 리터럴 인코딩을 사용하는 SOAP 바인딩만 지원됩니다. 따라서 `use` 속성의 값을 `literal`로 설정해야 합니다. `use="encoded"`는 지원되지 않습니다.
- 데이터 유형 정의는 XML 스키마 정의 언어(XSD)를 사용하여 인코딩해야 합니다. 스키마에서, SOAP 메시지에서 사용되는 데이터 유형이 명시적으로 선언되어야 합니다.
- 웹 서비스 설명에서 일부 키워드의 길이가 제한됩니다. 예를 들어, 조작, 바인딩, 파트 이름은 255자로 제한됩니다. 경우에 따라 최대 조작 이름의 길이가 약간 짧을 수 있습니다.
- 웹 서비스 설명에서 정의된 SOAP 결함은 무시됩니다. 서비스 제공자 애플리케이션으로 SOAP 결함 메시지를 보내려면 **EXEC CICS SOAPFAULT** 명령을 사용하십시오.
- DFHWS2LS 및 DFHSC2LS는 특정 범위의 단일 `<xsd:any>` 요소만 지원합니다. 예를 들어, 다음 스키마 단편은 지원되지 않습니다.

```
<xsd:sequence>
  <xsd:any/>
  <xsd:any/>
</xsd:sequence>
```

여기서 `<xsd:any>`는 필요한 경우 `minOccurs` 및 `maxOccurs`를 지정할 수 있습니다. 예를 들어, 다음 스키마 단편이 지원됩니다.

```
<xsd:sequence>
  <xsd:any minOccurs="2" maxOccurs="2"/>
</xsd:sequence>
```

- 주기적 참조는 지원되지 않습니다. 예를 들어, 유형 A가 유형 B를 포함하고 유형 B가 다시 유형 A를 포함하는 경우입니다.
- `<xsd:choice>`, `<xsd:sequence>`, `<xsd:group>` 또는 `<xsd:all>` 요소와 같은 그룹 요소에서는 반복이 지원되지 않습니다. 예를 들어, 다음 스키마 단편은 지원되지 않습니다.

```
<xsd:choice maxOccurs="2">
  <xsd:element name="name1" type="string"/>
</xsd:choice>
```

예외는 맵핑 레벨이 2.1 이상인 경우입니다. 이 경우 이 요소에서 maxOccurs="1" 과 minOccurs="0"이 지원됩니다.

- DFHSC2LS 및 DFHWS2LS는 맵핑 레벨 2.2 이상에서 상위 요소 또는 유형이 추상으로 정의된 경우를 제외하고, xsi:type 속성 또는 대체 그룹에서 XML 스키마에서 있는 선언된 데이터 유형과 요소에서 파생된 SOAP 메시지의 데이터 유형과 요소를 지원하지 않습니다.
- <xsd:choice> 요소 내부에 임베드된 <xsd:sequence> 및 <xsd:group> 요소는 맵핑 레벨 2.2 이하에서는 지원되지 않습니다. <xsd:choice> 요소 내부에 임베드된 <xsd:choice> 및 <xsd:all> 요소는 지원되지 않습니다.

맵핑 레벨 1.1 이상의 지원 향상

맵핑 레벨이 1.1 이상인 경우 DFHWS2LS는 다음 XML 요소 및 요소 유형에 대한 지원을 제공합니다.

- <xsd:list> 요소
- <xsd:union> 요소
- xsd:anySimpleType 유형
- <xsd:attribute> 요소. 맵핑 레벨이 1.0인 경우에는 이 요소가 무시됩니다.

맵핑 레벨 2.1 이상의 지원 향상

맵핑 레벨이 2.1 이상인 경우 DFHWS2LS는 다음 XML 요소와 요소 속성을 지원합니다.

- <xsd:any> 요소
- xsd:anyType 유형
- 추상 요소. 이전 맵핑 레벨에서는 추상 요소가 상속 계층 구조의 비터미널 유형으로만 지원됩니다.
- <xsd:all>, <xsd:choice>, <xsd:sequence> 요소의 maxOccurs 및 minOccurs 속성 (maxOccurs="1" 및 minOccurs="0"인 경우만 해당)
- "FILLER" 필드(COBOL)와 "*" 필드(PL/I)가 억제됩니다. 이 필드는 생성된 WSDL에 나타나지 않으며 런타임에 데이터 구조에서 적절한 갭이 유지됩니다.

맵핑 레벨 2.2 이상의 지원 향상

맵핑 레벨이 2.2 이상인 경우 DFHSC2LS 및 DFHWS2LS는 <xsd:choice> 요소에 대한 향상된 지원을 제공합니다. 예를 들어, <xsd:choice> 요소에서 최대 255개 옵션을 지원합니다. <xsd:choice> 지원에 대한 자세한 정보는 141 페이지의 『<xsd:choice>에 대한 지원』을 참조하십시오.

맵핑 레벨이 2.2 이상인 경우 CICS 지원이 다음 XML 맵핑을 지원합니다.

- 대체 그룹
- 요소의 고정 값
- 추상 데이터 유형

맵핑 레벨이 2.2 이상인 경우 <xsd:choice> 요소에 임베드된 <xsd:sequence> 및 <xsd:group> 요소가 지원됩니다. 예를 들어, 다음 스키마 단편이 지원됩니다.

```
<xsd:choice>
  <xsd:element name="name1" type="string"/>
  <xsd:sequence/>
</xsd:choice>
```

SOAP 메시지의 상위 요소 또는 유형이 추상으로 정의되는 경우, DFHSC2LS 및 DFHWS2LS는 XML 스키마에서 선언된 데이터 유형 및 요소에서 파생되는 데이터 유형과 요소를 지원합니다.

맵핑 레벨 3.0 이상의 지원 향상

맵핑 레벨이 3.0 이상인 경우 CICS 지원이 다음 맵핑 개선사항을 지원합니다.

- DFHSC2LS 및 DFHWS2LS가 xsd:dateTime 데이터 유형을 CICS ASKTIME 형식에 맵핑합니다.
- DFHLS2WS가 하나의 컨테이너가 아닌 많은 컨테이너를 사용하는 애플리케이션에서 WSDL 문서와 웹 서비스 바인딩을 생성할 수 있습니다.
- 고정 길이 데이터 구조로 설명하는 잘린 데이터 허용. 이 동작은 CICS 지원에 **DATA-TRUNCATION** 매개변수를 사용하여 설정할 수 있습니다.

COBOL 대 XML 스키마 맵핑

DFHLS2SC 및 DFHLS2WS 유틸리티 프로그램은 COBOL 데이터 구조와 XML 스키마 정의 간의 맵핑을 지원합니다.

COBOL 이름은 다음 규칙에 따라 XML 이름으로 변환됩니다.

1. 중복된 이름은 하나 이상의 숫자를 추가하여 고유한 이름이 됩니다.

예를 들어, year의 두 인스턴스는 year와 year1이 됩니다.

2. 하이픈은 밑줄 문자로 바뀝니다. 연속 하이픈 문자열은 연속 밑줄로 바뀝니다.

예를 들어, current-user--id는 current_user__id가 됩니다.

3. 하이픈으로 분리되고 대문자만 포함하는 이름의 세그먼트는 소문자로 변환됩니다.

예를 들어, CA-REQUEST-ID는 ca_request_id가 됩니다.

4. 숫자로 시작하는 이름에는 선행 밑줄 문자가 추가됩니다.

예를 들어, 9A-REQUEST-ID는 _9a_request_id가 됩니다.

CICS는 다음 표에 따라 COBOL 데이터 설명 요소를 스키마 요소로 맵핑합니다. 표에 없는 COBOL 데이터 설명 요소는 DFHLS2SC 또는 DFHLS2WS에서 지원되지 않습니다. 다음 제한사항도 적용됩니다.

- 레벨 번호가 66 및 77인 데이터 설명 항목은 지원되지 않습니다. 레벨 번호가 88인 데이터 설명 항목은 무시됩니다.
- 데이터 설명 항목에 대한 다음 절은 지원되지 않습니다.

OCCURS DEPENDING ON

OCCURS INDEXED BY

REDEFINES

RENAMES; 즉 레벨 66

DATE FORMAT

- 데이터 설명 항목에 대한 다음 절은 무시됩니다.

BLANK WHEN ZERO

JUSTIFIED

VALUE

- SIGN 절 SIGN TRAILING은 지원됩니다. SIGN 절 SIGN LEADING은 DFHLS2SC 또는 DFHLS2WS에서 지정된 맵핑 레벨이 1.2 이상인 경우에만 지원됩니다.
- SEPARATE CHARACTER는 SIGN TRAILING 및 SIGN LEADING 절 둘 다에 대해 1.2 이상의 맵핑 레벨에서 지원됩니다.
- USAGE 절에서 다음 구문이 지원되지 않습니다.

OBJECT REFERENCE

POINTER

FUNCTION-POINTER

PROCEDURE-POINTER

- USAGE 절에서 다음 구문은 1.2 이상의 맵핑 레벨에서 지원됩니다.

COMPUTATIONAL-1

COMPUTATIONAL-2

- DISPLAY 및 COMPUTATIONAL-5 데이터 설명 항목에 지원되는 유일한 PICTURE 문자는 9, S, Z입니다.
- PACKED-DECIMAL 데이터 설명 항목에 지원되는 PICTURE 문자는 9, S, V, Z입니다.
- 편집된 숫자 데이터 설명 항목에 지원되는 유일한 PICTURE 문자는 9와 Z입니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 NULL로 설정된 경우, 문자 배열은 xsd:string으로 맵핑되고 널(null) 종료 문자열로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 BINARY로 설정된 경우, 문자 배열은 xsd:base64Binary로 맵핑되고 2진 데이터로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 COLLAPSE로 설정된 경우, 문자열의 후행 공백이 무시됩니다.

COBOL 데이터 설명	스키마 simpleType
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> 여기서 $m = n$
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> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.
PIC S9(<i>z</i>) DISPLAY 여기서 $5 \leq z \leq 9$	<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> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.
PIC S9(<i>z</i>) DISPLAY 여기서 $9 < z$	<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> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.

COBOL 데이터 설명	스키마 simpleType
PIC 9 DISPLAY PIC 99 DISPLAY PIC 999 DISPLAY PIC 9999 DISPLAY	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="n"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.
PIC 9(z) DISPLAY 여기서 $5 \leq z \leq 9$	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="n"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.
PIC 9(z) DISPLAY 여기서 $9 < z$	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="n"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 n은 '9' 문자의 패턴으로 표현 가능한 최대값입니다.
PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY 여기서 $n \leq 4$ 입니다.	<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 여기서 $5 \leq n \leq 9$ 입니다.	<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 여기서 $9 < n$ 입니다.	<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 여기서 $n \leq 4$ 입니다.	<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 여기서 $5 \leq n \leq 9$ 입니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>

COBOL 데이터 설명	스키마 simpleType
PIC 9(<i>n</i>) COMP PIC 9(<i>n</i>) COMP-4 PIC 9(<i>n</i>) COMP-5 PIC 9(<i>n</i>) BINARY 여기서 9 < <i>n</i> 입니다.	<xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType>
PIC S9(<i>m</i>)V9(<i>n</i>) COMP-3	<xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" <i>p</i> "/> <xsd:fractionDigits value=" <i>n</i> "/> </xsd:restriction> </xsd:simpleType> 여기서 $p = m + n$ 입니다.
PIC 9(<i>m</i>)V9(<i>n</i>) COMP-3 PIC S9(<i>m</i>) COMP-3 DATETIME=PACKED15인 경우 맵핑 레벨 3.0에서 지원됨	<xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" <i>p</i> "/> <xsd:fractionDigits value=" <i>n</i> "/> <xsd:minInclusive value="0"/> </xsd:restriction> </xsd:simpleType> 여기서 $p = m + n$ 입니다. <xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType> 시간소인 형식은 CICS ABSTIME입니다.
PIC S9(<i>m</i>)V9(<i>n</i>) DISPLAY 맵핑 레벨 1.2 이상에서 지원됨	<xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" <i>p</i> "/> <xsd:fractionDigits value=" <i>n</i> "/> </xsd:restriction> </xsd:simpleType> 여기서 $p = m + n$ 입니다.
COMP-1 맵핑 레벨 1.2 이상에서 지원됨	<xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType>
COMP-2 맵핑 레벨 1.2 이상에서 지원됨	<xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType>

XML 스키마 대 COBOL 맵핑

DFHSC2LS 및 DFHWS2LS 유틸리티 프로그램은 XML 스키마 정의와 COBOL 데이터 구조 간 맵핑을 지원합니다.

CICS 지원은 다음 규칙을 사용하여 스키마 요소 이름에서 COBOL 변수의 고유하고 올바른 이름을 생성합니다.

1. COBOL 예약어 앞에는 'X'가 붙습니다.

예를 들어, DISPLAY는 XDISPLAY가 됩니다.

2. A-Z, a-z, 0-9 또는 하이픈 이외의 문자는 'X'로 대체됩니다.

예를 들어, monthly_total은 monthlyXtotal이 됩니다. **MAPPING-OVERRIDES** 매개변수를 사용하여 다른 문자 처리 방법을 변경할 수 있습니다. 예를 들어, 값 **UNDERScores-AS-HYPHENS**를 설정하는 경우, XML의 모든 밑줄은 X가 아닌 하이픈으로 변환됩니다. 따라서 monthly_total은 monthly-total이 됩니다.

3. 마지막 문자가 하이픈이면 'X'로 대체됩니다.

예를 들어, ca-request-는 ca-requestX가 됩니다.

4. 스키마가 변수의 카디널리티가 가변적이고(즉, minOccurs 및 maxOccurs가 다른 값으로 xsd:element에 지정됨) 스키마 요소 이름이 23자보다 긴 것으로 지정하면 해당 길이에서 잘립니다.

스키마가 변수의 카디널리티가 고정적이고 스키마 요소 이름이 28자보다 긴 것으로 지정하면 해당 길이에서 잘립니다.

5. 동일한 범위의 중복된 이름은 이름의 두 번째 및 후속 인스턴스에 하나 또는 두 개 숫자를 추가하여 고유한 이름이 됩니다.

예를 들어, year의 세 가지 인스턴스는 year, year1, year2가 됩니다.

6. 문자열 _cont 또는 _num에는 다섯 개 문자가 예약되어 스키마가 변수의 카디널리티가 가변적인 것으로 지정하는 경우(즉, minOccurs 및 maxOccurs 속성이 다른 값으로 지정되는 경우) 사용됩니다.

자세한 정보는 132 페이지의 『요소의 가변 배열』을 참조하십시오.

7. 속성의 경우 이전 규칙이 요소 이름에 적용됩니다. 접두부 attr-이 요소 이름에 추가되고 그 다음에 -value 또는 -exist가 나타납니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다. 자세한 정보는 137 페이지의 『XML 속성에 대한 지원』을 참조하십시오.

nilable 속성은 특수 규칙을 갖습니다. 접두부 attr-이 추가되지만 요소 이름 처음에 nil-도 추가됩니다. 요소 이름 다음에는 -value가 나타납니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다.

결과 이름의 총 길이는 30자 이하입니다.

DFHSC2LS 및 DFHWS2LS는 다음 표에 따라 지정된 �핑 레벨을 사용하여 COBOL 데이터 설명 요소에 스키마 유형을 �핑합니다. 다음 사항에 주의하십시오.

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 NULL로 설정되면 가변 길이 문자 데이터가 널 종료 문자열에 맵핑되고 널 종료자에 추가 문자가 할당됩니다.

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 YES로 설정되면 가변 길이 문자 데이터가 관련 두 요소(길이 피드와 데이터 필드)에 맵핑됩니다. 예를 들어, 다음과 같습니다.

```
<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"/>
```

다음에 맵핑됩니다.

```
15 textString-length PIC S9999 COMP-5 SYNC
15 textString          PIC X(10000)
```

스키마 단순 유형	COBOL 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 2.0 이하: 지원되지 않음 맵핑 레벨 2.1: 지원됨
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: PIC X(255)
<pre><xsd:simpleType> <xsd:restriction base="xsd:type" <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. <pre>string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY hexBinary</pre>	모든 맵핑 레벨: PIC X(z)

스키마 단순 유형	COBOL 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:type" </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. duration date time gDay gMonth gYear gMonthDay gYearMonth	모든 맵핑 레벨: PIC X(32)
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.2 이하: PIC X(32) 맵핑 레벨 2.0 이상: PIC X(40) 맵핑 레벨 3.0 이상: PIC S9(15) COMP-3 형식은 CICS ABSTIME입니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. byte unsignedByte	모든 맵핑 레벨: PIC X DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC S9999 COMP-5 SYNC 또는 PIC S9999 DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC 9999 COMP-5 SYNC 또는 PIC 9999 DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC S9(18) COMP-3

스키마 단순 유형	COBOL 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC S9(9) COMP-5 SYNC 또는 PIC S9(9) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC 9(9) COMP-5 SYNC 또는 PIC 9(9) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC S9(18) COMP-5 SYNC 또는 PIC S9(18) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC 9(18) COMP-5 SYNC 또는 PIC 9(18) DISPLAY
<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="m"> <xsd:fractionDigits value="n"> </xsd:fractionDigits> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC 9(p)V9(n) COMP-3 여기서 $p = m - n$입니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: PIC X DISPLAY 값 x'00'은 false를 의미하고 x'01'은 true를 의미합니다.
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: PIC X(255)
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: PIC X(255)

스키마 단순 유형	COBOL 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> 길이 지정되지 않은 경우.	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1: PIC X(y) 여기서 $y = 4 \times (\text{ceil}(z/3))$. $\text{ceil}(x)$은 x보다 크거나 같은 가장 작은 정수입니다. 맵핑 레벨 1.2 이상: PIC X(z) 여기서 길이는 고정적입니다. PIC X(16) 길이 지정되지 않은 경우. 이 필드는 2진 데이터를 저장하는 컨테이너의 16바이트 이름을 갖습니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: PIC X(32) 맵핑 레벨 1.2 이상: COMP-1
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: PIC X(32) 맵핑 레벨 1.2 이상: COMP-2

테이블에 표시되는 스키마 유형 중 일부는 minInclusive 및 maxInclusive 패킷에 지정되는 값(있는 경우)에 따라 COMP-5 SYNC 또는 DISPLAY의 COBOL 형식에 맵핑됩니다.

- 부호 있는 유형(short, int, long)의 경우 다음이 지정되면 DISPLAY가 사용됩니다.

```
<xsd:minInclusive value="-a"/>
<xsd:maxInclusive value="a"/>
```

여기서 a 는 '9' 문자열입니다.

- 부호 없는 유형(unsignedShort, unsignedInt, unsignedLong)의 경우 DISPLAY는 다음이 지정된 경우 사용됩니다.

```
<xsd:minInclusive value="0"/>
<xsd:maxInclusive value="a"/>
```

여기서 a 는 '9' 문자열입니다.

다른 모든 값이 지정되거나 값이 지정되지 않은 경우에는 COMP-5 SYNC가 사용됩니다.

C 및 C++ 대 XML 스키마 맵핑

DFHLS2SC 및 DFHLS2WS 유틸리티 프로그램은 C 및 C++ 데이터 유형과 XML 스키마 정의 간의 맵핑을 지원합니다.

다음 규칙에 따라 C 및 C++ 이름이 XML 이름으로 변환됩니다.

1. XML 요소 이름에서 유효하지 않은 문자는 'X'로 바꿉니다.

예를 들어, monthly-total은 monthlyXtotal이 됩니다.

2. 중복된 이름은 하나 이상의 숫자를 추가하여 고유한 이름이 됩니다.

예를 들어, year의 두 인스턴스는 year와 year1이 됩니다.

DFHLS2SC 및 DFHLS2WS는 다음 표에 따라 C 및 C++ 데이터 유형을 스키마 요소로 맵핑합니다. 표에 없는 C 및 C++ 유형은 DFHLS2SC 또는 DFHLS2WS에서 지원되지 않습니다. `_Packed` 규정자는 구조에 대해 지원됩니다. 다음과 같은 제한사항이 적용됩니다.

- 헤더 파일은 최상위 레벨 struct 인스턴스를 포함해야 합니다.
- 자체를 멤버로 포함하는 구조 유형은 선언할 수 없습니다.
- 다음 C 및 C++ 데이터 유형은 지원되지 않습니다.

decimal
long double
wchar_t(C++만 해당)

- 다음 문자가 헤더 파일에 있는 경우에는 무시됩니다.

저장영역 클래스 지정자:

auto
register
static
extern
mutable

규정자

const
volatile
_Export(C++만 해당)

함수 지정자

inline(C++만 해당)
virtual(C++만 해당)

초기값

- 헤더 파일에 다음 항목이 포함되지 않아야 합니다.

공용체

클래스 선언

열거 데이터 유형

포인터 유형 변수

템플릿 선언

사전 정의된 매크로. 즉, 해당 이름이 두 개의 밑줄 문자(_)로 시작하고 끝나는 매크로

행 연속 시퀀스(줄 바꾸기 문자 바로 앞에 오는 \ 기호)

프로토타입 함수 선언자

프리프로세서 지시문

비트 필드

`_cdecl`(또는 `_cdecl`) 키워드(C++만 해당)

- 애플리케이션 프로그래머는 `int`가 4바이트로 맵핑되도록 32비트 컴파일러를 사용해야 합니다.
- 다음의 C++ 예약 키워드는 지원되지 않습니다.

`explicit`

`using`

`namespace`

`typename`

`typeid`

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 NULL로 설정된 경우, 문자 배열은 `xsd:string`으로 맵핑되고 널(null) 종료 문자 열로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 BINARY로 설정된 경우, 문자 배열은 `xsd:base64Binary`로 맵핑되고 2진 데이터로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 COLLAPSE로 설정된 경우, 문자열에 대해 `<xsd:whiteSpace value="collapse"/>`가 생성됩니다.

C 및 C++ 데이터 유형	스키마 <code>simpleType</code>
<code>char[z]</code>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType> </pre>

C 및 C++ 데이터 유형	스키마 simpleType
char[8] DATETIME=PACKED15인 경우 맵핑 레벨 3.0 이상에서 지원됨	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre> 시간소인 형식은 CICS ABSTIME입니다.
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++만 해당)	<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>
float 맵핑 레벨 1.2 이상에서 지원됨	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
double 맵핑 레벨 1.2 이상에서 지원됨	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

XML 스키마 대 C 및 C++ 맵핑

DFHSC2LS 및 DFHWS2LS 유틸리티 프로그램은 각 웹 서비스 설명에 포함된 XML 스키마 정의와 C 및 C++ 데이터 유형 간의 맵핑을 지원합니다.

CICS 지원은 다음 규칙을 사용하여 스키마 요소 이름에서 C 및 C++ 변수의 고유하고 올바른 이름을 생성합니다.

1. A-Z, a-z, 0-9 또는 _ 이외의 문자는 'X'로 대체됩니다.

예를 들어, monthly-total은 monthlyXtotal이 됩니다.

2. 첫 번째 문자가 영문자가 아니면 선행 'X'로 대체됩니다.

예를 들어, _monthlysummary는 Xmonthlysummary가 됩니다.

3. 스키마 요소 이름이 50자보다 길면 해당 길이로 잘립니다.
4. 동일한 범위의 중복된 이름은 하나 이상의 숫자를 추가하여 고유하게 됩니다.

예를 들어, year의 두 인스턴스는 year와 year1이 됩니다.

5. 문자열 _cont 또는 _num에는 다섯 개 문자가 예약되어 스키마가 변수의 카디널리티가 가변적인 것으로 지정하는 경우(즉, minOccurs 및 maxOccurs가 xsd:element에서 지정되는 경우) 사용됩니다.

자세한 정보는 132 페이지의 『요소의 가변 배열』을 참조하십시오.

6. 속성의 경우 이전 규칙이 요소 이름에 적용됩니다. 접두부 attr_이 요소 이름에 추가되고 그 다음에 _value 또는 _exist가 옵니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다.

nillable 속성은 특수 규칙을 갖습니다. 접두부 attr_이 추가되지만 요소 이름 처음에 nil-도 추가됩니다. 요소 이름 다음에는 -value가 나타납니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다.

결과 이름의 총 길이는 57자 이하입니다.

FHSC2LS 및 DFHWS2LS는 다음 표에 따라 스키마 유형을 C 및 C++ 데이터 유형에 매핑합니다. 다음 규칙도 적용됩니다.

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 NULL로 설정되면 가변 길이 문자 데이터가 널 종료 문자열에 매핑되고 널 종료자에 추가 문자가 할당됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 YES로 설정되면 가변 길이 문자 데이터가 관련 두 요소(길이 피드와 데이터 필드)에 매핑됩니다.

스키마 simpleType	C 및 C++ 데이터 유형
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	매핑 레벨 2.0 이하: 지원되지 않음 매핑 레벨 2.1 이상: 지원됨

스키마 simpleType	C 및 C++ 데이터 유형
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: char[255]
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. - string - normalizedString - token - Name - NMTOKEN - language - NCName - ID - IDREF - ENTITY - hexBinary	모든 맵핑 레벨: char[z]
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. - duration - date - decimal - time - gDay - gMonth - gYear - gMonthDay - gYearMonth	모든 맵핑 레벨: char[32]

스키마 simpleType	C 및 C++ 데이터 유형
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.2 이하: char[32] 맵핑 레벨 2.0 이상: char[40] 맵핑 레벨 3.0 이상: char[8] 시간소인 형식은 CICS ABSTIME입니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: signed char
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: char
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: short
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: unsigned short
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: char[33]
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: int
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: unsigned int
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: long long
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: unsigned long long
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: bool(C++만 해당) short(C만 해당)

스키마 simpleType	C 및 C++ 데이터 유형
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: char[255]
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1 이상: char[255]
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value="z"/> </xsd:restriction> </xsd:simpleType></pre> 길이가 정의되지 않는 경우	맵핑 레벨 1.1 이하: char[y] 여기서 $y = 4 \times (\text{ceil}(z/3))$. $\text{ceil}(x)$은 x보다 크거나 같은 가장 작은 정수입니다. 맵핑 레벨 1.2 이상: char[z] 여기서 길이는 고정적입니다. char[16] 길이가 정의되지 않은 경우 2진 데이터를 저장하는 컨테이너의 이름입니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: char[32] 맵핑 레벨 1.2 이상: float(*)
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.0 이하: char[32] 맵핑 레벨 1.2 이상: double(*)

PL/I 대 XML 스키마 맵핑

DFHLS2SC 및 DFHLS2WS 유틸리티 프로그램은 PL/I 데이터 구조와 XML 스키마 정의 간의 맵핑을 지원합니다. Enterprise PL/I 컴파일러와 이전 PL/I 컴파일러가 다르므로 두 개의 언어 옵션인 PLI-ENTERPRISE 및 PLI-OTHER가 지원합니다.

다음 규칙에 따라 PL/I 이름이 XML 이름으로 변환됩니다.

1. XML 요소 이름에서 유효하지 않은 문자는 'x'로 바꿉니다.

예를 들어 monthly\$total은 monthlyxtotal이 됩니다.

2. 중복된 이름은 하나 이상의 숫자를 추가하여 고유한 이름이 됩니다.

예를 들어, `year`의 두 인스턴스는 `year`와 `year1`이 됩니다.

DFHLS2SC 및 DFHLS2WS는 다음 표에 따라 PL/I 데이터 유형을 스키마 요소로 맵핑합니다. 표에 없는 PL/I 유형은 DFHLS2SC 또는 DFHLS2WS에서 지원되지 않습니다. 다음 제한사항도 적용됩니다.

- **COMPLEX** 속성이 있는 데이터 항목은 지원되지 않습니다.
- **FLOAT** 속성이 있는 데이터 항목은 1.2 이상의 맵핑 레벨에서 지원됩니다. 엔터프라이즈 PL/I **FLOAT IEEE**는 지원되지 않습니다.
- **VARYING** 및 **VARYINGZ** 순수 DBCS 문자열은 1.2 이상의 맵핑 레벨에서 지원됩니다.
- **DECIMAL**(p,q)로 지정된 데이터 항목은 $p \geq q$ 인 경우에만 지원됩니다.
- **BINARY**(p,q)로 지정된 데이터 항목은 $q = 0$ 인 경우에만 지원됩니다.
- 데이터 항목에 **PRECISION** 속성이 지정된 경우 무시됩니다.
- **PICTURE** 문자열은 지원되지 않습니다.
- **ORDINAL** 데이터 항목은 **FIXED BINARY(7)** 데이터 유형으로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 **NULL**로 설정된 경우, 문자 배열은 `xsd:string`으로 맵핑되고 널(null) 종료 문자열로 처리됩니다. 엔터프라이즈 PL/I의 경우에는 이 맵핑이 적용되지 않습니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 **BINARY**로 설정된 경우, 문자 배열은 `xsd:base64Binary`로 맵핑되고 2진 데이터로 처리됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 **COLLAPSE**로 설정된 경우, 문자열에 대해 `<xsd:whiteSpace value="collapse"/>`가 생성됩니다.

DFHLS2SC 및 DFHLS2WS는 PL/I의 채우기 알고리즘을 완전히 구현하지 않으므로, 데이터 구조에서 명시적으로 채우기 바이트를 선언해야 합니다. DFHLS2SC 및 DFHLS2WS는 채우기 바이트가 누락되었음을 발견하는 경우 메시지를 발행합니다. 각 최상위 레벨 구조는 double-word 경계에서 시작하며, 구조의 각 바이트가 해당 경계로 맵핑되어야 합니다. 다음 코드 단편을 살펴보겠습니다.

```
3 FIELD1 FIXED BINARY(7),  
3 FIELD2 FIXED BINARY(31),  
3 FIELD3 FIXED BINARY(63);
```

이 예제의 경우:

- **FIELD1**은 1바이트 길이이고, 모든 경계로 할당 가능합니다.
- **FIELD2**는 4바이트 길이이고, 전자(full word) 경계로 할당 가능합니다.
- **FIELD3**은 8바이트 길이이고, 더블 워드(double word) 경계로 할당 가능합니다.

엔터프라이즈 PL/I 컴파일러는 다음과 같은 순서로 필드를 할당합니다.

1. FIELD3이 가장 강력한 경계 요구사항을 가지므로 먼저 할당됩니다.
2. FIELD2는 FIELD3 바로 앞에 전자(fullword) 경계로 할당됩니다.
3. FIELD1은 FIELD3 바로 앞에 바이트 경계로 할당됩니다.

마지막으로, 전체 구조를 전자(fullword) 경계로 할당하기 위해 컴파일러가 FIELD1 바로 앞에 채우기 3바이트를 삽입합니다.

DFHLS2WS는 상응하는 채우기 바이트를 삽입하지 않으므로, DFHLS2WS에서 구조를 처리하기 전에 명시적으로 선언되어야 합니다. 예를 들어, 다음과 같습니다.

```
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);
```

또는 모든 필드를 unaligned로 선언하도록 구조를 변경하고, 해당 구조를 사용하는 애플리케이션을 다시 컴파일할 수 있습니다. PL/I 구조 메모리 할당 요구사항에 대한 자세한 정보는 *Enterprise PL/I Language Reference*를 참조하십시오.

PL/I 데이터 설명	스키마
FIXED BINARY (<i>n</i>) 여기서 $n \leq 7$	<code><xsd:simpleType></code> <code><xsd:restriction base="xsd:byte"/></code> <code></xsd:simpleType></code>
FIXED BINARY (<i>n</i>) 여기서 $8 \leq n \leq 15$	<code><xsd:simpleType></code> <code><xsd:restriction base="xsd:short"/></code> <code></xsd:simpleType></code>
FIXED BINARY (<i>n</i>) 여기서 $16 \leq n \leq 31$	<code><xsd:simpleType></code> <code><xsd:restriction base="xsd:int"/></code> <code></xsd:simpleType></code>
FIXED BINARY (<i>n</i>) 여기서 $32 \leq n \leq 63$ 제한사항: 엔터프라이즈 PL/I만 해당	<code><xsd:simpleType></code> <code><xsd:restriction base="xsd:long"/></code> <code></xsd:simpleType></code>
UNSIGNED FIXED BINARY(<i>n</i>) 여기서 $n \leq 8$ 제한사항: 엔터프라이즈 PL/I만 해당	<code><xsd:simpleType></code> <code><xsd:restriction base="xsd:unsignedByte"/></code> <code></xsd:simpleType></code>
UNSIGNED FIXED BINARY(<i>n</i>) 여기서 $9 \leq n \leq 16$ 제한사항: 엔터프라이즈 PL/I만 해당	<code><xsd:simpleType></code> <code><xsd:restriction base="xsd:unsignedShort"/></code> <code></xsd:simpleType></code>
UNSIGNED FIXED BINARY(<i>n</i>) 여기서 $17 \leq n \leq 32$ 제한사항: 엔터프라이즈 PL/I만 해당	<code><xsd:simpleType></code> <code><xsd:restriction base="xsd:unsignedInt"/></code> <code></xsd:simpleType></code>
UNSIGNED FIXED BINARY(<i>n</i>) 여기서 $33 \leq n \leq 64$ 제한사항: 엔터프라이즈 PL/I만 해당	<code><xsd:simpleType></code> <code><xsd:restriction base="xsd:unsignedLong"/></code> <code></xsd:simpleType></code>

PL/I 데이터 설명	스키마
FIXED DECIMAL(n, m)	<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="n"/> <xsd:fractionDigits value="m"/> </xsd:restriction> </xsd:simpleType></pre>
FIXED DECIMAL(15) DATETIME=PACKED15인 경우 맵핑 레벨 3.0 이상에서 지원됨	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre> 시간소인 형식은 CICS ABSTIME입니다.
BIT(n) 여기서 n 은 8의 배수입니다. 기타 값은 지원되지 않습니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="m"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 $m = n/8$
CHARACTER(n) VARYING 및 VARYINGZ도 맵핑 레벨 1.2 이상에서 지원됩니다. 제한사항: VARYINGZ는 엔터프라이즈 PL/I에 서만 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre>
GRAPHIC(n) VARYING 및 VARYINGZ도 맵핑 레벨 1.2 이상에서 지원됩니다. 제한사항: VARYINGZ는 엔터프라이즈 PL/I에 서만 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="m"/> </xsd:restriction> </xsd:simpleType></pre> 1.0 및 1.1의 맵핑 레벨이며, 여기서 $m = 2*n$ <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> 1.2 이상의 맵핑 레벨
WIDECHAR(n) 제한사항: 엔터프라이즈 PL/I만 해당	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="m"/> </xsd:restriction> </xsd:simpleType></pre> 1.0 및 1.1의 맵핑 레벨이며, 여기서 $m = 2*n$ <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="n"/> </xsd:restriction> </xsd:simpleType></pre> 1.2 이상의 맵핑 레벨

PL/I 데이터 설명	스키마
ORDINAL 제한사항: 엔터프라이즈 PL/I만 해당	<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"/> </xsd:simpleType></pre>
BINARY FLOAT(<i>n</i>) 여기서 $n \leq 21$ 맵핑 레벨 1.2 이상에서 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
BINARY FLOAT(<i>n</i>) 여기서 $21 < n \leq 53$ 53보다 큰 값은 지원되지 않습니다. 맵핑 레벨 1.2 이상에서 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>
DECIMAL FLOAT(<i>n</i>) 여기서 $n \leq 6$ 맵핑 레벨 1.2 이상에서 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
DECIMAL FLOAT(<i>n</i>) 여기서 $6 < n \leq 16$ 16보다 큰 값은 지원되지 않습니다. 맵핑 레벨 1.2 이상에서 지원됩니다.	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

XML 스키마 대 PL/I 맵핑

DFHSC2LS 및 DFHWS2LS 유틸리티 프로그램은 XML 스키마 정의와 PL/I 데이터 구조 간 맵핑을 지원합니다. 엔터프라이즈 PL/I 컴파일러와 이전 PL/I 컴파일러는 다르므로 두 가지 언어 옵션(PLI-ENTERPRISE 및 PLI-OTHER)이 지원됩니다.

CICS 지원은 다음 규칙을 사용하여 스키마 요소 이름에서 PL/I 변수의 고유하고 올바른 이름을 생성합니다.

1. A-Z, a-z, 0-9, @, # 또는 \$ 이외의 문자는 'X'로 대체됩니다.

예를 들어, monthly-total은 monthlyXtotal이 됩니다.

2. 스키마가 변수의 카디널리티가 가변적이고(즉, minOccurs 및 maxOccurs 속성이 xsd:element에서 다른 값으로 지정됨) 스키마 요소 이름이 24자보다 긴 것으로 지정하면 해당 길이에서 잘립니다.

스키마가 변수의 카디널리티가 고정적이고 스키마 요소 이름이 29자보다 긴 것으로 지정하면 해당 길이에서 잘립니다.

3. 동일한 범위의 중복된 이름은 이름의 두 번째 및 후속 인스턴스에 하나 이상의 숫자를 추가하여 고유한 이름이 됩니다.

예를 들어, year의 세 가지 인스턴스는 year, year1, year2가 됩니다.

4. 문자열 _cont 또는 _num에는 다섯 개 문자가 예약되어 스키마가 변수의 카디널리티가 가변적인 것으로 지정하는 경우(즉, minOccurs 및 maxOccurs 속성이 다른 값으로 지정되는 경우) 사용됩니다.

자세한 정보는 132 페이지의 『요소의 가변 배열』을 참조하십시오.

- 속성의 경우 이전 규칙이 요소 이름에 적용됩니다. 접두부 attr-이 요소 이름에 추가되고 그 다음에 -value 또는 -exist가 나타납니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다. 자세한 정보는 137 페이지의 『XML 속성에 대한 지원』을 참조하십시오.

nillable 속성은 특수 규칙을 갖습니다. 접두부 attr-이 추가되지만 요소 이름 처음에 nil-도 추가됩니다. 요소 이름 다음에는 -value가 나타납니다. 총 길이가 28자보다 길면 요소 이름이 잘립니다.

결과 이름의 총 길이는 31자 이하입니다.

DFHSC2LS 및 DFHWS2LS는 다음 표에 따라 스키마 유형을 PL/I 데이터 유형에 맵핑합니다. 다음 사항에 주의하십시오.

- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 NULL로 설정되면 가변 길이 문자 데이터가 널 종료 문자열에 맵핑되고 널 종료자에 추가 문자가 할당됩니다.
- **MAPPING-LEVEL** 매개변수가 1.2 이상으로 설정되고 **CHAR-VARYING** 매개변수가 지정되지 않는 경우에는 기본적으로 가변 길이 문자 데이터가 VARYINGZ 데이터 유형(엔터프라이즈 PL/I의 경우) 및 VARYING 데이터 유형(기타 PL/I의 경우)에 맵핑됩니다.
- 가변 길이 2진 데이터는 VARYING 데이터 유형(32 768바이트 미만인 경우) 및 컨테이너(32 768바이트보다 큰 경우)에 맵핑됩니다.

스키마	PL/I 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 2.0 이하: 지원되지 않음 맵핑 레벨 2.1 이상: 지원됨
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이상:CHAR(255)

스키마	PL/I 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> <xsd:maxLength value="z"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. - string - normalizedString - token - Name - NMTOKEN - language - NCName - ID - IDREF - ENTITY	모든 맵핑 레벨:CHARACTER(z)
<pre><xsd:simpleType> <xsd:restriction base="xsd:type"> </xsd:restriction> </xsd:simpleType></pre> 여기서 <i>type</i>은 다음 중 하나입니다. - duration - date - time - gDay - gMonth - gYear - gMonthDay - gYearMonth	모든 맵핑 레벨:CHAR(32)
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.2 이하: CHAR(32) 맵핑 레벨 2.0 이상: CHAR(40) 맵핑 레벨 3.0 이상: FIXED DECIMAL(15) 시간소인 형식은 CICS ABSTIME입니다.

스키마	PL/I 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value="y"/> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: BIT(z) 여기서 $z = 8 \times y$, $z < 4095$바이트 CHAR(z) 여기서 $z = 8 \times y$, $z > 4095$바이트 맵핑 레벨 1.2 이상: CHAR(y)
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I SIGNED FIXED BINARY (7) 기타 PL/I FIXED BINARY (7)
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I UNSIGNED FIXED BINARY (8) 기타 PL/I FIXED BINARY (8)
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I SIGNED FIXED BINARY (15) 기타 PL/I FIXED BINARY (15)
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I UNSIGNED FIXED BINARY (16) 기타 PL/I FIXED BINARY (16)
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I FIXED DECIMAL(31,0) 기타 PL/I FIXED DECIMAL(15,0)
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	모든 맵핑 레벨: 엔터프라이즈 PL/I SIGNED FIXED BINARY (31) 기타 PL/I FIXED BINARY (31)

스키마	PL/I 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: 엔터프라이즈 PL/I UNSIGNED FIXED BINARY(32) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I CHAR(y) 여기서 y는 16MB 미만의 고정 길이입니다. 모든 맵핑 레벨: 기타 PL/I BIT(64)
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: 엔터프라이즈 PL/I SIGNED FIXED BINARY(63) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I CHAR(y) 여기서 y는 16MB 미만의 고정 길이입니다. 모든 맵핑 레벨: 기타 PL/I BIT(64)
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.1 이하: 엔터프라이즈 PL/I UNSIGNED FIXED BINARY(64) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I CHAR(y) 여기서 y는 16MB 미만의 고정 길이입니다. 모든 맵핑 레벨: 기타 PL/I BIT(64)

스키마	PL/I 데이터 설명
<pre> <xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType> </pre>	맵핑 레벨 1.1 이하: 엔터프라이즈 PL/I SIGNED FIXED BINARY (7) 기타 PL/I FIXED BINARY (7) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I BIT(7) BIT(1) 기타 PL/I BIT(7) BIT(1) 여기서 BIT(7)은 맞추기 용도로 제공되며 BIT(1)은 부울 맵핑 값을 갖습니다.
<pre> <xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="n"/> <xsd:fractionDigits value="m"/> </xsd:restriction> </xsd:simpleType> </pre>	모든 맵핑 레벨:FIXED DECIMAL(n,m)
<pre> <xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType> </pre>	모든 맵핑 레벨:CHAR(255)
<pre> <xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType> </pre>	모든 맵핑 레벨:CHAR(255)

스키마	PL/I 데이터 설명
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value="y"/> </xsd:restriction> </xsd:simpleType></pre> 길이가 정의되지 않는 경우	맵핑 레벨 1.0: 지원되지 않음 맵핑 레벨 1.1: CHAR(<i>z</i>) 여기서 $z = 4 \times (\text{ceil}(y/3))$. $\text{ceil}(x)$은 x보다 크거나 같은 가장 작은 정수입니다. 맵핑 레벨 1.2 이상: CHAR(<i>y</i>) 여기서 길이는 고정적입니다. CHAR(16) 길이가 정의되지 않은 경우. 이 필드는 2진 데이터를 저장하는 컨테이너의 16바이트 이름을 갖습니다.
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.0 및 1.1: CHAR(32) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I DECIMAL FLOAT(6) HEXADEC 기타 PL/I DECIMAL FLOAT(6)
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	맵핑 레벨 1.0 및 1.1: CHAR(32) 맵핑 레벨 1.2 이상: 엔터프라이즈 PL/I DECIMAL FLOAT(16) HEXADEC 기타 PL/I DECIMAL FLOAT(16)

요소의 가변 배열

XML은 요소 수가 가변적인 배열을 포함할 수 있습니다. 일반적으로 요소 수가 일정하지 않은 WSDL 문서와 XML 스키마는 단일 고급 언어 데이터 구조에 효율적으로 맵핑되지 않습니다. CICS는 컨테이너 기반 맵핑 또는 인라인 맵핑을 사용하여 XML에서 수가 일정하지 않은 요소를 처리합니다.

요소 수가 가변적인 배열은 XML 스키마에서 요소 선언에 minOccurs 및 maxOccurs 속성을 사용하여 표시됩니다.

- minOccurs 속성은 요소가 발생할 수 있는 최소 횟수를 지정합니다. 0 이상의 양의 정수 값을 지정할 수 있습니다.
- maxOccurs 속성은 요소가 발생할 수 있는 최대 횟수를 지정합니다. minOccurs 속성 값보다 크거나 같은 양의 정수 값을 지정할 수 있습니다. unbounded 값을 가질 수도 있으며 이는 요소가 발생할 수 있는 횟수에 상한이 적용되지 않음을 나타냅니다.
- 두 속성의 기본값은 모두 1입니다.

이 예제는 선택적인 8바이트 문자열을 보여줍니다. 즉, 애플리케이션 XML 또는 SOAP 메시지에서 나타나지 않거나 한 번 나타날 수 있습니다.

```
<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>
```

다음 예제는 최소 한 번 나타나야 하는 8바이트 문자열을 보여줍니다.

```
<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>
```

일반적으로 요소 수가 일정하지 않은 WSDL 문서는 단일 고급 언어 데이터 구조에 효율적으로 매핑되지 않습니다. 따라서 이러한 사례를 처리하기 위해 CICS는 일련의 컨테이너에서 애플리케이션 프로그램에 전달되는 일련의 연결된 데이터 구조를 사용합니다. 이러한 구조는 애플리케이션의 입력 및 출력으로 사용됩니다.

- CICS가 XML을 애플리케이션 데이터로 변환하는 경우, 애플리케이션 데이터로 해당 구조를 채우고 애플리케이션이 해당 구조를 읽습니다.
- CICS가 애플리케이션 데이터를 XML로 변환하는 경우, 애플리케이션으로 채운 구조에서 애플리케이션 데이터를 읽습니다.

이러한 데이터 구조의 형식은 일련의 예로 가장 정확하게 설명할 수 있습니다. XML은 SOAP 메시지 또는 애플리케이션에서 가져올 수 있습니다. 이 예제는 단순 8바이트 필드의 배열을 사용합니다. 그러나 모델은 복잡한 데이터 유형 배열과 다른 배열을 포함하는 데이터 유형 배열을 지원합니다.

고정 요소 수

첫 번째 예제에서는 정확히 세 번 나타나는 요소를 보여줍니다.

```
<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>
```

이 예제에서는 요소 발생 횟수가 미리 알려져 있으므로 단순 COBOL 선언에서 고정 길이 배열(다른 언어의 경우 이에 상응하는 배열)로 표시될 수 있습니다.

```
05 component PIC X(8) OCCURS 3 TIMES
```

맵핑 레벨 2 이하의 가변 요소 수

이 예제는 한 번에서 다섯 번까지 발생할 수 있는 필수 요소를 나타냅니다.

```
<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>
```

주 데이터 구조에는 두 필드의 선언이 포함됩니다. CICS가 XML을 2진 데이터로 변환하는 경우 첫 번째 필드 component-num에는 요소가 XML에서 나타나는 횟수가 포함되고 두 번째 필드 component-cont에는 컨테이너 이름이 포함됩니다.

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

두 번째 데이터 구조에는 요소 선언 자체가 포함됩니다.

```
01 DFHWS-component
  02 component PIC X(8)
```

component-num(1 - 5 범위의 값을 가짐)의 값을 검사하여 요소 발생 횟수를 확인해야 합니다. 요소 콘텐츠는 component-cont 컨테이너에 있습니다. 이 컨테이너는 각 요소가 DFHWS-component 데이터 구조에 의해 맵핑되는 요소 배열을 포함합니다.

minOccurs="0" 및 maxOccurs="1"인 경우에는 요소가 선택사항입니다. 애플리케이션 프로그램에서 데이터 구조를 처리하려면 component-num 값을 검사해야 합니다.

- 이 값이 0이면 메시지에 컴포넌트 요소가 없고 component-cont의 콘텐츠가 정의되지 않습니다.
- 이 값이 1이면 컴포넌트 요소가 component-cont 컨테이너에 있습니다.

이 컨테이너의 콘텐츠는 DFHWS-component 데이터 구조에 따라 맵핑됩니다.

참고: SOAP 메시지가 단일 반복 요소로 구성되는 경우에는 DFHWS2LS가 두 개 언어 구조를 생성합니다. 주 언어 구조에는 배열의 요소 수와 요소 배열을 포함하는 컨테이너의 이름이 포함됩니다. 두 번째 언어 구조는 반복 요소의 단일 인스턴스를 맵핑합니다.

맵핑 레벨 2.1 이상의 가변 요소 수

맵핑 레벨 2.1 이상의 경우 CICS 지원에서 **INLINE-MAXOCCURS-LIMIT** 매개변수를 사용할 수 있습니다. **INLINE-MAXOCCURS-LIMIT** 매개변수는 가변적인 수의 요소가 처리되는 방식을 지정합니다. 가변적인 수의 요소에 대한 맵핑 옵션은 컨테이너 기반 맵핑(133 페이지의 『맵핑 레벨 2 이하의 가변 요소 수』의 설명 참조) 또는 인라인 맵핑입니다. 이 매개변수의 *value*는 0 - 32767 범위의 양의 정수일 수 있습니다.

- **INLINE-MAXOCCURS-LIMIT**의 기본값은 1이며 이 경우 선택적 요소가 인라인으로 맵핑됩니다.
- **INLINE-MAXOCCURS-LIMIT** 매개변수의 값이 0이면 인라인 맵핑을 방지합니다.
- maxOccurs가 **INLINE-MAXOCCURS-LIMIT** 값 이하인 경우 인라인 맵핑이 사용됩니다.
- maxOccurs가 **INLINE-MAXOCCURS-LIMIT** 값보다 크면 컨테이너 기반 맵핑이 사용됩니다.

가변적인 수의 요소를 인라인으로 맵핑하면 위 고정 발생 예제의 경우와 같이 배열과 카운터가 생성됩니다. component-num 필드는 존재하는 요소 인스턴스 수를 나타내며 해당 인스턴스는 배열로 가리킵니다. 예를 들어, 133 페이지의 『맵핑 레벨 2 이하의 가변 요소 수』처럼 **INLINE-MAXOCCURS-LIMIT**이 5 이하인 경우에는 다음과 같은 데이터 구조가 생성됩니다.

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

첫 번째 필드, component-num은 이전 섹션에 나와 있는 컨테이너 기반 맵핑 예제의 출력과 같습니다. 두 번째 필드에는 생성될 수 있는 최대 요소 수를 포함할 수 있는 길이가 5인 배열이 포함됩니다.

인라인 맵핑은 요소 발생 횟수와 데이터가 배치되는 컨테이너의 이름을 저장하는 컨테이너 기반 맵핑과 다릅니다. 이 맵핑은 현재 컨테이너에 모든 데이터를 저장하기 때문입니다. 데이터를 현재 컨테이너에 저장하면 일반적으로 성능이 향상되므로 인라인 맵핑을 선호하게 됩니다.

중첩된 가변 배열

복잡한 WSDL 문서와 XML 스키마에는 가변 반복 요소가 포함될 수 있으며 이 요소에도 가변 반복 요소가 포함될 수 있습니다. 이러한 경우, 설명되는 구조는 예제에서 설명하는 두 가지 레벨 이상으로 확장됩니다.

이 예제는 필수 요소 <component1>에 중첩되는 선택적 요소 <component2>를 보여 줍니다. 이 경우 필수 요소는 한 번에서 다섯 번까지 발생할 수 있습니다.

```
<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>
```

최상위 레벨 데이터 구조는 이전 예제와 정확하게 일치합니다.

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

그러나 두 번째 데이터 구조에는 다음 요소가 포함됩니다.

```
01 DFHWS-component1
  02 component2-num PIC S9(9) COMP-5
  02 component2-cont PIC X(16)
```

세 번째 구조에는 다음 요소가 포함됩니다.

```
01 DFHWS-component2
  02 component2 PIC X(8)
```

가장 바깥쪽 요소 <component1>의 발생 횟수는 component1-num에 나타납니다.

component1-cont에서 이름이 지정된 컨테이너는 두 번째 데이터 구조 DFHWS-component1의 인스턴스 수를 갖는 배열을 포함합니다.

component2-cont의 각 인스턴스는 다른 컨테이너의 이름을 지정하며 각각 세 번째 레벨 구조 DFHWS-component2가 맵핑하는 데이터 구조를 포함합니다.

이 구조를 나타내려면 예제와 일치하는 XML 단편을 고려하십시오.

```
<component1><component2>string1</component2></component1>
<component1><component2>string2</component2></component1>
<component1></component1>
```

<component1> occurs three times. 처음 두 인스턴스 각각 <component2> 인스턴스를 포함합니다. 세 번째 인스턴스는 해당되지 않습니다.

최상위 레벨 데이터 구조의 경우 component1-num은 값 3을 갖습니다.

component1-cont에서 이름이 지정된 컨테이너는 다음 세 가지 DFHWS-component1 인스턴스를 갖습니다.

1. 첫 번째의 경우 component2-num의 값은 1이고 component2-cont에서 이름이 지정된 컨테이너는 *string1*을 포함합니다.
2. 두 번째의 경우 component2-num의 값은 1이고 component2-cont에서 이름이 지정된 컨테이너는 *string2*를 포함합니다.
3. 세 번째의 경우 component2-num의 값은 0이고 component2-cont의 콘텐츠가 정의되지 않습니다.

이 인스턴스에서 전체 데이터 구조는 모두 네 개의 컨테이너로 나타냅니다.

- 컨테이너 DFHWS-DATA의 루트 데이터 구조
- component1-cont에서 이름이 지정된 컨테이너
- component2-cont의 처음 두 인스턴스에서 이름이 지정된 두 개 컨테이너

선택적 구조와 **xsd:choice**

DFHWS2LS 및 DFHSC2LS는 맵핑 레벨이 2.1 이상이 경우 **<xsd:sequence>**, **<xsd:choice>**, **<xsd:all>** 요소에서 maxOccurs 및 minOccurs 사용을 지원합니다. 여기서 minOccurs 및 maxOccurs 속성은 minOccurs="0" 및 maxOccurs="1"로 설정됩니다.

지원은 이 요소를 각 하위 요소가 선택사항인 것처럼 처리하는 맵핑을 생성합니다. 이러한 요소로 애플리케이션을 구현하는 경우 애플리케이션에 의해 올바르지 않은 옵션 조합이 생성되지 않도록 해야 합니다. 각각의 요소는 생성된 언어 구조에서 고유한 count 필드를 가지며 해당 필드는 모두 "0"으로 설정되거나 모두 "1"로 설정되어야 합니다. **<xsd:choice>** 요소를 제외하고는 다른 값 조합은 모두 올바르지 않습니다.

<xsd:choice> 요소는 요소의 옵션 중 하나만 사용할 수 있음을 나타냅니다. 이 요소는 모든 맵핑 레벨에서 지원됩니다. 지원은 **<xsd:choice>**에서 각 옵션을 minOccurs="0" 및 maxOccurs="1"인 **<xsd:sequence>** 요소에 있는 것처럼 처리합니다. 애플리케이션에 의해 올바르지 않은 조합의 옵션이 생성되지 않도록 하기 위해 **<xsd:choice>** 요소를 사용하여 애플리케이션을 구현하는 경우 유의해야 합니다. 각 요소는 생성된 언어 구조에서 고유한 count 필드를 가집니다. 그 중 필드는 반드시 '1'로 설정되어야 하며 다른 필드는 모두 '0'으로 설정되어야 합니다. 다른 모든 값 조합은 올바르지 않은 조합입니다. 단, **<xsd:choice>** 요소 자체가 선택사항인 경우는 예외이며 이 경우에는 모든 필드가 '0'으로 설정되어야 합니다.

XML 속성에 대한 지원

XML 스키마는 XML에서 필요하거나 허용되는 속성을 지정할 수 있습니다. CICS 지원 유틸리티 DFHWS2LS 및 DFHSC2LS는 기본적으로 XML 속성을 무시합니다. XML 스키마에서 정의되는 XML 속성을 처리하려면 **MAPPING-LEVEL** 매개변수의 값이 1.1 이상이어야 합니다.

선택적 속성

속성은 선택사항 또는 필수사항이며 SOAP 메시지의 요소 또는 애플리케이션의 경우 XML과 연관될 수 있습니다. 스키마에서 정의되는 모든 선택적 속성의 경우 다음과 같이 해당 언어 구조에서 두 개 필드가 생성됩니다.

1. 존재 플래그. 이 필드는 부울 데이터 유형으로 처리되며 일반적으로 길이는 1바이트입니다.
2. 값. 이 필드는 같은 유형의 XML 요소와 동일한 방식으로 매핑됩니다. 예를 들어, NMTOKEN 유형의 속성은 NMTOKEN 유형의 XML 요소와 동일한 방식으로 매핑됩니다.

속성 존재 및 값 필드는 생성된 언어 구조에서 연관된 요소에 대한 필드 앞에 나타납니다. 인스턴스 문서에 나타나는 예상치 못한 속성은 무시됩니다.

예를 들어, 다음 스키마 속성 정의를 고려하십시오.

```
<xsd:attribute name="age" type="xsd:short" use="optional" />
```

이 선택적 속성은 다음 COBOL 구조에 매핑됩니다.

```
05 attr-age-exist PIC X DISPLAY  
05 attr-age-value PIC S9999 COMP-5 SYNC
```

선택적 속성의 런타임 처리

다음 런타임 처리는 선택적 속성에 대해 수행됩니다.

- 속성이 있으면 존재 플래그가 설정되고 값이 매핑됩니다.
- 속성이 없으면 존재 플래그가 설정되지 않습니다.
- 속성이 기본값을 갖고 존재하는 경우에는 값이 매핑됩니다.
- 속성이 기본값을 갖지만 존재하지 않는 경우에는 기본값이 매핑됩니다.

기본값을 갖는 선택적 속성은 필수 속성으로 처리됩니다.

CICS가 데이터를 XML로 변환하는 경우 다음 런타임 처리가 수행됩니다.

- 존재 플래그가 설정된 경우 속성이 변환되고 XML에 포함됩니다.
- 존재 플래그가 설정되지 않은 경우에는 속성이 XML에 포함되지 않습니다.

필수 속성 및 런타임 처리

필요한 모든 속성마다 해당 언어 구조에서 값 필드만 생성됩니다.

속성이 XML에 존재하는 경우에는 값이 매핑됩니다. 속성이 존재하지 않으면 다음과 같이 처리됩니다.

- 애플리케이션이 웹 서비스 제공자인 경우에는 CICS가 클라이언트 SOAP 메시지에서 오류를 나타내는 SOAP 결함 메시지를 생성합니다.

- 애플리케이션이 웹 서비스 요청자인 경우에는 CICS가 메시지를 발행하고 RESP2 코드 13과 함께 변환 오류 응답을 애플리케이션에 리턴합니다.
- 애플리케이션이 **TRANSFORM XMLTODATA** 명령을 사용하는 경우에는 CICS가 메시지를 발행하고 RESP2 코드 3과 함께 올바른지 않은 요청 응답을 애플리케이션에 리턴합니다.

CICS가 COMMAREA 또는 컨테이너의 콘텐츠를 기반으로 하는 SOAP 메시지를 생성하는 경우에는 속성이 변환되어 메시지에 포함됩니다. 애플리케이션이 **TRANSFORM DATATOXML** 명령을 사용하는 경우에는 CICS가 또한 속성을 변환하고 XML에 포함시킵니다.

nillable 속성

nillable 속성은 XML 스키마에서 `xsd:element`에 나타날 수 있는 특수 속성입니다. 이 속성은 `xsi:nil` 속성이 XML의 요소에 올바른 것으로 지정합니다. 요소에 `xsi:nil` 속성이 지정되는 경우 요소는 있지만 값을 갖지 않음을 나타내므로 연관된 콘텐츠가 없습니다.

XML 스키마가 nillable 속성을 true로 정의한 경우에는 부울 값을 갖는 필수 속성으로 맵핑됩니다.

CICS가 SOAP 메시지를 수신하거나 `xsi:nil` 속성을 갖는 애플리케이션에 대해 XML을 변환해야 하는 경우 속성 값은 true 또는 false입니다. 값이 true인 경우에는 애플리케이션이 요소 또는 `xsi:nil` 속성 범위에 중첩된 요소의 값을 무시해야 합니다.

CICS가 `xsi:nil` 속성의 값이 true인 COMMAREA 또는 컨테이너의 콘텐츠를 기반으로 SOAP 메시지 또는 XML을 생성하는 경우에는 다음과 같이 처리됩니다.

- `xsi:nil` 속성은 XML 또는 SOAP 메시지에 생성됩니다.
- 연관된 요소의 값은 무시됩니다.
- 요소 내에 중첩된 요소는 무시됩니다.

SOAP 메시지 예

WSDL 문서의 일부일 수 있는 다음 XML 스키마 예를 고려하십시오.

```
<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>
```

다음은 이 스키마를 준수하는 부분 SOAP 메시지의 예입니다.

```

<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <num xsi:nil="true"/>
  <num>15</num>
  <num xsi:nil="true"/>
</root>

```

COBOL에서는 이 SOAP 메시지가 다음 요소에 맵핑됩니다.

```

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> 및 xsd:anyType에 대한 지원

DFHWS2LS 및 DFHSC2LS는 XML 스키마에서 <xsd:any> 및 xsd:anyType 사용을 지원합니다. <xsd:any> XML 스키마 요소를 사용하여 정의되지 않은 콘텐츠가 있는 XML 문서의 섹션을 설명할 수 있습니다. xsd:anyType은 단순하고 복잡한 모든 데이터 유형이 파생되는 기본 데이터 유형입니다. 데이터 콘텐츠에는 제한사항 또는 제한조건이 없습니다.

CICS 지원과 함께 <xsd:any> 및 xsd:anyType을 사용하려면 먼저 다음 매개변수를 설정하십시오.

- **MAPPING-LEVEL** 매개변수를 2.1 이상으로 설정하십시오.
- 웹 서비스 제공자 애플리케이션의 경우 **PGMINT** 매개변수를 CHANNEL로 설정하십시오.

<xsd:any> 예제

이 예제에서는 <xsd:any> 요소를 사용하여 "Customer" 태그에서 "Surname" 태그 다음에 있는 구조화되지 않은 일부 선택적 XML 콘텐츠를 설명합니다.

```

<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>

```

이 XML 스키마를 준수하는 SOAP 메시지 예는 다음과 같습니다.

```

<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>

```

이 SOAP 메시지가 CICS로 전송되면 CICS가 다음 XML 데이터로 Customer-xml-cont 컨테이너를 채웁니다.

```

<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>

```

CICS는 또한 범위에 포함되는 다음 XML 네임스페이스 선언으로 Customer-xmlns-cont 컨테이너를 채웁니다. 이러한 선언은 공백으로 구분됩니다.

```
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns="http://www.example.org/anyExample"
```

xsd:anyType 예제

*xsd:anyType*은 단순 및 복잡 데이터 유형이 모두 파생되는 기본 데이터 유형입니다. 데이터 콘텐츠를 제한하지 않습니다. 데이터 유형을 지정하지 않는 경우 기본값은 *xsd:anyType*입니다. 예를 들어, 다음 두 XML 단편은 동일합니다.

```

<xsd:element name="Name" type="xsd:anyType"/>
<xsd:element name="Name"/>

```

생성된 언어 구조

<xsd:any> 또는 *xsd:anyType*에 대해 생성된 언어 구조는 COBOL에서 다음 양식을 나타내며 다른 언어의 경우에는 그에 상응하는 양식을 나타냅니다.

elementName-xml-cont PIC X(16)

원시 XML을 포함하는 컨테이너의 이름. CICS가 수신 SOAP 메시지를 처리하는 경우, <xsd:any> 또는 *xsd:anyType*이 정의하는 SOAP 메시지의 서브 세트를 이 컨테이너에 저장합니다. 애플리케이션은 기본적으로 XML 데이터만 처리할 수 있습니다. 애플리케이션은 XML을 생성하고 이 컨테이너를 채워야 하며 컨테이너 이름을 제공해야 합니다.

이 컨테이너는 텍스트 모드에서 채워야 합니다. CICS가 이 컨테이너를 채우는 경우, 사용하도록 정의된 웹 서비스와 동일한 EBCDIC 변형을 사용합니다. 대상 EBCDIC 코드 페이지에 없는 문자는 애플리케이션이 컨테이너를 UTF-8로 읽는 경우라도 대체 문자로 바꿉니다.

elementName-xmlns-cont PIC X(16)

범위에 있는 네임스페이스 접두부 선언을 포함하는 컨테이너의 이름. 이 컨테이너의 콘텐츠는 SOAP 엔벨로프 태그의 서브세트만이 아니라 범위 내에 있고 관련이 있는 모든 네임스페이스 선언을 포함한다는 점을 제외하고는 DFHWS-XMLNS 컨테이너의 콘텐츠와 유사합니다.

이 컨테이너는 텍스트 모드에서 채워야 합니다. CICS가 이 컨테이너를 채우는 경우, 사용하도록 정의된 웹 서비스와 동일한 EBCDIC 변환을 사용합니다. 대 상 EBCDIC 코드 페이지에 없는 문자는 애플리케이션이 컨테이너를 UTF-8로 읽는 경우라도 대체 문자로 바꿉니다.

이 컨테이너는 CICS로 보낸 SOAP 메시지를 처리하는 경우에만 사용됩니다. 애플리케이션이 출력 SOAP 메시지가 생성될 때 컨테이너에 네임스페이스 선언을 제공하려고 시도하는 경우, 컨테이너와 해당 콘텐츠를 CICS가 무시합니다. CICS에서는 애플리케이션이 제공하는 XML이 네임스페이스 선언에 있어 전적으로 자체 포함되어야 합니다.

`<xsd:any>` 요소를 포함하는 XML 요소의 이름은 `<xsd:any>` 요소에 대해 생성되는 변수 이름에 포함됩니다. `<xsd:any>` 예제에서, `<xsd:any>` 요소는 `<xsd:element name="Customer">` 요소 내부에 중첩되고 `<xsd:any>` 요소에 대해 생성되는 변수 이름은 `Customer-xml-cont PIC X(16)` 및 `Customer-xmlns-cont PIC X(16)`입니다.

`xsd:anyType` 유형의 경우에는 직접 XML 요소 이름이 사용됩니다. 위 `xsd:anyType` 예제에서 변수 이름은 `Name-xml-cont PIC X(16)` 및 `Name-xmlns-cont PIC X(16)`입니다.

`<xsd:choice>`에 대한 지원

`<xsd:choice>` 요소는 요소의 옵션 중 하나만 사용할 수 있음을 나타냅니다. CICS 지원은 다양한 맵핑 레벨에서 `<xsd:choice>` 요소에 대한 다양한 수준의 지원을 제공합니다.

맵핑 레벨 2.2 이상의 `<xsd:choice>`에 대한 지원

맵핑 레벨이 2.2 이상인 경우 DFHWS2LS 및 DFHSC2LS는 `<xsd:choice>` 요소에 대한 향상된 지원을 제공합니다. 지원은 `<xsd:choice>` 요소와 연관된 값을 저장하는 새 컨테이너를 생성합니다. 지원은 새 컨테이너의 이름과 추가 필드를 포함하는 언어 구조를 생성합니다.

fieldname-enum

`<xsd:choice>` 요소가 사용하는 옵션을 표시하는 구분 필드

fieldname-cont

사용될 옵션을 저장하는 컨테이너의 이름. 옵션 값을 맵핑하기 위한 추가 언어 구조가 생성됩니다.

다음 XML 스키마 단편에는 <xsd:choice> 요소가 포함됩니다.

```
<xsd:element name="choiceExample">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="option1" type="xsd:string" />
      <xsd:element name="option2" type="xsd:int" />
      <xsd:element name="option3" type="xsd:short" maxOccurs="2" minOccurs="2" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

이 XML 스키마 단편이 맵핑 레벨 2.2 이상에서 처리되는 경우 지원은 다음 COBOL 언어 구조를 생성합니다.

```
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.
```

맵핑 레벨 2.2 이상의 <xsd:choice>에 대한 제한사항

DFHSC2LS 및 DFHWS2LS는 중첩된 <xsd:choice> 요소를 지원하지 않습니다. 예를 들어, 다음 XML은 지원되지 않습니다.

```
<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 및 DFHWS2LS는 반복 <xsd:choice> 요소를 지원하지 않습니다. 예를 들어, 다음 XML은 지원되지 않습니다.

```
<xsd:choice maxOccurs="2">
  <xsd:element name="name1" type="string"/>
</xsd:choice>
```

DFHSC2LS 및 DFHWS2LS는 <xsd:choice> 요소에서 최대 255개 옵션을 지원합니다.

맵핑 레벨 2.1 이하의 <xsd:choice>에 대한 지원

맵핑 레벨 2.1 이하의 경우 DFHWS2LS는 <xsd:choice> 요소에 대한 제한된 지원을 제공합니다. DFHWS2LS는 <xsd:choice> 요소의 각 옵션을 최대한 한 번 발생할 수 있는 <xsd:sequence> 요소인 것처럼 처리합니다.

<xsd:choice> 요소의 옵션 중 하나만 사용할 수 있으므로 <xsd:choice> 요소를 사용하여 애플리케이션을 구현하는 경우 주의해야 합니다. 올바른 옵션 조합만 생성합니다. 각 요소는 생성된 언어 구조에서 고유한 count 필드를 가집니다. 그 중 필드는 반드시 1로 설정되어야 하며 다른 필드는 모두 0으로 설정되어야 합니다. 다른 모든 값 조합은 잘못된 조합입니다. 단, <xsd:choice>가 선택사항인 경우는 예외이며 이 경우에는 모든 필드가 0으로 설정되어야 합니다.

관련 참조:

139 페이지의 『<xsd:any> 및 xsd:anyType에 대한 지원』

DFHWS2LS 및 DFHSC2LS는 XML 스키마에서 <xsd:any> 및 xsd:anyType 사용을 지원합니다. <xsd:any> XML 스키마 요소를 사용하여 정의되지 않은 콘텐츠가 있는 XML 문서의 섹션을 설명할 수 있습니다. xsd:anyType은 단순하고 복잡한 모든 데이터 유형이 파생되는 기본 데이터 유형입니다. 데이터 콘텐츠에는 제한사항 또는 제한조건이 없습니다.

145 페이지의 『추상 요소 및 추상 데이터 유형에 대한 지원』

CICS 지원은 맵핑 레벨 2.2 이상에서 추상 요소 및 추상 데이터 유형에 대한 지원을 제공합니다. CICS 지원은 추상 요소와 추상 데이터 유형을 유사한 방식으로 대체 그룹에 맵핑합니다.

144 페이지의 『대체 그룹에 대한 지원』

대체 그룹을 사용하여 상호 교환할 수 있는 XML 요소의 그룹을 정의할 수 있습니다. CICS 지원은 맵핑 레벨 2.2 이상에서 대체 그룹에 대한 지원을 제공합니다.

대체 그룹에 대한 지원

대체 그룹을 사용하여 상호 교환할 수 있는 XML 요소의 그룹을 정의할 수 있습니다. CICS 지원은 맵핑 레벨 2.2 이상에서 대체 그룹에 대한 지원을 제공합니다.

맵핑 레벨이 2.2 이상인 경우, DFHSC2LS 및 DFHWS2LS는 <xsd:choice> 요소에 사용한 맵핑과 유사한 맵핑을 사용하여 대체 그룹을 지원합니다. 지원은 언어 구조에서 열거 필드와 새 컨테이너 이름을 생성합니다.

다음 XML 스키마 단편에는 각각 replacementOption1 또는 replacementOption2:로 바꿀 수 있는 두 subGroupParent 요소의 배열이 포함됩니다.

```
<xsd:element name="subGroupExample" >
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="subGroupParent" maxOccurs="2" minOccurs="2" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```

    </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" />

```

이 XML 단편을 지원과 함께 처리하면 다음 COBOL 언어 구조가 생성됩니다.

```

03 subGroupExample.
  06 subGroupParent OCCURS2.
    09 subGroupExample-enum PIC X DISPLAY.
      88 empty VALUE X '00'.
      88 replacementOption1 VALUE X '01'.
      88 replacementOption2 VALUE X '02'.
      88 subGroupParent VALUE X '03'.
    09 subGroupExample-cont PIC X (16).

01 Example-replacementOption1.
  03 replacementOption1 PIC S9(9) COMP-5 SYNC.

01 Example-replacementOption2.
  03 replacementOption2 PIC S9999 COMP-5 SYNC.

01 Example-subGroupParent.
  03 subGroupParent-length PIC S9999 COMP-5 SYNC.
  03 subGroupParent PIC X(255).

```

대체 그룹에 대한 자세한 정보는 *W3C XML 스키마 파트 1: 구조 Second Edition 스펙*(http://www.w3.org/TR/xmlschema-1/#Elements_Equivalence_Class)을 참조하십시오.

관련 참조:

139 페이지의 『<xsd:any> 및 xsd:anyType에 대한 지원』

DFHWS2LS 및 DFHSC2LS는 XML 스키마에서 <xsd:any> 및 xsd:anyType 사용을 지원합니다. <xsd:any> XML 스키마 요소를 사용하여 정의되지 않은 콘텐츠가 있는 XML 문서의 섹션을 설명할 수 있습니다. xsd:anyType은 단순하고 복잡한 모든 데이터 유형이 파생되는 기본 데이터 유형입니다. 데이터 콘텐츠에는 제한사항 또는 제한조건이 없습니다.

141 페이지의 『<xsd:choice>에 대한 지원』

<xsd:choice> 요소는 요소의 옵션 중 하나만 사용할 수 있음을 나타냅니다. CICS 지원은 다양한 맵핑 레벨에서 <xsd:choice> 요소에 대한 다양한 수준의 지원을 제공합니다.

145 페이지의 『추상 요소 및 추상 데이터 유형에 대한 지원』

CICS 지원은 맵핑 레벨 2.2 이상에서 추상 요소 및 추상 데이터 유형에 대한 지원을 제공합니다. CICS 지원은 추상 요소와 추상 데이터 유형을 유사한 방식으로 대체 그룹에 맵핑합니다.

추상 요소 및 추상 데이터 유형에 대한 지원

CICS 지원은 맵핑 레벨 2.2 이상에서 추상 요소 및 추상 데이터 유형에 대한 지원을 제공합니다. CICS 지원은 추상 요소와 추상 데이터 유형을 유사한 방식으로 대체 그룹에 맵핑합니다.

맵핑 레벨 2.2 이상의 추상 요소에 대한 지원

맵핑 레벨 2.2 이상의 경우 DFHSC2LS 및 DFHWS2LS는 추상 요소가 그룹의 올바른 멤버가 아니라는 점을 제외하고 대체 그룹과 거의 동일한 방식으로 추상 요소를 처리합니다. 대체 요소가 없는 경우에는 추상 요소가 `<xsd:any>` 요소로 처리되며 맵핑 레벨 2.1의 `<xsd:any>` 요소와 동일한 맵핑을 사용합니다.

다음 XML 스키마 단편은 추상 요소 대신 사용할 수 있는 두 가지 옵션을 지정합니다. 추상 요소 자체는 올바른 옵션이 아닙니다.

```
<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" />
```

이 XML 단편을 지원과 함께 처리하면 다음 COBOL 언어 구조가 생성됩니다.

```
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.
```

추상 요소에 대한 자세한 정보는 *W3C XML 스키마 파트 0: 입문서 Second Edition 스펙*(<http://www.w3.org/TR/xmlschema-0/#SubsGroups>)을 참조하십시오.

맵핑 레벨 2.2 이상의 추상 데이터 유형에 대한 지원

맵핑 레벨 2.2 이상의 경우 DFHSC2LS 및 DFHWS2LS는 추상 데이터 유형을 대체 그룹으로 처리합니다. 지원은 언어 구조에서 열거 필드와 새 컨테이너 이름을 생성합니다.

다음 XML 스키마 단편은 추상 유형 대신 사용할 수 있는 두 가지 대안을 지정합니다.

```
<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>
```

이 XML 단편을 지원과 함께 처리하면 다음 COBOL 언어 구조가 생성됩니다.

```
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).
```

언어 구조는 개별 카피북에 생성됩니다. option1에 대해 생성된 언어 구조는 하나의 카피북에 생성됩니다.

```
03 option1                      PIC X(5).
```

option2에 대한 언어 구조는 다른 카피북에 생성됩니다.

```
03 option2                      PIC X(10).
```

추상 데이터 유형에 대한 자세한 정보는 *W3C XML Schema Part 0: 입문서 Second Edition* 스펙(<http://www.w3.org/TR/xmlschema-0/#SubsGroups>)을 참조하십시오.

관련 참조:

139 페이지의 『<xsd:any> 및 xsd:anyType에 대한 지원』

DFHWS2LS 및 DFHSC2LS는 XML 스키마에서 <xsd:any> 및 xsd:anyType 사용을 지원합니다. <xsd:any> XML 스키마 요소를 사용하여 정의되지 않은 콘텐츠가 있는 XML 문서의 섹션을 설명할 수 있습니다. xsd:anyType은 단순하고 복잡한 모든 데이터 유형이 파생되는 기본 데이터 유형입니다. 데이터 콘텐츠에는 제한사항 또는 제한조건이 없습니다.

141 페이지의 『<xsd:choice>에 대한 지원』

<xsd:choice> 요소는 요소의 옵션 중 하나만 사용할 수 있음을 나타냅니다. CICS 지원은 다양한 맵핑 레벨에서 <xsd:choice> 요소에 대한 다양한 수준의 지원을 제공합니다.

144 페이지의 『대체 그룹에 대한 지원』

대체 그룹을 사용하여 상호 교환할 수 있는 XML 요소의 그룹을 정의할 수 있습니다. CICS 지원은 맵핑 레벨 2.2 이상에서 대체 그룹에 대한 지원을 제공합니다.

COBOL의 가변 반복 콘텐츠 핸들링 방법

COBOL의 경우 포인터 산술식을 사용하여 데이터의 각 인스턴스를 처리함으로써 가변 반복 콘텐츠를 처리할 수 없습니다. 다른 프로그래밍 언어에는 이 제한사항이 없습니다. 이 예는 웹 서비스 애플리케이션에 대해 COBOL로 가변 반복 콘텐츠를 처리하는 방법을 보여줍니다.

이 기술은 **TRANSFORM** API 명령을 사용하여 XML을 애플리케이션 데이터로 변환하는 경우에도 적용됩니다. 다음 예제 WSDL 문서는 가변 횟수만큼 반복되는 8자 문자열로 구성되는 애플리케이션 데이터를 포함하는 웹 서비스를 나타냅니다.

```
<?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>

```

이 WSDL 문서를 DFHWS2LS를 통해 처리하면 다음 COBOL 언어 구조가 생성됩니다.

```

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).

```

8자 component 필드는 별도 구조 DFHWS-component에서 정의됩니다. 기본 데이터 구조는 applicationData이며 두 필드, component-num 및 component-cont를 포함합니다. component-num 필드는 존재하는 component 데이터 인스턴스 수를 나타내고 component-cont 필드는 component 필드의 연결 목록을 포함하는 컨테이너의 이름을 나타냅니다.

다음 COBOL 코드는 가변 반복 데이터 목록을 처리하는 한 가지 방법을 보여줍니다. 이 코드는 연계 섹션 배열을 사용하여 데이터의 후속 인스턴스를 처리합니다. 각 인스턴스는 DISPLAY 문을 사용하여 표시됩니다.

```

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.
```

위 코드는 가변 반복 콘텐츠 핸들링을 위한 일반적인 솔루션을 제공합니다. BIG-ARRAY 배열은 각 컴포넌트의 처음으로 차례로 이동하며 데이터 시작 시 고정된 상태를 유지하

지 않습니다. 컴포넌트 데이터 구조는 다음 컴포넌트의 첫 번째 바이트를 가리키도록 이 동합니다. COMPONENT-PTR은 필요한 경우 컴포넌트 데이터의 시작 위치를 복구하는 데 사용될 수 있습니다.

다음은 WSDL 문서를 준수하는 SOAP 메시지 예입니다.

```
<?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>
```

다음은 SOAP 메시지를 처리할 때 COBOL 프로그램에서 생성되는 출력입니다.

```
CPIH 20080115103151 component value is: VALUE1
CPIH 20080115103151 component value is: VALUE2
CPIH 20080115103151 component value is: VALUE3
```

웹 서비스 지원을 사용하여 웹 서비스 제공자 작성

고급 언어 데이터 구조에서 또는 WSDL 1.1 또는 WSDL 2.0을 준수하는 웹 서비스 설명에서 서비스 제공자 애플리케이션을 작성할 수 있습니다. CICS 웹 서비스 지원은 서비스 제공자 설정에서 CICS 애플리케이션을 배치하는 데 도움을 줍니다.

이 태스크 정보

지원을 사용하여 CICS 애플리케이션을 서비스 제공자로 배치할 때 다음 두 가지 옵션이 있습니다.

- 웹 서비스 설명을 시작하고 지원을 사용하여 언어 데이터 구조를 생성합니다.

기존 웹 서비스 설명에 부합되는 서비스 제공자를 구현할 때에는 이 옵션을 사용하십시오.

- 언어 데이터 구조를 시작하고 지원을 사용하여 웹 서비스 설명을 생성합니다.

기존 프로그램을 웹 서비스로 노출시키고 웹 서비스 설명 및 SOAP 메시지에 프로그램 인터페이스의 측면을 노출시키려는 경우에 이 옵션을 사용하십시오.

URI를 사용하여 서비스 제공자와 연관된 웹 서비스 설명을 노출시킬 수 있습니다. 이 URI는 접미부 ?wsdl이 추가된 WEBSERVICE와 연관된 URI와 같은 경로를 가집니다. 이 경우 비즈니스 내 또는 외부의 요청자가 서비스 제공자와 연관된 WSDL 파일을 검색할 수 있게 됩니다.

웹 서비스 설명에서 서비스 제공자 애플리케이션 작성

CICS 웹 서비스 지원을 사용하여 WSDL 1.1 또는 WSDL 2.0을 준수하는 웹 서비스 설명에서 서비스 제공자 애플리케이션을 작성할 수 있습니다.

시작하기 전에

서비스 제공자 애플리케이션을 작성하려면 먼저 다음 조건이 만족되어야 합니다.

- 웹 서비스 설명이 z/OS의 UNIX 파일에 있어야 하고 CICS 리전에서 적절한 제공자 모드 파이프라인을 작성해야 합니다.
- OMVS에 DFHLS2WS를 실행하는 사용자 ID를 정의해야 합니다.
- 사용자 ID는 z/OS UNIX 및 PDS 라이브러리에 대한 읽기 권한 및 **LOGFILE**, **WSBIND** 및 **WSDL** 매개변수에서 지정되는 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- Java를 실행할 수 있도록 사용자 ID에 충분한 저장영역을 할당해야 합니다. 지원되는 모든 Java 버전을 사용할 수 있습니다. 기본적으로 DFHWS2LS는 **JAVADIR** 매개변수에 지정된 Java 버전을 사용합니다.

이 태스크 정보

웹 서비스 지원을 사용하여 서비스 제공자 애플리케이션의 WSDL에서 언어 구조를 작성할 수 있습니다. IBM WSRR(WebSphere Service Registry and Repository) 서버에 저장되는 WSDL 문서를 사용할 수도 있습니다.

프로시저

1. DFHWS2LS 일괄처리 프로그램을 사용하여 웹 서비스 바인딩 파일 및 한 개 이상의 언어 데이터 구조를 생성하십시오. DFHWS2LS에는 사용자 애플리케이션이 필요로 하는 바인딩 파일 및 언어 구조를 작성할 수 있도록 사용자에게 유연성을 제공하는 대형 선택적 매개변수 세트가 포함되어 있습니다. 웹 서비스에 대해 기존 애플리케이션을 사용으로 설정할 때 이러한 옵션을 고려하십시오.
 - a. 데이터를 서비스 제공자 애플리케이션 프로그램에 전달하기 위해 CICS가 사용하는 메커니즘은?
 - 채널을 사용하여 컨테이너로 데이터를 전달하거나 COMMAREA를 사용할 수 있습니다. 채널 및 컨테이너가 권장됩니다. 이를 **PGMINT** 매개변수로 지정하십시오.
 - b. 생성할 언어는?
 - DFHWS2LS는 COBOL, C/C++ 또는 PL/I 언어 데이터 구조를 생성할 수 있습니다. **LANG** 매개변수를 사용하여 언어를 지정하십시오.
 - c. 사용할 맵핑 레벨은?
 - 맵핑 레벨이 높을수록 런타임 시 문자 및 2진 데이터의 핸들링에 사용 가능한 제어 및 지원이 많아집니다. 일부 선택적 매개변수는 높은 맵핑 레벨에서

만 사용 가능합니다. 사용 가능한 가장 높은 레벨의 맵핑을 사용하도록 권장됩니다. **MAPPING-LEVEL** 매개변수로 맵핑 레벨을 지정하십시오.

d. 웹 서비스 요청자가 사용하게 할 **URI**는?

- **URI** 매개변수를 사용하여 상대 **URI**를 지정하십시오(예: **URI=/my/test/webservice**). 이 값은 **URIMAP** 자원을 작성할 때 **CICS**에서 사용됩니다.

e. 웹 서비스 요청 및 응답을 실행할 트랜잭션 및 사용자 **ID**는?

- 서비스 요청자에 대한 응답을 구성하기 위해 별명 트랜잭션을 사용하여 애플리케이션을 실행할 수 있습니다. 별명 트랜잭션은 사용자 **ID** 아래 첨부됩니다.
- 이를 **TRANSACTION** 및 **USERID** 매개변수로 지정하십시오. 이러한 값은 **URIMAP** 자원을 작성할 때 사용됩니다. 특정 트랜잭션을 사용하지 않으려는 경우에는 이러한 매개변수를 사용하지 마십시오.

f. **WSDL** 문서가 저장되는 위치는?

- 로컬 파일 시스템 대신 **WSRR** 서버에서 **WSDL** 문서를 검색하려면 **DFHWS2LS**에서 특정 매개변수를 지정해야 합니다.
- 최소한 **WSRR** 서버의 위치로 **WSRR-SERVER** 매개변수를 지정하고 **WSRR**에서 검색하려는 **WSDL** 문서의 이름으로 **WSRR-NAME** 매개변수를 지정해야 합니다.
- **WSRR**을 사용하는 경우 지정할 기타 매개변수에 대한 정보는 79 페이지의 『**DFHWS2LS: WSDL**을 고급 언어로 변환』의 내용을 참조하십시오.

g. **WSRR** 서버에서 **WSDL** 문서를 검색하려는 경우 보안 연결을 사용하여 검색하시겠습니까?

- 적절한 매개변수를 설정하여 **SSL(Secure Socket Layer)** 암호화를 사용하면 안전하게 **WSRR**과 상호 운용할 수 있습니다. 예를 들어, 244 페이지의 『웹 서비스 지원 및 **WSRR**에서 **SSL**을 사용하는 방법의 예제』의 내용을 참조하십시오.
- **DFHWS2LS**를 제출할 때 **CICS**는 웹 서비스 바인딩 파일을 생성하고 이 파일을 **WSBIND** 매개변수로 지정한 위치에 배치합니다. 언어 구조는 **PDSLIB** 매개변수로 지정한 파티션된 데이터 세트에 배치됩니다.

2. 생성된 웹 서비스 바인딩 파일을 웹 서비스 애플리케이션에 사용할 제공자 모드 **PIPELINE** 자원의 **pickup** 디렉토리에 복사하십시오. 바인딩 파일은 2진 모드로 복사해야 합니다.
3. 옵션: 웹 서비스 설명 또는 한 개 이상의 웹 서비스 설명을 포함하는 아카이브 파일을 웹 서비스 바인딩 파일과 같은 디렉토리에 복사하십시오. 아카이브 파일은 **.zip** 파일이어야 하며 파일 이름은 **WSDL** 파일 이름과 일치해야 합니다. 이 사본을 사용하여 **WSDL**을 검색할 수 있습니다.

4. 생성된 언어 구조와 인터페이스할 서비스 제공자 애플리케이션 프로그램을 작성하고 필수 비즈니스 로직을 구현하십시오.
5. WEBSERVICE 자원과 두 개의 URIMAP 자원을 작성하십시오.
 - WEBSERVICE 자원은 CICS의 웹 서비스 바인딩 파일을 요약하고 런타임 시 사용됩니다.
 - 첫 번째 URIMAP 자원은 CICS에 WEBSERVICE 자원과 특정 URI를 연관시키기 위한 정보를 제공합니다.
 - 두 번째 URIMAP 자원은 CICS에 WSDL 아카이브 파일 또는 WSDL 문서와 특정 URI를 연관시키기 위한 정보를 제공합니다.
 - 이 URI는 접미부 ?wsdl이 추가된 WEBSERVICE와 연관된 URI와 같은 경로를 가집니다.
 - 이 URIMAP 자원은 외부 요청자가 URI를 사용하여 WSDL 아카이브 파일 또는 WSDL 문서를 검색할 수 있도록 작성됩니다.
 - 웹 서비스 설명 또는 한 개 이상의 웹 서비스 설명을 포함하는 아카이브 파일이 웹 서비스 바인딩 파일과 같은 디렉토리에 복사된 경우에만 이 URIMAP 자원이 작성됩니다.
 - 픽업 디렉토리에 WSDL 아카이브 파일과 WSDL 문서가 포함된 경우 URI는 아카이브 파일에 있는 WSDL만 리턴합니다.
 - 이 기능은 파이프라인 스캔 조작을 사용하여 설치된 웹 서비스에서만 사용 가능합니다.

다음과 같은 방식으로 자원을 작성할 수 있습니다.

- a. **PIPELINE SCAN** 명령을 사용하여 WEBSERVICE 자원 및 URIMAP 자원을 동적으로 작성하십시오.
- b. 직접 자원을 정의합니다. CICS Explorer를 사용하여 CICS 번들에서 WEBSERVICE 자원을 정의하는 경우 웹 서비스 바인딩 파일 및 WSDL 문서 또는 WSDL 아카이브 파일을 가져와서 번들에 포함시킬 수 있습니다. 그러면 웹 서비스 자원을 위한 URIMAP 정의를 생성하고 이러한 정의를 번들로 패키징할 수 있습니다. CICS Explorer를 사용하여 CICS 번들에서 자원을 작성하고 편집하는 작업에 대한 자세한 도움말은 CICS Explorer 제품 문서의 번들에 대한 작업의 내용을 참조하십시오.

결과

DFHWS2LS 제출에 문제점이 있거나 자원이 올바르게 설치되지 않는 경우 703 페이지의 『배치 오류 진단』의 내용을 참조하십시오.

데이터 구조에서 서비스 제공자 애플리케이션 작성

CICS 웹 서비스 지원을 사용하여 고급 언어 데이터 구조에서 서비스 제공자 애플리케이션을 작성할 수 있습니다.

시작하기 전에

서비스 제공자 애플리케이션을 작성하기 전에 다음 전제조건이 완료되었는지 확인하십시오.

- 고급 언어 데이터 구조가 다음 기준을 충족시켜야 합니다.
 - 데이터 구조는 소스 프로그램(예: COBOL 카피북)과 별도로 정의되어야 합니다.
 - PL/I 또는 COBOL 애플리케이션 프로그램이 입력 및 출력에 서로 다른 데이터 구조를 사용하는 경우 데이터 구조는 파티션된 데이터 세트에 있는 두 개의 다른 멤버에서 정의되어야 합니다. 입력 및 출력에 같은 구조를 사용하는 경우 이 구조는 단일 멤버에서 정의되어야 합니다.

C 및 C++에서 데이터 구조는 파티션된 데이터 세트에서 같은 멤버에 있을 수 있습니다.

- 처리하는 데이터 구조는 랩퍼 프로그램을 사용하는지 여부에 따라 다릅니다.
 - 랩퍼 프로그램을 사용하는 경우 카피북은 랩퍼 프로그램에 대한 인터페이스입니다.
 - 랩퍼 프로그램을 사용하지 않는 경우 카피북은 비즈니스 로직에 대한 인터페이스입니다.
- 언어 구조는 파티션된 데이터 세트에서 사용 가능해야 하며 CICS 리전에서 적절한 PIPELINE 자원을 작성해야 합니다.
 - OMVS에 DFHLS2WS를 실행하는 사용자 ID를 정의해야 합니다.
 - 사용자 ID는 z/OS UNIX 및 PDS 라이브러리에 대한 읽기 권한 및 **LOGFILE**, **WSBIND** 및 **WSDL** 매개변수에서 지정되는 디렉토리에 대한 쓰기 권한이 있어야 합니다.
 - 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다. 지원되는 모든 Java 버전을 사용할 수 있습니다. 기본적으로 DFHLS2WS는 **JAVADIR** 매개변수에 지정된 Java 버전을 사용합니다.

프로시저

데이터 구조에서 서비스 제공자 애플리케이션을 작성하려면 다음 단계를 따르십시오.

1. 서비스 제공자 애플리케이션 인터페이스에서 채널 및 다수의 컨테이너를 사용하는 경우 XML로 인터페이스를 설명하는 채널 설명 문서를 작성하십시오. 채널 설명 문서는 z/OS UNIX의 적절한 디렉토리에 넣어야 합니다. CICS는 이 문서를 사용하

여 채널의 여러 컨테이너에서 SOAP 메시지를 구성하고 구성 해제합니다. 또는 한 채널에서 한 개의 컨테이너를 사용하고 채널 설명 문서를 작성하지 않을 수 있습니다.

- 채널 설명 문서를 작성하는 방법에 대한 자세한 정보는 158 페이지의 『채널 설명 문서 작성』의 내용을 참조하십시오.

2. DFHLS2WS 일괄처리 프로그램을 사용하여 언어 구조에서 웹 서비스 설명 및 웹 서비스 바인딩 파일을 생성하십시오. DFHLS2WS에는 사용자 애플리케이션이 필요로 하는 바인딩 파일 및 언어 구조를 작성할 수 있도록 사용자에게 유연성을 제공하는 대형 선택적 매개변수 세트가 포함되어 있습니다. 웹 서비스가 기존 애플리케이션을 사용으로 설정할 때 다음 옵션을 고려하십시오.

a. 데이터를 서비스 제공자 애플리케이션 프로그램에 전달하기 위해 CICS가 사용하는 메커니즘은?

- 채널을 사용하여 컨테이너로 데이터를 전달하거나 COMMAREA를 사용할 수 있습니다. **PGMINT** 매개변수를 사용하여 메커니즘을 지정하십시오. 애플리케이션 인터페이스가 채널 및 다수의 컨테이너를 사용하는 경우 **REQUEST-CHANNEL** 매개변수 및 선택적으로 **RESPONSE-CHANNEL**을 지정하십시오. 이러한 매개변수는 맵핑 레벨이 3.0 이상인 경우에만 사용할 수 있습니다.

b. 생성할 웹 서비스 설명(WSDL 문서)의 레벨은?

- CICS는 WSDL 1.1 또는 WSDL 2.0 문서를 준수하는 설명을 생성합니다. 서비스 제공자 애플리케이션이 두 레벨의 WSDL을 모두 준수하는 요청을 지원하도록 하려는 경우 **WSDL_1.1** 및 **WSDL_2.0** 매개변수에 값을 지정하십시오. 둘 이상의 WSDL 매개변수를 사용할 때 파일 이름이 서로 다른지 확인하십시오. 이 스펙은 두 개의 웹 서비스 설명과 한 개의 바인딩 파일을 생성합니다.

c. 사용할 SOAP 프로토콜의 버전은?

- **SOAPVER** 매개변수로 버전을 지정할 수 있습니다. ALL 값 사용이 권장됩니다. 이 값은 SOAP 1.2 메시지 핸들러를 사용하여 구성되는 파이프라인에 웹 서비스를 설치해야 하는 단점도 있지만 웹 서비스 설명에 대한 바인딩으로 SOAP 1.1 또는 SOAP 1.2를 사용할 수 있는 유연성을 제공합니다. 이 매개변수는 **MINIMUM-RUNTIME-LEVEL**이 2.0 이상인 경우에만 사용할 수 있습니다.

d. 사용할 맵핑 레벨은?

- 맵핑 레벨이 높을수록 런타임 시 문자 및 2진 데이터의 핸들링에 사용 가능한 제어 및 지원이 많아집니다. 일부 선택적 매개변수는 높은 맵핑 레벨에서만 사용 가능합니다. **MAPPING-LEVEL** 매개변수에 사용 가능한 가장 높은 레벨의 맵핑을 지정하도록 권장됩니다.

e. 웹 서비스 요청자가 사용하게 할 URI는?

- **URI** 매개변수를 사용하여 절대 URI를 지정하십시오(예: **URI**=http://www.example.org:80/my/test/webservice). 이 주소의 상대 파트인 /my/test/webservice는 URIMAP 자원을 작성할 때 사용됩니다. 전체 URI는 웹 서비스 설명에서 <soap:address> 요소로 사용됩니다. 이 사용법은 HTTP 및 WebSphere MQ URI 둘 다에 대해 true입니다.
- f. **WSDL 문서를 WSRR(IBM WebSphere Service Registry and Repository)에 공개하시겠습니까?**
- WSDL 문서를 WSRR에 공개하려는 경우 DFHLS2WS에서 **WSRR-SERVER** 매개변수를 지정해야 합니다. WSRR을 사용할 때 지정할 수 있는 매개변수에 대한 자세한 정보는 63 페이지의 『DFHLS2WS: 고급 언어 대 WSDL 변환』의 내용을 참조하십시오.
- g. **WSDL 문서를 WSRR 서버에 공개하려는 경우 보안 연결을 사용하여 공개하시겠습니까?**
- 적절한 매개변수를 설정하여 SSL(Secure Socket Layer) 암호화를 사용하면 안전하게 WSRR과 상호 운용할 수 있습니다. 예를 들어, 244 페이지의 『웹 서비스 지원 및 WSRR에서 SSL을 사용하는 방법의 예제』의 내용을 참조하십시오.
 - DFHLS2WS를 제출할 때 CICS는 웹 서비스 바인딩 파일을 생성하고 이 파일을 **WSBIND** 매개변수로 지정한 위치에 배치합니다. 생성된 웹 서비스 설명은 **WSDL**, **WSDL_1.1** 또는 **WSDL_2.0** 매개변수로 지정한 위치에 배치됩니다.
 - DFHLS2WS에서 WSRR 매개변수를 사용한 경우 WSDL 문서는 사용자가 지정한 WSRR 서버에 공개됩니다.
3. 생성된 웹 서비스 설명을 검토하고 필요한 사용자 정의를 수행하십시오. 자세한 정보는 159 페이지의 『생성된 웹 서비스 설명 문서 사용자 정의』의 내용을 참조하십시오.
4. 웹 서비스 바인딩 파일을 웹 서비스 애플리케이션에 사용할 제공자 모드 파이프라인의 픽업 디렉토리에 복사하십시오. 웹 서비스 바인딩 파일은 2진 모드로 복사해야 합니다.
5. 옵션: 웹 서비스 설명 또는 한 개 이상의 웹 서비스 설명을 포함하는 아카이브 파일을 웹 서비스 바인딩 파일과 같은 디렉토리에 복사하십시오. 아카이브 파일은 .zip 파일이어야 하며 파일 이름은 WSDL 파일 이름과 일치해야 합니다. 이 사본을 사용하여 WSDL을 검색할 수 있습니다.
6. **PIPELINE SCAN** 명령을 사용하여 WEBSERVICE 자원 및 URIMAP 자원을 동적으로 작성하십시오. WEBSERVICE 자원은 CICS의 웹 서비스 바인딩 파일을 요약하고 런타임 시 사용됩니다.
- 첫 번째 URIMAP 자원은 CICS에 WEBSERVICE 자원과 특정 URI를 연관시키기 위한 정보를 제공합니다.

- 두 번째 URIMAP 자원은 CICS에 WSDL 아카이브 파일 또는 WSDL 문서와 특정 URI를 연관시키기 위한 정보를 제공합니다.
 - 이 URI는 접미부 ?wsdl이 추가된 WEBSERVICE와 연관된 URI와 같은 경로를 가집니다.
 - 이 URIMAP 자원은 외부 요청자가 URI를 사용하여 WSDL 아카이브 파일 또는 WSDL 문서를 검색할 수 있도록 작성됩니다.
 - 웹 서비스 설명 또는 한 개 이상의 웹 서비스 설명을 포함하는 아카이브 파일이 웹 서비스 바인딩 파일과 같은 디렉토리에 복사된 경우에만 이 URIMAP 자원이 작성됩니다.
 - 픽업 디렉토리에 WSDL 아카이브 파일과 WSDL 문서가 포함된 경우 URI는 아카이브 파일에 있는 WSDL만 리턴합니다.
 - 이 기능은 파이프라인 스캔 조작을 사용하여 설치된 웹 서비스에서만 사용 가능합니다.

또는 직접 자원을 정의할 수 있지만 권장되지 않습니다.

결과

CICS 자원을 성공적으로 작성한 경우에 서비스 제공자 애플리케이션의 작성이 완료됩니다.

DFHLS2WS 제출에 문제점이 있거나 자원이 올바르게 설치되지 않는 경우 703 페이지의 『배치 오류 진단』의 내용을 참조하십시오.

다음에 수행할 작업

서비스에 액세스할 웹 서비스 요청자를 개발해야 하는 모든 사용자가 사용할 수 있도록 웹 서비스 설명을 공개하십시오.

채널 설명 문서 작성

서비스 제공자 애플리케이션이 여러 컨테이너와의 채널 인터페이스를 사용하는 경우 채널 설명 문서를 작성합니다.

이 태스크 정보

XML 편집기를 사용하여 채널 설명 문서를 작성하십시오. 채널 설명을 위한 스키마는 channel.xsd라고 하고 /usr/lpp/cicsts/cicsts52/schemas/channel 디렉토리(여기서, /usr/lpp/cicsts/cicsts52는 CICS 파일의 기본 설치 디렉토리임)에 있습니다.

프로시저

1. <채널> 요소 및 CICS 채널 네임스페이스를 사용하여 XML 문서를 작성하십시오.

```
<channel name="myChannel" xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```

2. 애플리케이션 프로그램 인터페이스가 채널에서 사용하는 모든 컨테이너에 <컨테이너> 요소를 추가하십시오. 이름, 유형 및 속성을 사용하여 각 컨테이너를 설명해야 합니다. 다음 예제는 서로 다른 속성 값을 가지는 여섯 개의 컨테이너를 보여줍니다.

```
<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>
```

이 구조 요소는 파티션된 데이터 세트 멤버에 있는 언어 구조로 콘텐츠가 정의됨을 표시합니다.

3. z/OS UNIX에서 XML 문서를 저장하십시오.

채널 스키마

채널 설명 문서는 다음 스키마를 준수해야 합니다.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.ibm.com/xmlns/prod/CICS/channel"
  xmlns:tns="http://www.ibm.com/xmlns/prod/CICS/channel" elementFormDefault="qualified">
  <element name="channel"> 1
    <complexType>
      <sequence>
        <element name="container" maxOccurs="unbounded" "unbounded" minOccurs="0"> 2
          <complexType>
            <sequence>
              <element name="structure" minOccurs="0"> 3
                <complexType>
                  <attribute name="location" type="string" use="required"/>
                  <attribute name="structure" type="string" use="optional"/>
                </complexType>
              </element>
            </sequence>
            <attribute name="name" type="tns:name16Type" use="required"/>
            <attribute name="type" type="tns:typeType" use="required"/>
            <attribute name="use" type="tns:useType" use="required"/>
          </complexType>
        </element>
      </sequence>
      <attribute name="name" type="tns:name16Type" use="optional" />
    </complexType>
  </element>
  <simpleType name="name16Type">
    <restriction base="string">
      <maxLength value="16"/>
    </restriction>
  </simpleType>
  <simpleType name="typeType">
    <restriction base="string">
      <enumeration value="char"/>
      <enumeration value="bit"/>
    </restriction>
  </simpleType>
```

```

        </restriction>
    </simpleType>
    <simpleType name="useType">
        <restriction base="string">
            <enumeration value="required"/>
            <enumeration value="optional"/>
        </restriction>
    </simpleType>
</schema>

```

1. 이 요소는 CICS 채널을 나타냅니다.
2. 이 요소는 채널 내 CICS 컨테이너를 나타냅니다.
3. 구조는 '비트' 모드 컨테이너와 함께만 사용할 수 있습니다. '위치' 속성은 컨테이너의 콘텐츠를 맵핑하는 파일의 위치를 표시합니다. '구조' 속성은 C 및 C++에서 사용되어 구조의 이름을 표시할 수 있습니다.

다음에 수행할 작업

DFHLS2WS를 실행하여 웹 서비스 제공자 애플리케이션에 대한 맵핑 및 WSDL 문서를 작성하십시오. DFHLS2WS는 채널 설명 문서에서 컨테이너가 지정되는 순서대로 채널의 맵핑을 WSDL 문서에 넣습니다.

생성된 웹 서비스 설명 문서 사용자 정의

DFHLS2WS에 의해 생성되는 웹 서비스 설명(WSDL) 문서에는 공개 전 변경하기에 적합할 수 있는 일부 자동 생성 콘텐츠가 있습니다. WSDL 문서를 사용자 정의하면 웹 서비스 바인딩 파일이 재생성될 수 있으며 일부 경우에는 랩퍼 프로그램이 작성될 수 있습니다.

이 태스크 정보

다음 단계에 따라 생성된 웹 서비스 설명 문서를 사용자 정의하십시오.

프로시저

1. HTTPS에 대한 지원을 광고하거나 WebSphere MQ를 사용하여 통신하려는 경우 DFHLS2WS에서 **URI** 매개변수를 사용하여 절대 URI를 설정하십시오. **URI** 매개변수를 사용하지 않은 경우 WSDL 문서의 끝에서 <wsdl:service> 및 <wsdl:binding> 요소를 변경해야 합니다. 생성된 WSDL에는 이러한 변경사항 작성을 지원하는 주석이 포함됩니다. 이러한 요소를 변경하더라도 웹 서비스 바인딩 파일을 재생성하지 않아도 됩니다.
2. 웹 서비스의 네트워크 위치를 제공하려면 DFHLS2JS에서 **URI** 매개변수를 사용하여 절대 URI를 설정하십시오. **URI** 매개변수를 사용하지 않은 경우 wsdl:service 요소의 soap:address에 세부사항을 추가하십시오.
 - a. HTTP 기반 프로토콜을 사용하는 경우 *my-server*를 CICS 리전의 TCP/IP 호스트 이름으로 대체하고 *my-port*를 TCPIP SERVICE 자원의 포트 번호로 대체하십시오.

- b. WebSphere MQ를 전송 프로토콜로 사용하는 경우 *myQueue*를 해당 큐의 이름으로 대체하십시오.

웹 서비스 바인딩 파일에 대한 변경 없이도 이러한 변경을 수행할 수 있습니다.

WSBind 파일을 재생성하지 않고 포트 이름 및 네임스페이스를 변경하는 경우 런타임 레벨 2.1 이상에서 모니터링 정보가 틀릴 수 있습니다.

3. WSDL 문서의 자동으로 생성된 이름이 사용자의 목적에 적합한지 여부를 고려해 보십시오. 이러한 값의 이름을 바꿀 수 있습니다.

- WSDL 문서의 targetNamespace
- WSDL 문서 내 XML 스키마의 targetNamespace
- <wsdl:portType> 이름
- <wsdl:operation> 이름
- <wsdl:binding> 이름
- <wsdl:service> 이름
- WSDL 문서 내 XML 스키마의 필드 이름.

이러한 값은 클라이언트 프로그램을 코드화할 프로그래밍 방식 인터페이스 일부를 구성합니다. 생성된 이름의 의미가 분명하지 않으면 장기간에 걸친 애플리케이션 코드의 유지보수가 좀 더 어려울 수 있습니다. DFHLS2WS **REQUEST-NAMESPACE** 및 **RESPONSE-NAMESPACE** 매개변수를 사용하여 XML 스키마의 targetNamespace 및 WSDL 문서의 targetNamespace를 변경하기 위한 **WSDL-NAMESPACE** 매개변수를 변경하십시오.

이러한 값을 변경하는 경우 DFHWS2LS를 사용하여 웹 서비스 바인딩 파일을 재생성해야 합니다. 생성되는 언어 구조는 기존 언어 구조와 같지 않지만 기존 애플리케이션과는 호환 가능하기 때문에 애플리케이션 변경이 필요하지 않습니다. 그러나 새 언어 구조를 무시하고 원래 구조로 새 웹 서비스 바인딩 파일을 사용할 수 있습니다.

4. XML 스키마에서 노출된 COMMAREA 필드가 적합한지 고려하십시오. 웹 서비스 클라이언트 개발자에게 유용하지 않은 필드 제거를 고려할 수 있습니다.

- 출력 값에만 사용되는 필드는 입력 데이터 구조를 맵핑하는 스키마에서 제거할 수 있습니다.
- 필터 필드입니다.
- 자동으로 생성되는 어노테이션입니다.

이러한 변경사항을 작성하는 경우 DFHWS2LS를 사용하여 웹 서비스 바인딩 파일을 재생성해야 합니다. 생성되는 새 언어 구조가 원래 언어 구조와 호환 가능하지 않기 때문에 래퍼 프로그램을 작성하여 새 표시에서 이전 표시로 데이터를 맵핑해

야 합니다. 이 랩퍼 프로그램은 대상 애플리케이션 프로그램에 대해 **EXEC CICS LINK** 명령을 수행한 다음 리턴된 데이터를 맵핑해야 합니다.

이 사용자 정의 레벨에서는 가장 많은 노력이 필요하지만 웹 서비스 클라이언트 개발자에게 가장 유용한 프로그래밍 방식 인터페이스가 생성됩니다.

5. DFHWS2LS를 통해 생성된 WSDL 문서를 넣어 새 언어 구조를 작성하려는 경우 WSDL 문서에 어노테이션을 보관할지 여부를 결정하십시오. 어노테이션은 DFHWS2LS가 언어 구조를 생성할 때 일반 맵핑 규칙을 대체합니다. 맵핑 규칙을 대체할 때 생성된 언어 구조가 DFHLS2WS에서 사용된 버전과 호환 가능한지 확인하십시오. 기본 맵핑 규칙을 사용하여 언어 구조를 생성하려면 어노테이션을 제거하십시오.

결과

사용자 정의한 WSDL 문서를 WSRR(IBM WebSphere Service Registry and Repository) 서버에 공개하려면 WSRR 인터페이스를 사용하여 수동으로 공개해야 합니다. 다음 위치에서 WSRR에 대한 자세한 정보를 찾을 수 있습니다. WebSphere Service Registry and Repository.

예

WSDL 문서의 예제는 생성된 WSDL 문서의 예제를 참조하십시오.

SOAP 결함 전송

서비스 제공자에서 CICS API를 사용하여 웹 서비스 요청자에게 SOAP 결함을 보낼 수 있습니다. 이 결함은 파이프라인에서 서비스 제공자 애플리케이션 또는 헤더 처리 프로그램이 발행할 수 있습니다.

시작하기 전에

API를 사용하려면 서비스 제공자 애플리케이션이 채널 및 컨테이너를 사용해야 합니다. 애플리케이션이 COMMAREA를 사용하는 경우 SOAP 결함 메시지를 작성하는 데 채널 및 컨테이너를 사용하는 랩퍼 프로그램을 작성하십시오. 헤더 처리 프로그램의 API는 CICS 제공 SOAP 메시지 핸들러가 직접 호출하는 경우에만 사용할 수 있습니다.

이 태스크 정보

애플리케이션 로직이 요청을 충족시킬 수 없는 경우(예: 요청 메시지에 기본적인 문제점이 있는 경우) 웹 서비스 요청자에게 SOAP 결함을 발행하려고 할 수 있습니다. CICS에서는 SOAP 결함 발행을 오류로 고려하지 않기 때문에 오류 처리가 아닌 정상적인 메시지 응답 파이프라인 처리가 발생할 수 있음에 유의하십시오. 트랜잭션을 롤백하지 않으려면 애플리케이션 프로그램을 사용해야 합니다.

프로시저

1. 프로그램에서 **EXEC CICS SOAPFAULT CREATE** 명령을 사용하여 SOAP 결함을 전송하십시오(SOAPFAULT CREATE 참조).
2. 명령에 **CLIENT** 또는 **SERVER** 옵션을 추가하십시오. 이 옵션은 문제점이 발생한 위치가 클라이언트 측인지 또는 서버 측인지 표시합니다.
 - **CLIENT**는 문제점이 수신한 요청 메시지에 있음을 표시합니다.
 - **SERVER**는 요청 메시지가 CICS에서 처리될 때 문제점이 발생함을 표시합니다. 이 문제점은 애플리케이션 프로그램에 있을 수 있습니다. 예를 들어, 요청을 충족시킬 수 없거나 파이프라인 처리 중 발생하는 기본적인 문제점이 될 수 있습니다.
3. 서비스 제공자가 결함을 발행한 이유의 요약을 제공하는 **FAULTSTRING** 옵션 및 **FAULTSTRLEN** 옵션에 지정된 그 길이를 추가하십시오. 이 옵션의 콘텐츠는 XML로 되어 있습니다. 애플리케이션이 제공하는 모든 데이터는 XML 문서에 직접 포함시키기에 적합한 형식이어야 합니다. 애플리케이션은 일부 문자를 XML 엔티티로 지정해야 할 수도 있습니다. 예를 들어, < 문자가 XML 태그 시작부가 아닌 다른 위치에 사용되면 애플리케이션은 이 문자를 <로 변경해야 합니다. 다음 예제는 잘못된 **FAULTSTRING** 옵션을 보여줍니다.

```
dc1 msg_faultString char(*) constant('Error: Value A < Value B');
```

이 **FAULTSTRING** 옵션을 지정하는 올바른 방식은 다음과 같습니다.

```
dc1 msg_faultString char(*) constant('Error: Value A &lt; Value B');
```

팁: XML 엔티티 사용을 피하기 위해 XML CDATA 구성으로 데이터를 래퍼할 수 있습니다. XML 프로세서는 이 구성의 문자 데이터를 구문 분석하지 않습니다. 이 메소드를 사용하여 다음 **FAULTSTRING** 옵션을 지정할 수 있습니다.

```
dc1 msg_faultString char(*) constant('<![CDATA[Error: Value A < Value B]]>');
```

4. 서비스 제공자가 결함을 발행한 이유의 세부사항을 제공하는 **DETAIL** 옵션 및 **DETAILLENGTH** 옵션에 지정된 그 길이를 코드화하십시오. 이 옵션의 콘텐츠는 XML로 되어 있습니다. **FAULTSTRING** 옵션에 대한 지침과 동일한 지침이 **DETAIL** 옵션에 적용됩니다.
5. 옵션: 헤더 처리 프로그램에서 API를 호출하는 경우 파이프라인 구성 파일에서 프로그램을 정의하십시오. 헤더 처리 프로그램은 `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` 또는 `<cics_soap_1.2_handler_java>` 요소에서 정의됩니다.

결과

프로그램이 이 명령을 발행하는 경우 CICS는 적절한 SOAP 레벨에서 SOAP 결함 응답 메시지를 작성합니다. 서비스 제공자 애플리케이션이 이 명령을 발행하는 경우 SOAP

응답을 작성하고 DFHRESPONSE 컨테이너에 넣지 않아도 됩니다. 파이프라인은 메시지 핸들러를 통해 SOAP 결함을 처리하고 웹 서비스 제공자에게 응답을 전송합니다.

예제

SOAPFAULT CREATE 명령에는 웹 서비스 요청자에게 적절하게 응답할 수 있도록 유연성을 제공하는 여러 옵션이 있습니다. 다음은 SOAP 1.1 및 SOAP 1.2 둘 다에 사용할 수 있는 SOAP 결함을 작성하는 완료된 명령의 단순 예제입니다.

```
EXEC CICS SOAPFAULT CREATE CLIENT
      DETAIL(msg_detail)
      DETAILLENGTH(length(msg_detail))
      FAULTSTRING(msg_faultString)
      FAULTSTRLEN(length(msg_faultString));
```

다음 값을 사용하여 *msg_detail* 및 *msg_faultString*을 코드화할 수 있습니다.

```
dcl 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>');
dcl msg_faultString char(*) constant('Something went wrong');
```

웹 서비스 지원을 사용하여 웹 서비스 요청자 작성

WSDL 1.1 또는 WSDL 2.0을 준수하는 웹 서비스 설명에서 서비스 요청자 애플리케이션을 작성할 수 있습니다. CICS 웹 서비스 지원은 서비스 요청자 설정에서 CICS 애플리케이션을 배치하는 데 도움을 줍니다.

시작하기 전에

웹 서비스 설명이 z/OS UNIX의 파일에 있거나 IBM WSRR(WebSphere Services Registry and Repository) 서버에 공개되어야 하고 요청자 모드 파이프라인이 CICS 리전에 설치되어야 합니다.

Java를 실행할 수 있도록 사용자 ID에 충분한 저장영역을 할당해야 합니다. 지원되는 모든 Java 버전을 사용할 수 있습니다. 기본적으로 DFHWS2LS는 **JAVADIR** 매개변수에 지정된 Java 버전을 사용합니다.

이 태스크 정보

CICS 웹 서비스 지원을 사용하여 CICS 애플리케이션을 서비스 요청자로 배치할 때 웹 서비스 설명을 시작하고 이 설명으로부터 언어 데이터 구조를 생성해야 합니다.

프로시저

1. DFHWS2LS 일괄처리 프로그램을 사용하여 웹 서비스 바인딩 파일 및 한 개 이상의 언어 구조를 생성하십시오. 웹 서비스 설명에서 서비스 요청자 애플리케이션을 작성할 때 다음 옵션을 고려하십시오.

- 사용할 맵핑 레벨은? 맵핑 레벨이 높을수록 런타임 시 문자 및 2진 데이터의 핸들링에 사용 가능한 제어 및 지원이 많아집니다. 일부 선택적 매개변수는 높은 맵핑 레벨에서만 사용 가능합니다. 사용 가능한 가장 높은 레벨의 맵핑을 사용하도록 권장됩니다.
- 런타임 시 데이터를 변환할 때 사용할 코드 페이지는? CICS 리전의 코드 페이지와는 다른 특정 코드 페이지를 애플리케이션에 사용하려는 경우에 **CCSID** 매개변수를 사용하십시오.
- 웹 서비스 설명에서 선언된 조작의 서브세트를 지원하시겠습니까? 매우 큰 웹 서비스 설명을 가지고 있고 서비스 요청자 애플리케이션이 특정 수의 조작을 지원하게 하려는 경우 **OPERATION** 매개변수를 사용하여 원하는 조작을 나열하십시오. 각 조작은 공백으로 구분되어야 하고 대소문자를 구분합니다.
- WSDL 문서가 저장되는 위치는? DFHWS2LS에 대한 입력으로 사용할 WSDL 문서가 WSRR 서버에 저장되는 경우 특정 매개변수가 지정된 DFHWS2LS를 실행하여 해당 문서를 검색할 수 있습니다. **WSRR-SERVER** 매개변수를 사용하여 WSRR 서버의 위치를 지정하고 **WSRR-NAME** 매개변수를 사용하여 검색할 WSDL 문서의 이름을 지정하십시오. WSRR과 상호작용하기 위해 사용할 수 있는 DFHWS2LS의 기타 매개변수에 대한 정보는 79 페이지의 『DFHWS2LS: WSDL을 고급 언어로 변환』의 내용을 참조하십시오.
- WSRR 서버에서 WSDL 문서를 검색하려는 경우 보안 연결을 사용하여 검색하시겠습니까? 웹 서비스 지원에서 SSL(Secure Socket Layer) 암호화를 사용하여 안전하게 WSRR과 상호 운용할 수 있습니다. 예를 들어, 244 페이지의 『웹 서비스 지원 및 WSRR에서 SSL을 사용하는 방법의 예제』의 내용을 참조하십시오.

DFHWS2LS를 사용할 때 **PROGRAM**, **URI**, **TRANSACTION** 및 **USERID**와 같은 매개변수를 지정하지 마십시오. 이러한 매개변수는 서비스 제공자 애플리케이션에만 적용되며 이러한 매개변수를 지정하면 제공자 모드 웹 서비스 바인딩 파일이 생성되는 원인이 됩니다. 웹 서비스 바인딩 파일 외에도 프로그램은 언어 데이터 구조를 생성합니다.

2. 로그 파일을 확인하여 DFHWS2LS이 바인딩 파일 및 언어 구조를 생성하는 동안 문제점이 발생했는지 여부를 확인하십시오. CICS는 웹 서비스 설명의 일부 요소 또는 옵션을 지원하지 않을 수 있습니다. 경고 또는 오류 메시지가 발생된 경우 제공되는 조언을 읽고 적절한 조치를 취하십시오. 일괄처리 프로그램을 재실행해야 할 수도 있습니다.
3. 웹 서비스 바인딩 파일을 웹 서비스 애플리케이션에 사용할 요청자 모드 파이프라인의 픽업 디렉토리에 복사하십시오.
4. PIPELINE 자원이 서비스 요청자 애플리케이션에 대해 구성되는지 확인하십시오. **MODE** 매개변수의 값은 파이프라인이 요청자 웹 서비스 애플리케이션을 지원하는지 또는 제공자 웹 서비스 애플리케이션을 지원하는지 표시합니다.

5. 웹 서비스의 파이프라인에서 정확한 SOAP 프로토콜이 지원되는지 확인하십시오. **SOAPLEVEL** 매개변수는 지원되는 버전을 표시합니다. 서비스 요청자 모드에서 웹 서비스의 바인딩은 파이프라인에서 지원되는 SOAP의 버전과 일치해야 합니다. SOAP 1.2를 지원하는 서비스 요청자 파이프라인에 SOAP 1.1 바인딩을 사용하여 웹 서비스를 설치할 수 없습니다.
6. 파이프라인에 대해 구성된 제한시간이 서비스 요청자 애플리케이션에 대해 적절한지 확인하십시오. 제한시간은 **PIPELINE** 자원에서 **RESPWAIT** 속성의 값으로 표시됩니다. 파이프라인에서 제한시간이 지정되지 않은 경우에는 전송 기본값이 사용됩니다.
 - HTTP에 대한 기본 제한시간은 10초입니다.
 - WebSphere MQ에 대한 기본 제한시간은 60초입니다.

CICS 리전의 각 트랜잭션에는 디스패처 제한시간이 있습니다. 이 값이 두 프로토콜 중 하나의 기본값보다 적으면 제한시간 초과가 디스패처와 함께 발생합니다.
7. 옵션: 런타임 시 웹 서비스에 대한 유효성 검증을 켤 수 있도록 웹 서비스 바인딩 파일과 같은 픽업 디렉토리에 웹 서비스 설명을 복사하십시오.
8. **WEBSERVICE** 자원을 작성하십시오. 이 자원은 CICS의 웹 서비스 바인딩 파일을 포함하며 런타임 시 사용됩니다.

다음과 같은 방식으로 이를 수행할 수 있습니다.

- a. **PIPELINE SCAN** 명령을 사용하여 **WEBSERVICE** 자원을 동적으로 작성하십시오.
 - b. 직접 자원을 정의합니다. CICS Explorer를 사용하여 CICS 번들에서 **WEBSERVICE** 자원을 정의하는 경우 웹 서비스 바인딩 파일 및 WSDL 문서 또는 WSDL 아카이브 파일을 가져와서 번들에 포함시킬 수 있습니다. 그러면 웹 서비스 지원을 위한 **URIMAP** 정의를 생성하고 이러한 정의를 번들로 패키징할 수 있습니다. CICS Explorer를 사용하여 CICS 번들에서 자원을 작성하고 편집하는 작업에 대한 자세한 도움말은 CICS Explorer 제품 문서의 번들에 대한 작업의 내용을 참조하십시오.
9. 통신 로직을 대체할 수 있는 랩퍼 프로그램을 작성하십시오. 1단계에서 생성된 언어 데이터 구조를 사용하여 랩퍼 프로그램을 작성하십시오. 랩퍼 프로그램에서 **EXEC CICS INVOKE SERVICE** 명령을 사용하여 웹 서비스와 통신하십시오. 이 명령에는 다음 옵션이 포함됩니다.
 - 웹 서비스의 URI
 - 웹 서비스가 호출되는 조작

웹 서비스를 호출할 때 웹 서비스의 URI에 대한 정보를 포함하는 **URIMAP** 자원을 지정할 수 있습니다. **URIMAP** 자원을 사용하는 대신 **INVOKE SERVICE** 명령에서 직접 이 정보를 지정할 수 있습니다. 그러나 **URIMAP** 자원의 사용은 서비스 제공자의 URI가 변경되는 경우 사용자 애플리케이션을 다시 컴파일하지 않아도

됨을 의미합니다. 후속 요청을 위해 애플리케이션이 재사용하거나 같은 서비스를 호출하는 다른 애플리케이션이 재사용할 수 있도록 URIMAP 자원을 통해 CICS가 클라이언트 연결 사용 후 이 연결을 열린 상태로 유지하는 연결 풀 구현을 선택할 수도 있습니다. PIPELINE SCAN 명령은 서비스 요청자의 URIMAP 자원을 작성하지 않으므로 애플리케이션 개발에서 HTTP 클라이언트로 CICS에 대한 URIMAP 자원 작성의 지시사항에 따라 URIMAP 자원을 직접 정의해야 합니다.

결과

CICS 자원을 성공적으로 작성한 경우에 서비스 요청자 애플리케이션의 작성이 완료됩니다.

PIPELINE 자원의 구성 확인

다음 인터페이스로 PIPELINE의 구성을 확인할 수 있습니다.

CICS Explorer



CICS Explorer 관리 보기

파이프라인 보기를 사용하십시오.

CICSplex SM

PIPELINE 정의 보기

CEMT



INQUIRE PIPELINE 명령

CICS SPI



INQUIRE PIPELINE 명령

도구를 사용하여 웹 서비스 작성

웹 서비스 지원 JCL을 사용하는 대신 z Systems용 Rational Developer을 사용하거나 자체 Java 프로그램을 작성하여 CICS에서 필수 파일을 작성할 수 있습니다.

프로시저

1. 두 가지 선택사항이 있습니다.

- z Systems용 Rational Developer 도구를 사용하여 웹 서비스 바인딩 파일 및 웹 서비스 설명 또는 언어 구조를 작성합니다. 이 도구에 대한 자세한 정보는 Rational Developer for System z의 내용을 참조하십시오.

- 제공된 API를 사용하여 웹 서비스 지원을 호출하기 위한 자체 Java 프로그램을 작성합니다. 이 API는 웹 서비스 지원: 클래스 참조 Javadoc에 설명되어 있습니다. 여기에는 클래스를 설명하는 주석이 포함되고 웹 서비스 지원을 호출하는 방식에 대한 예제를 제공하기 위해 샘플 코드가 제공됩니다. Javadoc에는 필수인 JAR 파일 및 z/OS UNIX에서의 그 위치에 대한 전체 목록도 포함됩니다.

z/OS, Windows 또는 Linux 플랫폼에서 Java 프로그램을 실행할 수 있습니다. Windows 또는 Linux에서 프로그램을 실행하는 경우 FTP 또는 동등한 프로세스를 사용하여 생성된 웹 서비스 바인딩 파일을 2진 모드로 적절한 픽업 디렉토리에 전송하십시오.

2. 옵션: 언어 구조에서 웹 서비스 설명을 생성하는 경우 파일을 검토하고 필요한 사용자 정의를 수행하십시오. 자세한 정보는 159 페이지의 『생성된 웹 서비스 설명 문서 사용자 정의』의 내용을 참조하십시오.
3. 생성된 웹 서비스 바인딩 파일을 적절한 파이프라인 픽업 디렉토리에 배치하십시오.
4. 옵션: 예상대로 작동하는지 확인하기 위해 웹 서비스의 유효성 검증을 수행할 수 있도록 웹 서비스 설명을 파이프라인의 픽업 디렉토리에 복사하십시오.
5. 웹 서비스 설명을 시작한 경우 생성된 언어 구조와 인터페이스할 서비스 제공자 또는 서비스 요청자 애플리케이션 프로그램을 작성하십시오.
6. 필수 CICS 자원이 자동으로 작성되게 하려면 **PIPELINE SCAN** 명령을 실행하십시오.

자체 XML 인식 웹 서비스 애플리케이션 작성

CICS 제공 데이터 맵핑을 사용하지 않기로 선택한 경우 대신 두 가지 방식으로 자체 XML 인식 데이터 애플리케이션을 작성할 수 있습니다. DFHWS2LS에서 **XML-ONLY** 매개변수를 사용하거나 어떠한 도구도 사용하지 않고 자체 애플리케이션을 작성할 수 있습니다. **XML-ONLY** 매개변수를 사용하는 것은 처리할 XML 데이터를 애플리케이션에 전달하도록 CICS 파이프라인 프로세스를 구성하는 가장 간단한 방식입니다.

이 태스크 정보

자체 XML 인식 애플리케이션을 작성하는 작업에는 XML 문서를 구문 분석하고 생성하기 위한 코드 작성이 포함됩니다. 자체 XML 인식 애플리케이션을 작성하는 한 가지 방식은 COBOL에서 XML PARSE 및 XML GENERATE 명령문을 사용하는 것입니다. 자체 XML 인식 애플리케이션을 작성하는 다른 방식은 다른 IBM 도구를 사용하는 것입니다. 예를 들어, 애플리케이션에서 호출할 수 있는 COBOL XML 변환기 프로그램을 생성하기 위한 도구로 z Systems용 Rational Developer을 사용할 수 있습니다.

XML 인식 서비스 제공자 애플리케이션 작성

XML 인식 서비스 제공자 애플리케이션은 이 애플리케이션에 전달되는 컨테이너에서 작동하고 XML과 프로그램 언어 간 데이터 변환을 처리해야 합니다.

이 태스크 정보

다음 단계는 CICS 도구의 사용에 대한 의사결정을 포함하여 XML 인식 애플리케이션의 작성 과정을 안내합니다.

프로시저

1. DFHWS2LS를 사용하여 XML 인식 애플리케이션의 웹 서비스 바인딩 파일을 생성할지 여부를 결정하십시오. 웹 서비스 바인딩 파일 생성의 장점은 유효성 검증과 같은 CICS 서비스를 사용하여 웹 서비스 및 글로벌 사용자 종료를 사용하는 CICS 모니터링을 테스트할 수 있다는 것입니다.
 - 웹 서비스 바인딩 파일을 생성하려면 **XML-ONLY** 매개변수를 지정하고 2.1 이상의 **MINIMUM-RUNTIME-LEVEL**을 지정하여 DFHWS2LS를 실행하십시오. 웹 서비스 바인딩 파일을 사용하면 애플리케이션 프로그램이 DFHWS-BODY 컨테이너의 콘텐츠에 대해 직접 작업할 수 있습니다. 모든 면에서 생성된 바인딩 파일은 **XML-ONLY** 매개변수 없이 생성된 파일과 같은 배치 특성 및 같은 런타임 동작(예: SOAP 메시지 핸들링 중 XML의 구문 분석)을 공유합니다. 이러한 구문 분석을 방지하려면 파이프라인 구성 파일에서 SOAP 메시지 핸들러를 지정하지 말아야 합니다.
 - 웹 서비스 바인딩 파일을 사용하지 않으려면 웹 서비스 요청이 XML 인식 애플리케이션에 도달하도록 서비스 제공자 파이프라인을 구성하십시오. 파이프라인 구성 파일에서 터미널 핸들러를 구성하여 XML 인식 애플리케이션 프로그램을 사용하거나 파이프라인에서 수신되는 URI에 따라 애플리케이션을 동적으로 전환하는 메시지 핸들러를 작성할 수 있습니다.
2. 다음 컨테이너에 보관되는 웹 서비스 요청을 처리할 애플리케이션을 작성하십시오.

DFHWS-BODY

파이프라인에 CICS 제공 SOAP 메시지 핸들러가 포함된 경우 인바운드 SOAP 요청에 대한 SOAP 본문의 콘텐츠입니다.

DFHREQUEST

파이프라인에서 수신한 SOAP 요청의 엔벨로프를 포함하는 전체 요청입니다.

DFHWS-XMLNS

네임스페이스 접두부를 요청의 XML 콘텐츠에 대한 네임스페이스로 매핑하는 이름 값 쌍의 목록입니다.

DFHWS-SOAPACTION

컨테이너 DFHWS-BODY에서 SOAP 메시지와 연관된 SOAPAction 헤더입니다.

컨테이너에 대한 작업을 위해 API 명령을 코드화할 때 CHANNEL 옵션을 지정하지 마십시오. 모든 컨테이너가 현재 채널(프로그램에 전달된 채널)과 연관되기 때문입니다. 채널의 이름을 알아야 하는 경우, **EXEC CICS ASSIGN CHANNEL** 명령을 사용하십시오.

3. 옵션: 애플리케이션은 메시지 핸들러가 처리의 일부로 작성하는 다른 컨테이너 외에도 파이프라인에서 메시지 핸들러에 사용 가능한 추가 컨테이너를 사용할 수도 있습니다. 컨테이너의 전체 목록은 345 페이지의 『파이프라인에서 사용되는 컨테이너』의 내용을 참조하십시오.
4. 애플리케이션이 요청 처리를 완료한 후에는 다음 컨테이너를 사용하여 웹 서비스 응답을 구성하십시오.

DFHRESPONSE

파이프라인에 전달될 전체 응답 메시지입니다. 메시지에 SOAP을 사용하지 않거나 프로그램에서 엔벨로프를 포함하는 전체 SOAP 메시지를 빌드하려는 경우에 CICS 제공 SOAP 메시지 핸들러를 사용하는 대신 이 컨테이너를 사용하십시오.

컨테이너 DFHWS-BODY에 SOAP 본문을 제공하면 DFHRESPONSE는 무시됩니다.

DFHWS-BODY

아웃바운드 SOAP 응답에서 SOAP 본문의 콘텐츠입니다. 파이프라인의 터미널 핸들러가 CICS 제공 SOAP 메시지 핸들러인 경우 이 컨테이너를 제공하십시오. 메시지 핸들러는 본문을 포함한 전체 SOAP 메시지를 구성합니다.

요청 및 응답이 동일하더라도 프로그램은 이 컨테이너를 작성해야 합니다. 이 컨테이너를 작성하지 않으면 CICS가 내부 서버 오류를 발행합니다.

다른 컨테이너를 사용하여 아웃바운드 응답을 처리하기 위해 파이프라인이 필요로 하는 정보를 전달할 수도 있습니다.

웹 서비스가 응답을 리턴하지 않는 경우 컨테이너 DFHNORESPONSE를 리턴하여 응답이 없음을 표시해야 합니다. 메시지 핸들러는 컨테이너가 있는지 여부만 확인하기 때문에 컨테이너의 콘텐츠는 중요하지 않습니다.

5. URIMAP 자원을 작성하십시오. **XML-ONLY** 매개변수를 사용하고 있고 DFHWS2LS의 **URI** 매개변수에 값을 지정한 경우 PIPELINE SCAN 프로세스 중에 URIMAP이 자동으로 작성됩니다.

XML 인식 서비스 요청자 애플리케이션 작성

XML 인식 웹 서비스 요청자 애플리케이션은 XML과 프로그래밍 언어 사이의 데이터 변환을 처리하고 파이프라인에서 제어 컨테이너를 채웁니다.

시작하기 전에

DFHWS2LS에서 **XML-ONLY** 매개변수를 사용하거나 어떠한 도구도 사용하지 않고 자체 XML 인식 서비스 요청자 애플리케이션을 작성할 수 있습니다. 자체 XML 인식 서비스 요청자 애플리케이션을 작성하는 가장 단순한 방법은 DFHWS2LS에서 **XML-ONLY** 매개변수를 사용하여 작성하는 것입니다. **XML-ONLY** 매개변수는 런타임 레벨 2.1 이상에서 사용 가능합니다.

이 태스크 정보

XML-ONLY 매개변수를 사용하면 CICS에 애플리케이션이 DFHWS-BODY 컨테이너의 콘텐츠를 사용하여 직접 작업하도록 지시하는 WSBInd 파일이 생성됩니다. 생성된 WSBInd 파일은 요청자 모드 PIPELINE에 설치되어 요청자 모드 WEBSERVICE 자원을 작성해야 합니다. 애플리케이션은 웹 서비스 요청의 본문에 대해 XML을 생성하고 생성된 XML을 DFHWS-BODY 컨테이너에 저장해야 합니다. 그런 다음 **EXEC CICS INVOKE SERVICE** 명령을 호출해야 합니다. 웹 서비스 제공자에게 아웃바운드 메시지가 전송됩니다. 호출 완료 후 응답 메시지의 본문은 DFHWS-BODY 컨테이너에도 있습니다.

SOAP 메시지 핸들링 중 응답 메시지의 XML이 구문 분석됩니다. 이러한 구문 분석을 방지하려면 파이프라인 구성 파일에서 SOAP 메시지 핸들러를 지정하지 말아야 합니다.

XML 인식 요청자 애플리케이션은 원격 제공자 모드 애플리케이션으로부터 SOAP 결합 메시지를 다시 수신할 수 있습니다. 이 경우, 요청자 애플리케이션은 SOAP 결합 해석 및 이 결합을 일반적인 응답 메시지와 구별하는 작업을 담당합니다. **INVOKE SERVICE** 명령이 **XML-ONLY WEBSERVICE**와 함께 사용되는 경우 CICS는 일반적으로 SOAP 결합을 수신했음을 나타내는 데 사용되는 응답 코드를 설정하지 않습니다.

XML-ONLY 옵션을 사용하지 않고 자체 XML 인식 서비스 요청자 애플리케이션을 작성하는 경우 다음 단계를 완료하십시오.

프로시저

1. 채널을 작성하고 이를 컨테이너로 채우십시오. 제어 컨테이너는 모두 CHAR 모드로 채워져야 합니다. 각 컨테이너에 다음 정보를 제공하십시오.

DFHWS-PIPELINE

아웃바운드 요청에 사용되는 PIPELINE 자원의 이름입니다.

DFHWS-URI

대상 웹 서비스의 URI입니다.

DFHWS-BODY

아웃바운드 SOAP 요청에서 SOAP 본문의 콘텐츠입니다. 파이프라인에 CICS 제공 SOAP 메시지 핸들러가 포함되는 경우 이 컨테이너를 제공하십시오. 메시지 핸들러는 본문을 포함한 전체 SOAP 메시지를 구성합니다.

DFHREQUEST

파이프라인에 전달될 전체 요청 메시지입니다. 메시지에 SOAP을 사용하지 않거나 프로그램에서 엔벨로프를 포함하는 전체 SOAP 메시지를 빌드하려는 경우에 이 컨테이너를 사용하십시오. 파이프라인은 아웃바운드 메시지로 중복 SOAP 헤더가 전송되는 것을 피하기 위해 CICS 제공 SOAP 메시지 핸들러를 포함하지 않아야 합니다.

컨테이너 DFHWS-BODY에 SOAP 본문이 제공되는 경우 DFHREQUEST는 비어 있어야 합니다. DFHWS-BODY 및 DFHREQUEST 둘 다에 콘텐츠를 제공하는 경우 CICS는 DFHREQUEST를 사용합니다.

DFHWS-XMLNS

네임스페이스 접두부를 요청의 XML 콘텐츠에 대한 네임스페이스로 맵핑하는 이름 값 쌍의 목록입니다.

DFHWS-SOAPACTION

컨테이너 DFHWS-BODY에서 지정된 SOAP 메시지에 추가할 SOAPAction 헤더입니다.

팁: 채널에 컨테이너 DFHWS-NOABEND를 추가하는 경우 DFHPIRT가 호출되면 DFHPIRT 내에서 이상 종료가 발생되지 않습니다. 이러한 작업은 C/C++ 프로그램을 실행 중인 경우에 유용합니다. DFHERROR 컨테이너를 통해 오류를 처리할 수 있기 때문입니다.

2. 프로그램 DFHPIRT에 대한 링크입니다. 다음 명령을 사용하십시오.

```
EXEC CICS LINK PROGRAM(DFHPIRT) CHANNEL(userchannel)
```

여기서, *userchannel*은 컨테이너를 보관하는 채널입니다. 아웃바운드 메시지는 파이프라인에서 메시지 핸들러 및 헤더 처리 프로그램에 의해 처리되고 웹 서비스 제공자에게 전송됩니다.

3. 같은 채널에서 웹 서비스 응답을 포함하는 컨테이너를 검색하십시오. 웹 서비스 제공자의 응답은 성공적인 응답이거나 SOAP 결함일 수 있습니다. 웹 서비스 요청자 애플리케이션은 서비스 제공자의 두 응답 유형 모두 처리할 수 있어야 합니다. 전체 응답은 다음 컨테이너에 포함됩니다.

DFHRESPONSE

웹 서비스 제공자에서 수신한 SOAP 응답의 엔벨로프를 포함하는 전체 응답입니다.

DFHWS-BODY

파이프라인에 CICS 제공 SOAP 메시지 핸들러가 포함될 때 SOAP 본문
의 콘텐츠입니다.

DFHERROR

파이프라인의 오류 정보입니다.

참고: 일부 오류 케이스에서 DFHWS-BODY가 갱신되지 않을 수 있습니
다. SOAP 결함에 대해 DFHRESPONSE를 확인해야 합니다.

웹 서비스에 Java 사용

Java를 사용하여 웹 서비스 애플리케이션을 작성할 수 있습니다. 이러한 애플리케이션
을 작성하는 데에는 다른 프로그래밍 언어가 사용된 기법과 비교할 때 다른 기법이 사
용됩니다.

대부분의 비Java 프로그래밍 언어의 경우 웹 서비스 지원을 사용하여 애플리케이션을
사용으로 설정합니다. 웹 서비스 지원 사용은 CICS가 데이터를 웹 서비스에서 애플리
케이션에 적합한 양식으로 변환하고 이를 컨테이너 또는 COMMAREA에 배치한다는
것을 의미합니다. Java 애플리케이션과 함께 웹 서비스 지원을 사용할 수 있지만 다음
태스크는 Java 애플리케이션용 Java 웹 서비스 작성에 보다 적합한 메소드를 제공합니
다.

Java 제공자 모드 웹 서비스를 Axis2 JVM 서버에 배치

CICS에서 Axis2 애플리케이션을 제공자 모드 웹 서비스로 배치할 수 있습니다. 이러
한 애플리케이션은 일반적으로 JAX-WS를 사용하여 생성되며 Java가 사용 설정된 파
이프라인에서 호스팅할 수 있습니다.

다음 이유 중 하나로 이 메소드를 사용하여 Java 애플리케이션을 배치하려고 할 수 있
습니다.

- Axis2 핸들러 인터페이스를 사용하는 기존 투자가 있습니다.
- CICS 파이프라인 구성을 사용하려고 합니다.

참고: Axis2 스타일 애플리케이션은 WEBSERVICE 자원을 사용하지 않습니다. 이러
한 애플리케이션은 Axis2 프로그래밍 모델을 사용하여 CICS와 상호작용하기 때문에 일
부 CICS 웹 서비스 지원을 사용할 수 없습니다. 다음 서비스는 Axis2 스타일 애플리
케이션에서 완전히 지원되지는 않습니다.

- SOAPFAULT CREATE
- WSACONTEXT GET
- 360 페이지의 『DFHWS-OPERATION 컨테이너』
- 359 페이지의 『DFHWS-MEP 컨테이너』

- 366 페이지의 『DFHWS-USERID 컨테이너』
- 361 페이지의 『DFHWS-TRANID 컨테이너』
- 웹 서비스 보안

시작하기 전에

Axis2에 배치하기에 적합한 Java 애플리케이션(예: JAX-WS를 사용하는 POJO 애플리케이션)이 있어야 합니다. 이 태스크에서 다음 POJO 애플리케이션이 예제로 사용됩니다.

```
/**
 * Simple example
 */
@javax.jws.WebService(targetNamespace = "com.ibm.cics.example", name = "pojoExample")
public class TestAxis2
{
    public String getMessage(String input)
    {
        return "CICS got this: '" + input + "'";
    }
}
```

이 애플리케이션은 WSDL을 생성하는 데 사용되는 XML 네임스페이스 및 웹 서비스와 연관시킬 이름을 지정합니다.

이 애플리케이션의 Java 코드를 컴파일해야 하고 JAX-WS 생성기를 실행하여 애플리케이션을 TestAxis2.jar이라고 하는 jar 파일로 패키징합니다. 이러한 작업은 다음 코드를 발행하여 수행할 수 있습니다.

```
javac TestAxis2.java
wsngen -cp . TestAxis2 -wsdl
jar -cvf TestAxis2.jar *
```

JAX-WS 생성기는 WSDL 문서 및 Axis2에서 사용되는 바인딩도 작성합니다.

이 태스크 정보

Axis2 웹 서비스를 배치하려면 웹 서비스를 위한 파이프라인 인프라를 작성해야 합니다. 파이프라인을 작성한 경우 웹 서비스를 작성할 수 있습니다. 필요한 수의 웹 서비스에 대해 작성된 파이프라인을 재사용할 수 있습니다. 다음 단계는 파이프라인 및 웹 서비스를 작성하는 방법에 대해 설명합니다.

참고: WEBSERVICE 자원은 이 태스크의 일부로 작성되거나 설치되지 않습니다.

프로시저

1. 파이프라인 인프라를 작성하십시오.

- a. Java 파이프라인에 대한 웹 서비스 인프라를 작성하십시오. 자세한 정보는 257 페이지의 『SOAP 서비스 제공자를 위한 CICS 인프라 작성』의 내용을 참조하십시오.

- b. Axis2 저장소를 작성하십시오. 이를 수행하려면 `$CICS_HOME/lib/pipeline/repository`에 있는 제공된 저장소의 사본을 작성하십시오.
 - c. 파이프라인 구성 파일에 `<repository>` 요소를 추가하십시오. 이 요소는 사용자가 작성한 Axis2 저장소의 이름을 지정해야 합니다.
 - d. PIPELINE 자원을 작성하고 사용으로 설정하십시오.
2. 파이프라인과 연관된 각 웹 서비스에 대해 다음 단계를 반복하여 웹 서비스를 작성하십시오.
- a. Axis2 애플리케이션을 Axis2 저장소에 배치하십시오. 예를 들어, 예제에서 작성된 jar 파일은 저장소 디렉토리에 있는 `servicejars`라고 하는 디렉토리에 배치되어야 합니다. 이 디렉토리가 없는 경우에는 작성해야 합니다.
 - b. 웹 서비스를 위한 URIMAP 자원을 정의하고 설치하십시오. URIMAP 자원은 웹 서비스와 연관된 URI 및 PIPELINE 자원을 지정해야 합니다. URI는 URI에 대한 Axis2 이름 지정 규칙을 따라야 합니다. 기본 Axis2 이름 지정 규칙은 `/name_of_service서비스.name_of_portPort/suffix`입니다. 여기서, `name_of_service`는 WSDL에서 웹 서비스의 이름이고 `name_of_port`는 WSDL에서 포트의 이름이며 `suffix`는 사용자가 정의할 수 있는 선택적 접미부입니다. 이전 예제에서 다음 URIMAP 자원을 사용할 수 있습니다.

```

Urimap      : EXAMPLE
Group       : EXAMPLE
STatus      : Enabled
USAge       : Pipeline
SCHEME      : HTTP
POrt        : No
HOST        : *
PAth        : /TestAxis2Service.pojoExamplePort/example/TestAxis2
TRansaction : CPIH
PiPeline    : EXAMPLE

```

이 예제에서는 사용되는 PIPELINE 자원을 EXAMPLE이라고 하는 것으로 가정합니다.

다음에 수행할 작업

웹 서비스가 올바르게 실행되는지 테스트하십시오.

XML을 생성하고 구문 분석하는 Java 웹 서비스 작성

XML을 스스로 구문 분석하고 생성하는 Java 애플리케이션을 작성할 수 있습니다. 이러한 애플리케이션은 다른 프로그래밍 언어로 작성된 XML 인식 애플리케이션과 일치하지만 XML 처리에 표준 Java 기술을 사용한다는 이점이 있습니다.

프로시저

1. XML-ONLY WEBSERVICE 자원을 작성하십시오. 자세한 정보는 170 페이지의 『XML 인식 서비스 요청자 애플리케이션 작성』 또는 168 페이지의 『XML 인식 서비스 제공자 애플리케이션 작성』의 내용을 참조하십시오.
2. SOAP 메시지의 본문에 대해 XML을 구문 분석하고 생성할 수 있는 Java 웹 서비스를 작성하십시오. 다음과 같은 기능이 있는 Java 웹 서비스를 작성하는 데 도움이 되는 다양한 도구(예: Java 6 JAXB(Java Architecture for XML Binding) 라이브러리)를 사용할 수 있습니다.
3. 옵션: 제공자 파이프라인을 사용하고 있고 SOAP 결함 메시지가 요청자에게 리턴 되도록 하는 기능을 추가하려는 경우 JCICS SoapFault 클래스를 사용하여 **EXEC CICS SOAPFAULT CREATE** 명령을 발행하십시오.
4. 옵션: 요청자 파이프라인을 사용하고 있는 경우 JCICS 서비스 클래스를 사용하여 **EXEC CICS INVOKE SERVICE** 명령을 발행하십시오.

COBOL 인터페이스가 있는 Java 웹 서비스 작성

다른 프로그래밍 언어에서 사용된 것과 같은 기법을 사용하여 CICS와 상호작용하는 Java 애플리케이션을 작성할 수 있습니다. 이러한 애플리케이션을 작성하려면 Java 코드를 작성하거나 생성하여 구조화된 COMMERA- 또는 container- 스타일 데이터를 작성해야 합니다.

프로시저

1. DFHWS2LS를 사용하여 웹 서비스를 위한 COBOL 언어 구조를 작성하십시오.
2. COBOL 언어 구조를 생성하고 구문 분석하는 Java 웹 서비스를 작성하십시오. Java 프로그램이 기존 CICS 애플리케이션 데이터에 액세스하기 위해 사용하는 도구에 대한 자세한 정보 및 COBOL 언어 구조를 생성하고 구문 분석할 수 있는 Java 웹 서비스 작성 방법 예제에 대한 링크는 애플리케이션 개발에서 Java의 구조 데이터와 상호 작용의 내용을 참조하십시오.
3. 옵션: 제공자 파이프라인을 사용하고 있고 SOAP 결함 메시지가 요청자에게 리턴 되도록 하는 기능을 추가하려는 경우 JCICS SoapFault 클래스를 사용하여 **EXEC CICS SOAPFAULT CREATE** 명령을 발행하십시오.
4. 옵션: 요청자 파이프라인을 사용하고 있는 경우 JCICS 서비스 클래스를 사용하여 CICS 서비스 API와 인터페이스하고 **EXEC CICS INVOKE SERVICE** 명령을 발행하십시오.

요청자 모드 JAX-WS 웹 서비스 배치

CICS에서 JAX-WS 애플리케이션을 요청자 모드 웹 서비스로 배치할 수 있습니다. 그러나 이러한 애플리케이션은 **EXEC CICS INVOKE** 명령을 사용하지 않는 대신 JAX-WS를 사용하여 원격 웹 서비스와 상호작용합니다.

시작하기 전에

OSGi를 지원하도록 JVM 서버가 구성되어 있어야 합니다. 자세한 정보는 구성에서 JVM 서버 설정의 내용을 참조하십시오.

이 태스크 정보

JAX-WS 애플리케이션을 요청자 모드 웹 서비스로 배치하는 것의 장점은 zEnterprise Application Assist Processor(zAAP)를 사용하는 플랫폼에 의존하지 않는 웹 서비스 요청자 애플리케이션을 작성할 수 있다는 것입니다. zAAP를 사용하면 트랜잭션의 비용을 줄일 수 있습니다. 자세한 정보는 IBM Redbooks 서적: zAAP(zSeries Application Assist Processor) 구현을 참조하십시오.

프로시저

1. Java에서 웹 서비스 요청자 애플리케이션을 작성하고 적절한 API(예: JAX-WS(Java API for XML web services))를 사용하여 원격 웹 서비스를 호출하십시오.
2. 옵션: 원격 웹 서비스를 시작하는 데 JAX-WS를 사용하는 경우 SOAP 메시지 생성, 네트워크 통신 처리 및 SOAP 응답 처리에도 JAX-WS를 사용해야 합니다.
3. JVM 서버에서 Java 애플리케이션을 배치하고 설치하십시오.

다음에 수행할 작업

웹 서비스가 올바르게 시작되는지 테스트하십시오.

Liberty JVM 서버에서 Java 제공자 모드 웹 서비스 배치

Liberty JVM 서버에서 웹 애플리케이션을 제공자 모드 웹 서비스로 배치할 수 있습니다. 이러한 애플리케이션은 Java 표준 JAX-WS 및 JAXB를 사용하여 작성됩니다.

이 태스크 정보

CICS TS V5.3에는 JAX-WS(Java API for XML Web Service) 및 JAXB(Java Architecture for XML Binding)에 대한 기능을 제공하는 최신 WebSphere Application Server Liberty Profile(WLP) V8.5.5가 포함됩니다. 이러한 기술을 함께 사용하면 Java에서 CICS 애플리케이션의 일부로 SOAP 웹 서비스를 작성할 수 있습니다. 다음 기사에서는 Eclipse를 설정하고 샘플 웹 서비스 프로젝트를 테스트하며 CICS에 샘플을 배치하고 JCICS를 사용하도록 샘플을 수정하며 웹 서비스 탐색기에서 이 샘플을 테스트하는 방법을 보여줍니다. 세부사항은 DeveloperWorks, CICS TS V5.2 오픈 베타 Liberty 프로파일의 JAX-WS 및 JAXB 자원의 내용을 참조하십시오.

다음 이유 중 하나로 이 메소드를 사용하여 Java 애플리케이션을 배치하려고 할 수 있습니다.

- Java에서 웹 서비스를 작성하려고 합니다.

- CICS 웹 서비스 지원을 사용하여 처리하기 어려울 수 있는 복잡한 WSDL 문서가 있습니다.
- 웹 서비스 애플리케이션의 핸들링을 zEnterprise Application Assist Processor(zAAP)에 분담하려고 합니다.

참고: Liberty JVM 서버에 배치된 웹 애플리케이션은 WEBSERVICE 또는 TCPIPSERVICE 자원을 사용하지 않습니다. 이러한 애플리케이션은 Liberty HTTP 리스너를 사용하여 웹 요청과 상호작용하기 때문에 CICS 웹 서비스 지원의 기능을 사용할 수 없습니다.

SOAP 메시지 유효성 검증

CICS 웹 서비스를 사용할 때 SOAP 메시지가 웹 서비스 설명에 포함되어 있는 스키마를 준수하는지 확인하기 위해 SOAP 메시지를 유효성 검증하도록 지정할 수 있습니다. 제공자 및 요청자 모드 애플리케이션을 둘 다 유효성 검증할 수 있습니다.

시작하기 전에

웹 서비스 배치를 개발하고 테스트하는 동안 전체 유효성 검증은 서비스 요청자와 서비스 제공자 사이의 메시지 교환에 있어서 문제점을 발견할 수 있도록 지원합니다. 그러나 SOAP 메시지의 전체 유효성 검증에는 상당한 오버헤드가 따르고 완전히 테스트되는 프로덕션 애플리케이션에서 메시지를 유효성 검증하는 것은 바람직하지 않습니다.

CICS는 Java 프로그램을 사용하여 SOAP 메시지를 유효성 검증합니다. 따라서 SOAP 메시지를 유효성 검증하려면 Java 지원이 CICS 리전에서 사용으로 설정되어 있어야 합니다.

이 태스크 정보

SOAP 메시지는 애플리케이션 데이터 구조로 변환되기 전 및 SOAP 메시지가 애플리케이션 데이터 구조에서 생성될 때 유효성 검증됩니다. SOAP 메시지는 WSDL에서 XML 스키마를 사용하여 유효성 검증되고 CICS의 변환 요구사항에 대해 다시 유효성 검증됩니다. WEBSERVICE 자원의 **WSDLFILE** 속성에서 지정된 WSDL 파일을 사용하거나 WEBSERVICE 자원의 **ARCHIVEFILE** 속성에서 지정된 .zip 파일에 포함되어 있는 WSDL 파일을 사용할 수 있습니다. 두 속성 모두 지정된 경우 **ARCHIVEFILE** 속성에서 지정된 아카이브 파일의 WSDL 파일이 사용됩니다.

유효성 검증이 꺼진 경우 CICS는 Java 프로그램을 사용하지 않습니다. CICS는 SOAP 메시지에 잘 구성된 XML이 포함되어 있는지 확인하고 변환해야 하는 정도로만 SOAP 메시지를 유효성 검증합니다. 따라서 SOAP 메시지는 성공적으로 유효성 검증되지만 런타임 환경에서 실패할 수 있고 그 반대의 경우도 발생할 수 있습니다.

프로시저

1. 옵션: CICS 리전에서 JVM 서버를 설정하십시오. OSGi 프레임워크 또는 Axis2에서 SOAP 유효성 검증을 실행할 수 있지만 Liberty 프로파일에서는 실행할 수 없습니다. CICS는 OSGi 프레임워크를 사용하는 JVM 서버를 빠르게 설정할 수 있도록 샘플을 제공합니다. Java 파이프라인을 사용하는 경우 추가 JVM 서버를 정의하지 않아도 됩니다.

- 유효성 검증에 사용할 JVM 서버가 이미 정의되어 있는 경우 JVMSERVER 자원의 이름을 참조하도록 그룹 DFHPIVAL에서 DFHPIVAL 프로그램 정의를 수정하십시오. DFHPIVAL 정의는 잠그지 않으므로 편집할 수 있습니다. 기본적으로 이 정의는 DFHJVMS를 참조합니다.

또는

- CICS 제공 샘플을 사용하려는 경우 샘플 JVM 서버 DFHJVMS를 그룹 DFH\$OSGI에 설치해야 합니다. 자세한 정보는 구성에서 JVM 서버 설정의 내용을 참조하십시오.
2. WEBSERVICE 자원과 연관된 웹 서비스 설명이 있는지 확인하십시오. 이 연관은 WSDL 파일 또는 한 개 이상의 WSDL 파일을 포함하는 .zip 파일이 파이프라인 스캔 중 파이프라인의 픽업 디렉토리에 있는 경우 자동으로 작성되는 WEBSERVICE 자원에 대해 작성됩니다.

RDO를 사용하여 작성되는 WEBSERVICE 정의의 경우 웹 서비스 설명은 WSDLFILE 속성에 지정됩니다.

3. WEBSERVICE 자원의 **VALIDATION=YES** 속성을 지정하여 웹 서비스 유효성 검증을 켜십시오. 자원을 정의할 때 유효성 검증이 필요한지 여부를 지정할 수 있고 자원이 설치된 후에는 이 설정을 변경할 수 있습니다.

결과

SOAP 메시지가 유효한지 여부를 알아보려면 시스템 로그를 확인하십시오. 메시지 DFHPI1002는 SOAP 메시지가 성공적으로 유효성 검증되었음을 나타내고 메시지 DFHPI1001은 유효성 검증에 실패했음을 나타냅니다.

다음에 수행할 작업

유효성 검증이 더 이상 필요하지 않으면 끄십시오.

웹 서비스의 유효성 검증 상태 변경

다음 인터페이스를 사용하여 웹 서비스의 유효성 검증 상태를 변경할 수 있습니다.



CICS Explorer

 CICS Explorer 관리 보기

웹 서비스 보기의 유효성 검증 상태 속성 사용

CICSplex SM

WEBSERVICE 정의 보기

CEMT

 SET WEBSERVICE 명령

CICS SPI

 SET WEBSERVICE 명령

제 8 장 JSON 웹 서비스 작성

기존 CICS 애플리케이션을 JSON 웹 서비스로 노출하여 SOAP 웹 서비스 제공자 역할을 할 새 CICS 애플리케이션을 작성할 수 있습니다.

시작하기 전에

JSON 웹 서비스 작성을 시작하기 전에 JSON 웹 서비스를 지원하도록 CICS 시스템을 구성해야 합니다. 자세한 정보는 261 페이지의 『JSON 서비스 제공자를 위한 CICS 인프라 작성』의 내용을 참조하십시오.

이 태스크 정보

CICS JSON 지원은 사용자가 새 JSON 웹 서비스 제공자 애플리케이션에 필요한 아티팩트를 작성하거나 기존 애플리케이션을 JSON 웹 서비스 제공자로 사용할 수 있도록 돕는 제공 유틸리티입니다.

CICS JSON 지원은 고급 언어 구조에서 JSON 스키마를 작성하거나 기존 JSON 스키마에서 고급 언어 구조를 작성할 수 있으며 COBOL, C/C++ 및 PL/I를 지원합니다. 또한 JSON 메시지를 컨테이너 및 COMMAREA로 또는 그 반대로 자동 런타임 변환을 가능하게 하기 위해 사용되는 정보를 생성합니다. 이 정보는 파이프라인 처리 중 CICS JSON 웹 서비스 지원에서 사용됩니다.

다음 프로시저에서 설명하는 대로 JSON 웹 서비스를 작성하고 작성한 웹 서비스가 올바르게 작동하는지 유효성 검증하십시오.

프로시저

1. JSON 웹 서비스를 작성하십시오. JSON 지원을 사용하여 JSON 스키마 또는 언어 구조를 작성하고 작성한 항목을 CICS에 배치합니다. **PIPELINE SCAN** 명령을 사용하여 필수 CICS 자원을 자동 작성합니다.
2. JSON 웹 서비스를 시작하여 웹 서비스가 의도한 대로 작동하는지 테스트하십시오.

다음에 수행할 작업

이러한 단계는 다음 주제에서 보다 자세히 설명됩니다.

CICS JSON 지원

CICS JSON 지원은 JSON 스키마와 언어 구조 사이에 맵핑을 작성하는 일괄처리 유틸리티 세트입니다. 이 맵핑은 JSON과 애플리케이션 데이터 사이의 변환을 수행하기 위해 런타임 시 CICS에 의해 사용됩니다. 지원은 최소 프로그래밍 노력으로 서비스 제공자 및 서비스 요청자에서 사용하기 위한 CICS 애플리케이션의 빠른 배치를 지원합니다.

CICS에 JSON 지원을 사용할 때, 인바운드 메시지를 구문 분석하고 아웃바운드 메시지를 구성하기 위한 사용자 자체 코드를 작성할 필요가 없습니다. CICS는 JSON 메시지와 애플리케이션 프로그램의 데이터 구조 사이의 데이터를 맵핑합니다.

지원은 고급 언어 구조에서 JSON 스키마를 작성하거나 기존 JSON 스키마에서 고급 언어 구조를 작성할 수 있고, COBOL, C/C++ 및 PL/I를 지원합니다. 이는 컨테이너 및 COMMAREA에 대한 JSON 메시지의 자동화 런타임 변환을 가능하게 하거나 또는 그 반대에 사용되는 정보를 생성합니다.

CICS JSON 지원은 두 개의 유틸리티 프로그램을 포함합니다.

DFHLS2JS

언어 구조에서 웹 서비스 바인딩 파일을 생성합니다. 이 유틸리티는 JSON 스키마도 생성합니다.

DFHJS2LS

JSON 스키마에서 웹 서비스 바인딩 파일을 생성합니다. 이 유틸리티는 애플리케이션 프로그램에서 사용할 수 있는 언어 구조도 생성합니다.

양쪽 모두의 프로그램을 실행하기 위한 JCL 프로시저는 *hlq.XDFHINST* 라이브러리에 있습니다.

JSON 지원 유틸리티 프로그램 및 데이터 맵핑에 대한 자세한 정보는 다음 주제를 참조하십시오.

DFHLS2JS: 요청-응답 서비스를 위한 고급 언어 대 JSON 스키마 변환

DFHLS2JS 프로시저는 고급 언어 데이터 구조에서 JSON 스키마 파일을 생성합니다. DFHLS2JS는 CICS 애플리케이션 프로그램을 서비스 제공자로 표시할 때 사용할 수 있습니다.

DFHLS2JS와 기호 매개변수, 입력 매개변수와 설명, 작업 예제에 대한 작업 제어 명령문은 이 프로시저를 사용하는 데 도움을 줍니다.

DFHLS2JS JCL 프로시저는 데이터 세트 *HLQ.XDFHINST*에 설치됩니다. 여기서 *HLQ*는 CICS가 설치된 상위 레벨 규정자입니다.

DFHLS2JS 또는 DFHJS2LS 프로시저의 관련 사용 모드는 다음과 같이 요구사항에 따라 다릅니다.

- DFHLS2JS: 링크 가능 인터페이스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 링크 가능 인터페이스를 위한 JSON 스키마 대 고급 언어 변환
- DFHLS2JS: 요청-응답 서비스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 요청-응답 서비스를 위한 JSON 스키마 대 고급 언어 변환
- DFHJS2LS: RESTful 서비스를 위한 JSON 스키마 대 고급 언어 변환

DFHLS2JS의 작업 제어 명령문

JOB 작업을 시작합니다.

EXEC

프로시저 이름(DFHLS2JS)을 지정합니다.

INPUT.SYSUT1 DD

입력을 지정합니다. 입력 매개변수는 일반적으로 입력 스트림에 지정됩니다. 그러나 데이터 세트 또는 파티션된 데이터 세트의 멤버에 정의될 수 있습니다.

기호 매개변수

DFHLS2JS에서는 다음 기호 매개변수가 정의됩니다.

JAVADIR=*path*

DFHLS2JS에서 사용되는 Java 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/에 추가되어 /usr/lpp/*path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

PATHPREF=*prefix*

다른 매개변수 또는 ''(빈 문자열)(접두부를 사용하지 않는 경우)에서 사용되는 z/OS UNIX 디렉토리 경로를 확장하는 접두부를 지정합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **PATHPREF** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

SERVICE=*value*

이 매개변수는 IBM 지원 센터에서 지시하는 경우에만 사용합니다.

TMPDIR=*tmpdir*

DFHLS2WS가 z/OS UNIX에서 임시 작업공간으로 사용하는 디렉토리의 위치를 지정합니다. 작업을 실행하는 데 사용되는 사용자 ID는 이 디렉토리에 대한 읽기 및 쓰기 권한이 있어야 합니다.

기본값은 /tmp입니다.

TMPFILE=*tmpprefix*

DFHLS2JS가 임시 작업공간 파일의 이름을 구성하기 위해 사용하는 접두부를 지정합니다.

기본값은 LS2JS입니다.

USSDIR=*path*

UNIX 시스템 서비스 파일 시스템에서 CICS TS 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/cicsts/에 추가되어 /usr/lpp/cicsts/*path*의 전체 경로 이름을 생성합니다. 기본값을 사용하는 경우 '.'(마침표)로 지정되어야 합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **USSDIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

임시 작업공간

DFHLS2JS는 런타임에 다음 세 개 임시 파일을 작성합니다.

tmpdir/tmpprefix.in

tmpdir/tmpprefix.out

tmpdir/tmpprefix.err

여기서,

*tmpdir*은 **TMPDIR** 매개변수에 지정된 값입니다.

*tmpprefix*는 **TMPFILE** 매개변수에 지정된 값입니다.

TMPDIR 및 **TMPFILE**이 지정되지 않은 경우 파일의 기본 이름은 다음과 같습니다.

/tmp/LS2JS.in

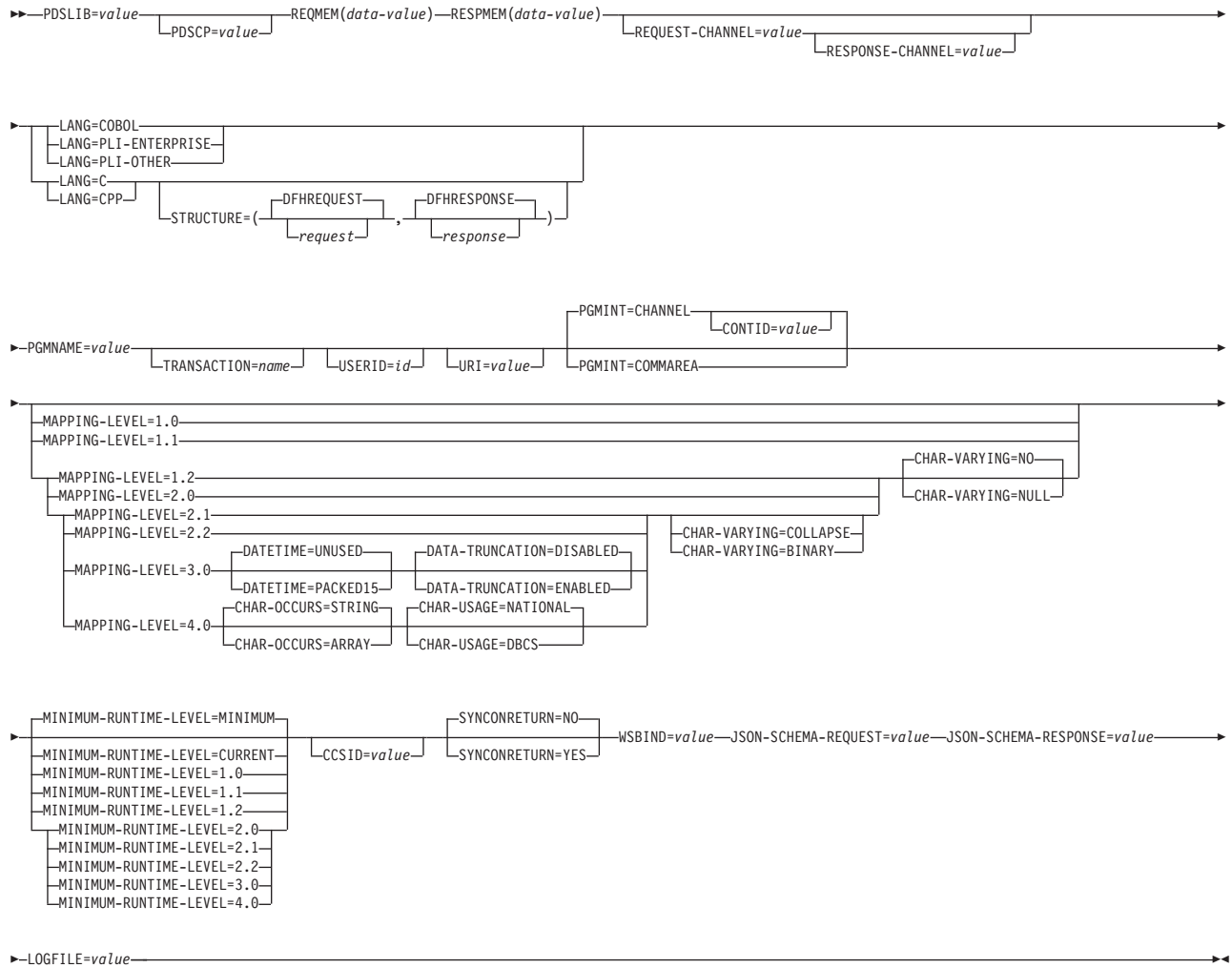
/tmp/LS2JS.out

/tmp/LS2JS.err

중요사항: DFHLS2JS는 z/OS UNIX 파일 또는 데이터 세트 멤버에 대한 액세스를 차단하지 않습니다. 따라서 두 개 이상의 DFHLS2JS 인스턴스가 동시에 실행되고 동일한 임시 작업공간 파일을 사용하는 경우, 다른 작업이 작업공간 파일을 사용하는 동안 특정 작업이 해당 파일을 겹쳐쓸 수 있어 예상치 못한 실패를 경험할 수 있습니다.

따라서 이러한 상황을 방지하는 이름 지정 규칙과 작동 프로시저를 고안해야 합니다. 예를 들어, 시스템 기호 매개변수 **SYSUID**를 사용하여 개별 사용자에게 고유한 작업공간 파일 이름을 생성할 수 있습니다. 이러한 임시 파일은 작업이 종료되기 전에 삭제됩니다.

DFHLS2JS의 입력 매개변수



매개변수 사용

- 입력 매개변수는 순서에 관계 없이 지정할 수 있습니다.
- 각 매개변수는 행을 바꿔 시작해야 합니다.
- 매개변수는 사용 시 해당 연속 문자와 함께 열 72를 넘어가서는 안됩니다.
열 73 - 80은 공백이어야 합니다.
- 매개변수가 너무 길어 한 행에 맞지 않는 경우에는 행 끝에서 별표(*) 문자를 사용하여 매개변수가 다음 행에서 연속됨을 표시해야 합니다. 별표 앞의 모든 내용은 공백을 포함하여 매개변수의 일부로 간주됩니다. 예를 들어, 다음과 같습니다.

```
WSBIND=wsbinddir*
/app1
```

구문은 다음과 같습니다.

```
WSBIND=wsbinddir/app1
```

- 행에서 첫 번째 문자 위치의 # 문자는 주석 문자입니다. 이 행은 무시됩니다.
- 행에서 마지막 문자 위치의 쉼표는 선택적 섹션 구분 기호이며 무시됩니다.

매개변수 설명

CCSID=*value*

런타임에 애플리케이션 데이터 구조에서 문자 데이터를 인코딩하는 데 사용되는 CCSID를 지정합니다. 이 매개변수의 값은 **LOCALCCSID** 시스템 초기화 매개변수의 값을 대체합니다. *value*는 Java 및 z/OS Unicode Services User's Guide and Reference에서 지원하는 EBCDIC CCSID여야 합니다. 이 매개변수를 지정하지 않으면 시스템 초기화 매개변수에 지정된 CCSID를 사용하여 애플리케이션 데이터 구조가 인코딩됩니다.

CHAR-VARYING={NO|NULL|COLLAPSE|BINARY}

맵핑 레벨이 1.2 이상일 때 언어 구조의 문자 필드가 맵핑되는 방식을 지정합니다. COBOL의 문자 필드가 X 유형의 Picture 절인 경우(예: PIC(X) 10) C/C++의 문자 필드는 문자 배열입니다. 다음 옵션을 선택할 수 있습니다.

NO 문자 필드가 JSON 문자열에 맵핑되고 고정 길이 필드로 처리됩니다. 데이터의 최대 길이는 필드 길이와 같습니다. 맵핑 레벨이 2.0 이하인 COBOL 및 PL/I의 경우 **CHAR-VARYING** 매개변수의 기본값은 NO입니다.

이 값은 엔터프라이즈 및 기타 PL/I 언어 구조에는 적용되지 않습니다.

NULL

문자 필드가 JSON 문자열에 맵핑되고 널 종료 문자열로 처리됩니다. CICS는 JSON 메시지에서 변환될 때 종료 널 문자를 추가합니다. 문자열의 최대 길이는 언어 구조에 표시된 길이보다 한 문자 적은 길이로 계산됩니다. C/C++의 경우 **CHAR-VARYING** 매개변수의 기본값은 NULL입니다.

이 값은 엔터프라이즈 및 기타 PL/I 언어 구조에는 적용되지 않습니다.

COLLAPSE

문자 필드가 JSON 문자열에 맵핑됩니다. 필드에서 후미 공백은 JSON 메시지에 포함되지 않습니다. 인바운드 JSON 메시지는 선두, 후미, 임베디드 공백을 모두 제거하기 위해 구문 분석됩니다. 맵핑 레벨이 2.1 이상인 COBOL 및 PL/I의 경우 **CHAR-VARYING** 매개변수의 기본값은 COLLAPSE입니다.

BINARY

문자 필드가 Base-64 인코딩 데이터를 포함하는 JSON 문자열에 맵핑되고 고정 길이 필드로 처리됩니다. **CHAR-VARYING** 매개변수의 BINARY 값은 맵핑 레벨이 2.1 이상인 경우에만 사용할 수 있습니다.

CHAR-OCCURS={STRING|ARRAY}

맵핑 레벨이 4.0 이상일 때 언어 구조의 문자 배열이 맵핑되는 방식을 지정합니다 (예: PIC X OCCURS 20). 이 매개변수는 COBOL 언어만 사용할 수 있습니다.

ARRAY

문자 배열이 JSON 배열에 맵핑됩니다. 이는 모든 문자가 개별 JSON 요소에 맵핑됨을 의미합니다. 맵핑 레벨이 3.0 이하인 경우의 동작이기도 합니다.

STRING

문자 배열이 JSON 문자열에 맵핑됩니다. 이는 전체 COBOL 배열이 단일 JSON 요소에 맵핑됨을 의미합니다.

CHAR-USAGE={NATIONAL|DBCS}

COBOL에서 국가적 데이터 유형인 PIC N은 UTF-16 또는 DBCS 데이터에 사용될 수 있습니다. 이 설정은 NSYMBOL 컴파일러 옵션에 의해 제어됩니다. 데이터가 적절하게 처리되는지 확인하려면 보조의 **CHAR-USAGE** 매개변수를 NSYMBOL 컴파일러 옵션과 동일한 값으로 설정해야 합니다. 이는 주로 UTF-16을 사용할 때 CHAR-USAGE=NATIONAL로 설정됩니다.

DBCS PIC (*n*) 필드의 데이터는 UTF-16 인코딩 데이터로 처리됩니다.

NATIONAL

PIC (*n*) 필드의 데이터는 DBCS 인코딩 데이터로 처리됩니다.

CONTID=*value*

서비스 제공자에서, JSON 메시지를 나타내는 데 사용되는 상위 레벨 데이터 구조를 보유하는 컨테이너의 이름을 지정합니다.

CICS가 대상 애플리케이션 프로그램으로 전달하는 컨테이너의 길이는 요청 컨테이너의 길이와 응답 컨테이너의 길이 중 긴 값입니다.

DATA-TRUNCATION={DISABLED|ENABLED}

가변 길이 데이터가 고정 길이 필드 구조에서 용인되는지 여부를 지정합니다.

DISABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 거부하고 오류 메시지를 발행합니다.

ENABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 용인하고 누락된 데이터를 널값으로 처리합니다.

DATETIME={UNUSED|PACKED15}

고급 언어 구조의 잠재적 ABSTIME 필드가 시간소인으로 맵핑되는지 여부를 지정합니다.

PACKED15

길이가 15(8바이트)인 팩형 10진수 필드가 CICS ABSTIME 필드로 처리되고 시간소인으로 맵핑됩니다.

UNUSED

길이가 15(8바이트)인 팩형 10진수 필드가 시간소인으로 처리되지 않습니다.

이 매개변수는 맵핑 레벨 3.0에서 설정할 수 있습니다.

JSON-SCHEMA-REQUEST=value

필수 매개변수입니다.

값은 요청 JSON 스키마가 저장되는 UNIX System Services 위치를 나타냅니다.

JSON-SCHEMA-RESPONSE=value

필수 매개변수입니다.

값은 응답 JSON 스키마가 저장되는 UNIX System Services 위치를 나타냅니다.

LANG=COBOL

고급 언어 구조의 프로그래밍 언어가 COBOL인 것으로 지정합니다.

LANG=PLI-ENTERPRISE

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I인 것으로 지정합니다.

LANG=PLI-OTHER

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I 이외의 PL/I 레벨인 것으로 지정합니다.

LANG=C

고급 언어 구조의 프로그래밍 언어가 C인 것으로 지정합니다.

LANG=CPP

고급 언어 구조의 프로그래밍 언어가 C++인 것으로 지정합니다.

LOGFILE=value

DFHLS2JS가 해당 활동 로그 및 추적 정보를 기록하는 파일의 완전한 z/OS UNIX 이름입니다. DFHLS2JS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다.

이 파일은 일반적으로 사용하지 않지만 DFHLS2JS 관련 문제점이 발생하는 경우 IBM 서비스 조직이 요청할 수 있습니다.

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0}

웹 서비스 바인딩 파일과 JSON 스키마를 생성할 때 DFHLS2JS가 사용하는 맵핑 레벨을 지정합니다. 다음 옵션을 선택할 수 있습니다.

1.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

- 1.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 3.0 사용 가능한 전체 옵션 세트를 사용하여 JSON 스키마를 생성하려면 이 맵핑 레벨을 사용하십시오.
- 4.0 이 맵핑 레벨은 CICS TS 5.2 리전에 사용됩니다. 이 맵핑 레벨에서는 COBOL OCCURS DEPENDING ON 필드와 **CHAR-OCCURS** 매개변수를 사용할 수 있습니다.

맵핑 레벨에 대한 자세한 정보는 CICS JSON 지원에 대한 맵핑 레벨을 참조하십시오.

MINIMUM - RUNTIME -

LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0|CURRENT}

웹 서비스 바인딩 파일이 배치될 수 있는 최소 CICS 런타임 환경을 지정합니다. 지정한 다른 매개변수와 일치하지 않는 레벨을 선택하면 오류 메시지를 받게 됩니다. 다음 옵션을 선택할 수 있습니다.

MINIMUM

선택한 매개변수의 경우 CICS의 런타임 레벨 중 가장 낮은 레벨이 자동으로 할당됩니다.

- 1.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

2.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

3.0 생성된 웹 서비스 바인딩 파일이 CICS TS 4.1 이상 리전에 배치됩니다.

참고: JSON 지원은 CICS TS 4.2 이상에서만 사용할 수 있습니다.

4.0 생성된 웹 서비스 바인딩 파일이 CICS TS 5.2 이상 리전에 배치됩니다. 이 런타임 레벨에서는 **MAPPING-LEVEL** 매개변수에 4.0 이하 맵핑 레벨을 사용할 수 있습니다. 이 레벨에서는 선택적 매개변수를 사용할 수 있습니다.

CURRENT

생성된 웹 서비스 바인딩 파일이 웹 서비스 바인딩 파일을 생성하기 위해 사용하는 런타임 레벨과 동일한 런타임 레벨에서 CICS 리전에 배치됩니다.

PDSLIB=value

처리할 고급 언어 데이터 구조를 포함하는 파티션된 데이터 세트의 이름을 지정합니다. 요청 및 응답에 사용되는 데이터 세트 멤버는 **REQMEM** 및 **RESPMEM** 매개변수에 지정됩니다.

제한사항: 파티션된 데이터 세트 내 레코드는 80바이트의 고정 길이를 갖습니다.

PDSCP=value

REQMEM 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에서 사용되는 코드 페이지를 지정합니다. 여기서 *value*는 CCSID 번호 또는 Java 코드 페이지 번호입니다. 이 매개변수가 지정되지 않으면 z/OS UNIX System Services 코드 페이지를 사용합니다. 예를 들어, **PDSCP=037**을 지정할 수 있습니다.

PGMINT={CHANNEL|COMMAREA}

서비스 제공자의 경우, CICS가 대상 애플리케이션 프로그램으로 데이터를 전달하는 방식을 지정합니다.

CHANNEL

CICS가 채널 인터페이스를 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

- 맵핑 레벨이 3.0 이하인 경우에는 입출력에 모두 사용되는 하나의 컨테이너만 채널에 포함될 수 있습니다. 컨테이너의 이름을 지정하려면 **CONTID** 매개변수를 사용하십시오. 기본 이름은 DFHWS-DATA입니다.
- 맵핑 레벨 3.0에서는 채널에 여러 컨테이너가 포함될 수 있습니다. **REQUEST-CHANNEL** 및 **RESPONSE-CHANNEL** 매개변수를 사용하십시오. **PDSLIB**, **REQMEM** 또는 **RESPMEM**은 지정하지 마십시오.

COMMAREA

CICS가 통신 영역을 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

대상 애플리케이션 프로그램이 요청을 처리한 경우에는 동일한 메커니즘으로 응답을 리턴해야 합니다. 즉, 요청이 통신 영역에서 수신된 경우에는 응답도 통신 영역에서 리턴되어야 하고 요청이 컨테이너에서 수신된 경우에는 응답도 컨테이너에서 리턴되어야 합니다. CICS가 대상 애플리케이션 프로그램으로 전달하는 통신 영역 또는 컨테이너의 길이는 요청 통신 영역 또는 컨테이너의 길이와 응답 통신 영역 또는 컨테이너의 길이 중 긴 값입니다.

PGMNAME=value

웹 서비스로 노출될 대상 애플리케이션 프로그램에 대한 CICS PROGRAM 자원의 이름을 지정합니다. CICS 웹 서비스 지원은 이 프로그램에 링크됩니다.

REQMEM=value

웹 서비스 요청에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 지정합니다. 서비스 제공자의 경우 웹 서비스 요청은 애플리케이션 프로그램에 입력됩니다.

REQUEST-CHANNEL=value

채널 설명 문서의 이름과 위치를 지정합니다. 채널 설명은 웹 서비스 요청자로부터 JSON 메시지를 수신할 때 웹 서비스 제공자 애플리케이션이 해당 인터페이스에서 사용할 수 있는 컨테이너를 설명합니다. 채널 설명은 CICS 제공 채널 스키마를 준수해야 하는 XML 문서입니다. 자세한 정보는 221 페이지의 『채널 설명 문서 작성』을 참조하십시오.

이 매개변수는 맵핑 레벨 3.0에서만 사용할 수 있습니다.

RESPMEM=value

웹 서비스 응답에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 지정합니다. 서비스 제공자의 경우 웹 서비스 응답은 애플리케이션 프로그램에서 출력됩니다.

RESPONSE-CHANNEL=value

채널 설명 문서의 이름과 위치를 지정합니다. 채널 설명은 웹 서비스 요청자로 JSON 응답 메시지를 보낼 때 웹 서비스 제공자 애플리케이션이 해당 인터페이스에서 사용할 수 있는 컨테이너를 설명합니다. 채널 설명은 CICS 제공 채널 스키마를 준수해야 하는 XML 문서입니다. 자세한 정보는 221 페이지의 『채널 설명 문서 작성』을 참조하십시오.

이 매개변수는 맵핑 레벨 3.0에서만 사용할 수 있습니다.

STRUCTURE=(request,response)

C 및 C++의 경우에만, **REQMEM** 및 **RESPMEM** 매개변수에 지정된 파티션된 데이터 세트 멤버에 포함된 상위 레벨 구조의 이름을 지정합니다.

request

REQMEM 매개변수가 지정되는 경우 요청을 포함하는 상위 레벨 구조의 이름을 지정합니다. 기본값은 DFHREQUEST입니다.

파티션된 데이터 세트 멤버에는 사용자가 지정하는 이름을 갖는 상위 레벨 구조 또는 이름을 지정하지 않는 경우 DFHREQUEST 구조가 포함되어야 합니다.

response

RESPMEM 매개변수가 지정되는 경우 응답을 포함하는 상위 레벨 구조의 이름을 지정합니다. 기본값은 DFHRESPONSE입니다.

값을 지정하는 경우, 파티션된 데이터 세트 멤버에는 사용자가 지정하는 이름을 갖는 상위 레벨 구조 또는 DFHRESPONSE 구조(이름을 지정하지 않는 경우)가 포함되어야 합니다.

SYNCONRETURN={NO|YES}

원격 웹 서비스가 동기점을 발행할 수 있는지 여부를 지정합니다.

NO 원격 웹 서비스가 동기점을 발행할 수 없습니다. 이 값은 기본값입니다. 원격 웹 서비스가 동기점을 발행하면 ADPL 이상 종료와 함께 실패합니다.

YES 원격 웹 서비스가 동기점을 발행할 수 있습니다. YES를 선택하면 원격 웹 서비스에서 제어가 리턴되는 경우 원격 태스크가 개별 작업 단위로 커밋됩니다. 원격 웹 서비스가 복구 가능 자원을 갱신하고 리턴 후 실패하면 해당 자원에 대한 갱신을 백아웃할 수 없습니다.

TRANSACTION=name

서비스 제공자에서 이 매개변수는 파이프라인을 시작할 수 있는 별명 트랜잭션의 1 - 4자 이름을 지정합니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 TRANSACTION 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ # _ < >

URI=value

이 매개변수는 클라이언트가 웹 서비스에 액세스하기 위해 사용하는 상대 또는 절대 URI를 지정합니다. CICS는 DFHLS2JS로 작성된 웹 서비스 바인딩 파일에서 URIMAP 자원을 생성할 때 지정되는 값을 사용합니다. 이 매개변수는 URIMAP 정의가 적용되는 URI의 경로 컴포넌트를 지정합니다.

USERID=id

서비스 제공자에서 이 매개변수는 웹 클라이언트가 사용할 수 있는 1 - 8자 사용자 ID를 지정합니다. 애플리케이션 생성 응답 또는 웹 서비스의 경우 이 사용자 ID 아래에 별명 트랜잭션이 첨부됩니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 때 URIMAP 자원의 USERID 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ #

WSBIND=value

웹 서비스 바인딩 파일의 완전한 z/OS UNIX 이름입니다. DFHLS2JS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 파일 확장자는 .wsbind입니다.

기타 정보

- DFHLS2JS 실행을 위해 사용되는 사용자 ID는 UNIX System Services를 사용하도록 구성되어야 합니다. 사용자 ID는 CICS z/OS UNIX 파일 구조 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND**, **JSON Schema** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다.
- JCL의 최대 매개변수 길이는 100자입니다. 이 매개변수 길이는 **STDPARM**문을 사용하여 늘릴 수 있습니다. 자세한 정보는 *z/OS UNIX System Services User Guide*를 참조하십시오.

예제

```
//LS2JS JOB 'accounting information',name,MSGCLASS=A
// SET QT='''
//JAVAPROG EXEC DFHLS2JS,
// TMPFILE=&QT.&SYSUID.&QT,
//INPUT.SYSUT1 DD *
PDSLIB=CICSHLQ.SDFHSAMP
REQMEM=DFH0XCP4
RESPMEM=DFH0XCP4
JSON-SCHEMA-REQUEST=/u/exampleapp/json/example_request.json
JSON-SCHEMA-RESPONSE=/u/exampleapp/json/example_response.json
LANG=COBOL
LOGFILE=/u/exampleapp/wsbinding/example.log
MAPPING-LEVEL=4.0
CHAR-VARYING=COLLAPSE
PGMNAME=DFH0XCMN
URI=http://myserver.example.org:8080/exampleApp/example
PGMINT=COMMAREA
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbinding/example.wsbinding
/*
```

DFHJS2LS: 요청-응답 서비스를 위한 JSON 스키마 대 고급 언어 변환

DFHJS2LS 프로시저는 JSON 스키마에서 고급 언어 데이터 구조와 웹 서비스 바인딩 파일을 생성합니다. DFHJS2LS는 CICS 애플리케이션 프로그램을 서비스 제공자로 작성하려는 경우 사용할 수 있습니다.

DFHJS2LS와 기호 매개변수, 입력 매개변수와 설명, 작업 예제에 대한 작업 제어 명령문은 이 프로시저를 사용하는 데 도움을 줍니다.

DFHJS2LS JCL 프로시저는 데이터 세트 *HLQ.XDFHINST*에 설치됩니다. 여기서 *HLQ*는 CICS가 설치된 상위 레벨 규정자입니다.

DFHLS2JS 또는 DFHJS2LS 프로시저의 관련 사용 모드는 다음과 같이 요구사항에 따라 다릅니다.

- DFHLS2JS: 링크 가능 인터페이스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 링크 가능 인터페이스를 위한 JSON 스키마 대 고급 언어 변환
- DFHLS2JS: 요청-응답 서비스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 요청-응답 서비스를 위한 JSON 스키마 대 고급 언어 변환
- DFHJS2LS: RESTful 서비스를 위한 JSON 스키마 대 고급 언어 변환

DFHJS2LS의 작업 제어 명령문

JOB 작업을 시작합니다.

EXEC

프로시저 이름(DFHJS2LS)을 지정합니다.

INPUT.SYSUT1 DD

입력을 지정합니다. 입력 매개변수는 일반적으로 입력 스트림에 지정됩니다. 그러나 데이터 세트 또는 파티션된 데이터 세트의 멤버에 정의될 수 있습니다.

기호 매개변수

DFHJS2LS에서는 다음 기호 매개변수가 정의됩니다.

JAVADIR=*path*

DFHJS2LS에서 사용되는 Java 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 */usr/lpp/*에 추가되어 */usr/lpp/path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

PATHPREF=*prefix*

다른 매개변수 또는 ''(빈 문자열)(접두부를 사용하지 않는 경우)에서 사용되는 z/OS UNIX 디렉토리 경로를 확장하는 접두부를 지정합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **PATHPREF** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

SERVICE=*value*

이 매개변수는 IBM 지원 센터에서 지시하는 경우에만 사용합니다.

TMPDIR=*tmpdir*

DFHJS2LS가 z/OS UNIX에서 임시 작업공간으로 사용하는 디렉토리의 위치를 지정합니다. 작업이 실행되는 사용자 ID는 이 디렉토리에 대한 읽기 및 쓰기 권한이 있어야 합니다.

기본값은 /tmp입니다.

TMPFILE=*tmpprefix*

DFHJS2LS가 임시 작업공간 파일의 이름을 구성하기 위해 사용하는 접두부를 지정합니다.

기본값은 JS2LS입니다.

USSDIR=*path*

UNIX 시스템 서비스 파일 시스템에서 CICS TS 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/cicsts/에 추가되어 /usr/lpp/cicsts/*path*의 전체 경로 이름을 생성합니다. 기본값을 사용하는 경우 '.'(마침표)로 지정되어야 합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **USSDIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

임시 작업공간

DFHJS2LS는 런타임에 다음 세 개 임시 파일을 작성합니다.

tmpdir/tmpprefix.in

tmpdir/tmpprefix.out

tmpdir/tmpprefix.err

여기서,

*tmpdir*은 **TMPDIR** 매개변수에 지정된 값입니다.

*tmpprefix*는 **TMPFILE** 매개변수에 지정된 값입니다.

TMPDIR 및 **TMPFILE**이 지정되지 않은 경우 파일의 기본 이름은 다음과 같습니다.

/tmp/JS2LS.in

/tmp/JS2LS.out

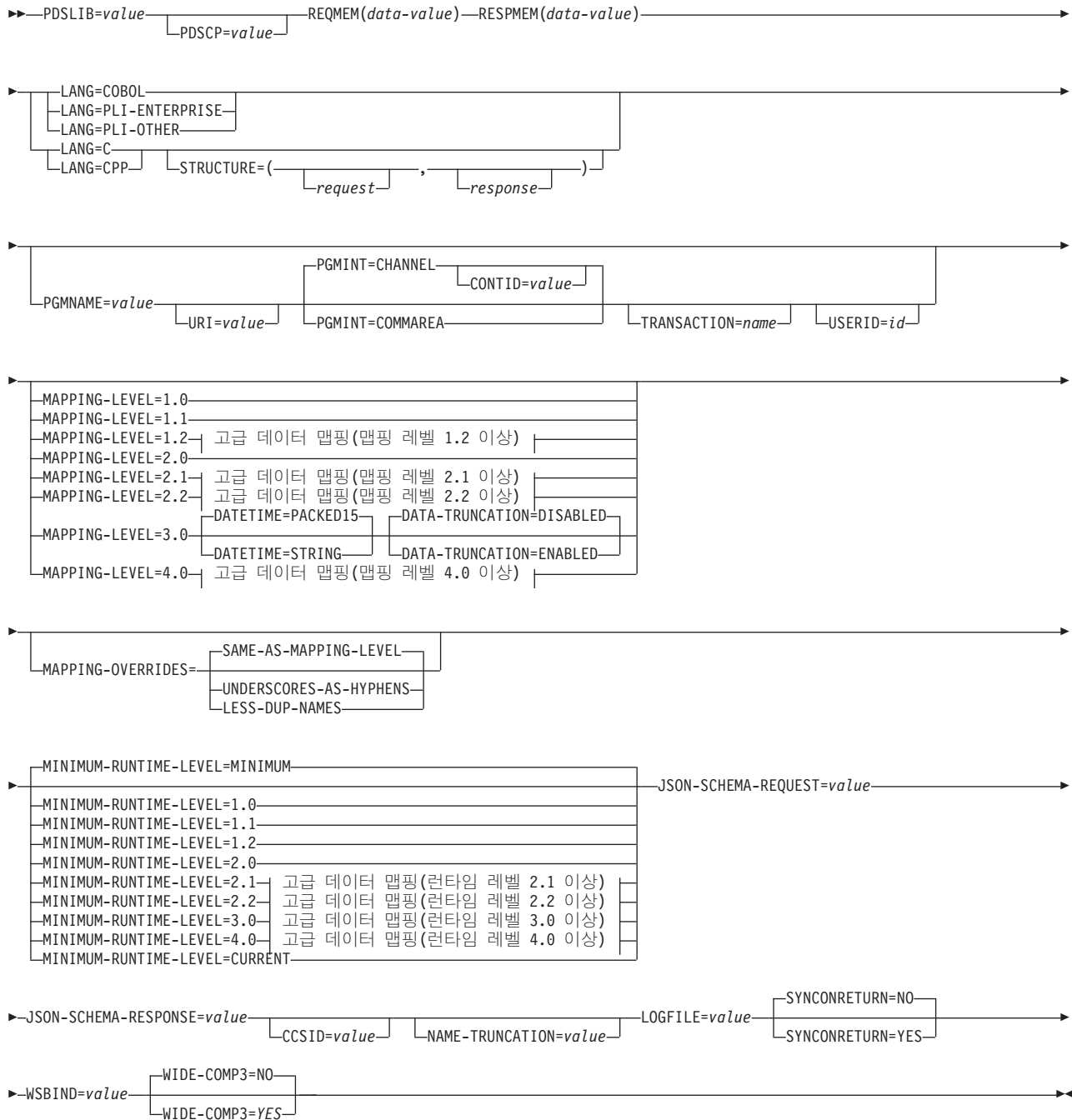
/tmp/JS2LS.err

중요사항: DFHJS2LS는 z/OS UNIX 파일 또는 데이터 세트 멤버에 대한 액세스를 차단하지 않습니다. 따라서 두 개 이상의 DFHJS2LS 인스턴스가 동시에 실행되고 동일한 임시 작업공간 파일을 사용하는 경우, 다른 작업이 작업공간 파일을 사용하는 동안 특정 작업이 해당 파일을 겹쳐쓸 수 있어 예상치 못한 실패를 경험할 수 있습니다.

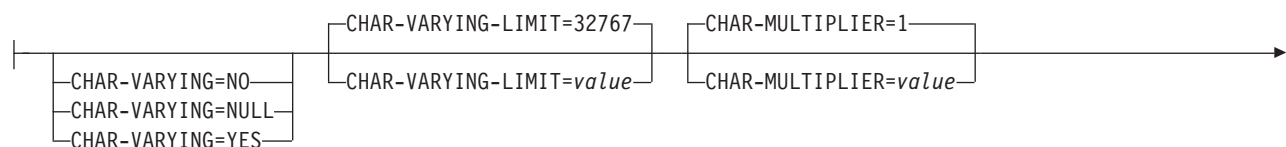
따라서 이러한 상황을 방지하는 이름 지정 규칙과 작동 프로시저를 고안해야 합니다. 예를 들어, 시스템 기호 매개변수 **SYSUID**를 사용하여 개별 사용자에게 고유한 작업공간

파일 이름을 생성할 수 있습니다. 이러한 임시 파일은 작업이 종료되기 전에 삭제됩니다.

DFHJS2LS의 입력 매개변수

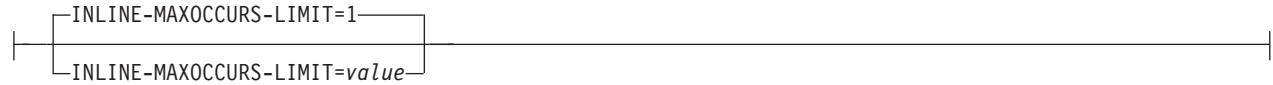


고급 데이터 �핑(맵핑 레벨 1.2 이상):





고급 데이터 맵핑(맵핑 레벨 2.1 이상):



매개변수 사용

- 입력 매개변수는 순서에 관계 없이 지정할 수 있습니다.
- 각 매개변수는 행을 바꿔 시작해야 합니다.
- 매개변수는 사용 시 해당 연속 문자와 함께 열 72를 넘어가서는 안됩니다. 열 73 - 80은 공백이어야 합니다.
- 매개변수가 너무 길어 한 행에 맞지 않는 경우에는 행 끝에서 별표(*) 문자를 사용하여 매개변수가 다음 행에서 연속됨을 표시해야 합니다. 별표 앞의 모든 내용은 공백을 포함하여 매개변수의 일부로 간주됩니다. 예를 들어, 다음과 같습니다.

```
WSBIND=wsbinddir*  
/app1
```

구문은 다음과 같습니다.

```
WSBIND=wsbinddir/app1
```

- 행에서 첫 번째 문자 위치의 # 문자는 주석 문자입니다. 이 행은 무시됩니다.
- 행에서 마지막 문자 위치의 쉼표는 선택적 선 구분 기호이며 무시됩니다.

매개변수 설명

CCSID=value

런타임에 애플리케이션 데이터 구조에서 문자 데이터를 인코딩하는 데 사용되는 CCSID를 지정합니다. 이 매개변수의 값은 **LOCALCCSID** 시스템 초기화 매개변수의 값을 대체합니다. *value*는 Java 및 z/OS Unicode Services User's Guide and Reference에서 지원하는 EBCDIC CCSID여야 합니다. 이 매개변수를 지정하지 않으면 시스템 초기화 매개변수에 지정된 CCSID를 사용하여 애플리케이션 데이터 구조가 인코딩됩니다.

CHAR-MULTIPLIER={1|value}

맵핑 레벨이 1.2 이상인 경우 각 문자에 허용되는 바이트 수를 지정합니다. 이 매개변수의 *value*는 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다. 모든 비숫자 문자 기반 맵핑은 이 승수가 적용됩니다. 2진, 숫자, 존형 및 팩형 10진수 필드는 이 승수의 대상이 아닙니다.

이 매개변수는 예를 들어, DBCS 문자를 사용하려는 경우에 유용할 수 있습니다. 이 경우 런타임 시 모든 2바이트 문자 앞뒤로 잠재적인 SO 및 SI 문자를 위한 공간을 허용하기 위해 승수 3을 선택할 수 있습니다.

CCSID=1200(UTF-16 표시)을 설정하면 **CHAR-MULTIPLIER**에 대한 유일한 올바른 값은 2 또는 4입니다. UTF-16을 사용할 때 기본값은 2입니다. 애플리케이션 데이터가 1개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우 **CHAR-MULTIPLIER=2**를 사용하십시오. 애플리케이션 데이터가 2개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우 **CHAR-MULTIPLIER=4**를 사용하십시오.

참고: **CHAR-MULTIPLIER**를 1로 설정하면 DBCS 문자의 사용이 배제되지 않으며 이를 2로 설정하면 UTF-16 대리 쌍의 사용을 배제하지 않습니다. 그러나 와이드 문자가 주기적으로 사용되면 일부 올바른 값이 할당된 필드에 맞지 않게 됩니다. **CHAR-MULTIPLIER**의 값으로 더 큰 값을 사용할 수록 XML에서 올바른 문자 수보다 더 많은 문자를 할당된 필드에 저장할 수 있습니다. 해당 범위 제한을 주의하여 따라야 합니다.

CHAR-VARYING={NO|NULL|YES}

맵핑 레벨이 1.2 이상인 경우 가변 길이 문자 데이터가 맵핑되는 방식을 지정합니다. 가변 길이 2진 데이터 유형은 항상 컨테이너 또는 가변 구조에 맵핑됩니다. 이 매개변수를 지정하지 않은 경우 지정한 언어에 따라 기본 맵핑이 달라집니다. 다음 옵션을 선택할 수 있습니다.

NO 가변 길이 문자 데이터가 고정 길이 문자열로 맵핑됩니다.

NULL

가변 길이 문자 데이터가 널 종료 문자열로 맵핑됩니다.

YES PL/I의 경우 가변 길이 문자 데이터가 CHAR VARYING 데이터 유형에 맵핑됩니다. COBOL, C, C++ 언어에서는 가변 길이 문자 데이터가 두 가지 관련 요소(데이터-길이 및 데이터)로 구성되는 해당 표시에 맵핑됩니다.

CHAR-VARYING-LIMIT={32767|value}

맵핑 레벨이 1.2 이상인 경우 언어 구조에 맵핑되는 가변 길이 문자 데이터와 2진 데이터의 최대 크기를 지정합니다. 문자 또는 2진 데이터가 이 매개변수에 지정된 값보다 큰 경우에는 컨테이너에 맵핑되고 생성된 언어 구조에서 컨테이너 이름이 사용됩니다. 값의 범위는 0 - 32,767바이트(기본값)입니다.

CONTID=value

서비스 제공자에서, JSON 메시지를 나타내는 데 사용되는 상위 레벨 데이터 구조를 보유하는 컨테이너의 이름을 지정합니다.

CICS가 대상 애플리케이션 프로그램으로 전달하는 컨테이너의 길이는 요청 컨테이너의 길이와 응답 컨테이너의 길이 중 긴 값입니다.

DATA-TRUNCATION={DISABLED|ENABLED}

가변 길이 데이터가 고정 길이 필드 구조에서 용인되는지 여부를 지정합니다.

DISABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 거부하고 오류 메시지를 발행합니다.

ENABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 용인하고 누락된 데이터를 널값으로 처리합니다.

DATETIME={PACKED15|STRING}

JSON 날짜-시간 요소가 언어 구조에 매핑되는 방식을 지정합니다.

PACKED15

기본적으로 모든 JSON 날짜-시간 요소는 시간소인으로 처리되고 CICS ABSTIME 형식에 매핑됩니다.

STRING

JSON 날짜-시간 요소가 텍스트로 처리됩니다.

DEFAULT-CHAR-MAXLENGTH={255|value}

매핑 레벨이 1.2 이상이고 웹 서비스 설명 문서에 길이가 나타나지 않는 경우 매핑 문자 내 문자 데이터의 기본 배열 길이를 지정합니다. 이 매개변수의 값은 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다.

INLINE-MAXOCCURS-LIMIT={1|value}

인라인 가변 반복 콘텐츠가 maxItems JSON 스키마 키워드를 기반으로 사용되는지 여부를 지정합니다. 인라인으로 매핑되는 가변 반복 콘텐츠는 생성된 언어 구조의 나머지와 함께 현재 컨테이너에 배치됩니다. 가변 반복 콘텐츠는 두 파트로, 즉 데이터 발생 회수를 저장하는 카운터와 데이터의 각 발생을 저장하는 배열로 저장됩니다. 가변 반복 콘텐츠의 대체 매핑은 컨테이너 기반 매핑으로, 데이터 발생 횟수와 데이터가 배치되는 컨테이너의 이름을 저장합니다. 개별 컨테이너에 데이터를 저장하는 경우 성능에 영향을 주어 인라인 매핑이 선호될 수 있습니다.

INLINE-MAXOCCURS-LIMIT 매개변수는 매핑 레벨 2.1 이상에서만 사용할 수 있습니다. **INLINE-MAXOCCURS-LIMIT** 값은 0 - 32,767 범위의 양의 정수일 수 있습니다. 값이 0이면 인라인 매핑이 사용되지 않음을 나타냅니다. 값이 1이면 선택적 요소가 인라인으로 매핑됩니다. maxOccurs 속성의 value가 **INLINE-MAXOCCURS-LIMIT**의 value보다 크면 컨테이너 기반 매핑을 사용하고 그렇지 않으면 인라인 매핑을 사용합니다.

가변 반복 목록을 인라인으로 매핑하는지 여부를 결정할 때 반복 데이터의 단일 항목 길이를 고려해야 합니다. 길이가 긴 인스턴스가 거의 없는 경우에는 컨테이너 기반 매핑이 선호됩니다. 길이가 짧은 인스턴스가 많은 경우에는 인라인 매핑이 선호됩니다.

JSON-SCHEMA-REQUEST=*value*

필수 매개변수입니다.

값은 요청 JSON 스키마가 저장되는 UNIX System Services 위치를 나타냅니다.

JSON-SCHEMA-RESPONSE=*value*

필수 매개변수입니다.

값은 응답 JSON 스키마가 저장되는 UNIX System Services 위치를 나타냅니다.

LANG=COBOL

고급 언어 구조의 프로그래밍 언어가 COBOL인 것으로 지정합니다.

LANG=PLI-ENTERPRISE

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I인 것으로 지정합니다.

LANG=PLI-OTHER

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I 이외의 PL/I 레벨인 것으로 지정합니다.

LANG=C

고급 언어 구조의 프로그래밍 언어가 C인 것으로 지정합니다.

LANG=CPP

고급 언어 구조의 프로그래밍 언어가 C++인 것으로 지정합니다.

LOGFILE=*value*

DFHJS2LS가 해당 활동 로그 및 추적 정보를 기록하는 파일의 완전한 z/OS UNIX 이름입니다. DFHJS2LS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다.

이 파일은 일반적으로 사용하지 않지만 DFHJS2LS 관련 문제점이 발생하는 경우 IBM 서비스 조직이 요청할 수 있습니다.

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0}

웹 서비스 바인딩 파일과 언어 구조를 생성할 때 DFHJS2LS가 사용하는 맵핑 레벨을 지정합니다. 다음 옵션을 선택할 수 있습니다.

- 1.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.1** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

- 2.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 3.0 사용 가능한 전체 옵션 세트를 사용하여 JSON 스키마를 생성하려면 이 맵핑 레벨을 사용하십시오.
- 4.0 UTF-16을 사용하려는 경우 CICS TS 5.2 리전에 이 맵핑 레벨을 사용하십시오.

맵핑 레벨에 대한 자세한 정보는 CICS 지원에 대한 맵핑 레벨을 참조하십시오.

MAPPING-OVERRIDES={SAME-AS-MAPPING-LEVEL|UNDERScores-AS-HYPHENS|LESS-DUP-NAMES}

언어 구조를 생성할 때 지정된 맵핑 레벨에 대해 기본 동작이 대체되는지 여부를 지정합니다.

LESS-DUP-NAMES

이 매개변수는 이름 끝에 `_value`가 있는 비구조적 구조 필드 이름을 생성하여 필드를 직접 참조할 수 있습니다. 예를 들어, 다음 PLI 언어 구조에서 `MAPPING-OVERRIDES=LESS-DUP-NAMES`가 지정되는 경우 레벨 12 필드 `streetName`에는 접미부 `_value`가 추가됩니다.

```
09 streetName,
   12 streetName                CHAR(255) VARYING
   UNALIGNED,
   12 filler                    BIT (7),
   12 attr_nil_streetName_value BIT (1),
```

결과적으로 구조는 다음과 같습니다.

```
09 streetName,
   12 streetName_value          CHAR(255) VARYING
   UNALIGNED,
   12 filler                    BIT (7),
   12 attr_nil_streetName_value BIT (1),
```

SAME-AS-MAPPING-LEVEL

이 매개변수는 맵핑 레벨과 동일한 스타일의 언어 구조를 생성합니다. 이는 기본값입니다.

UNDERScores-AS-HYPHENS

COBOL에서만 사용됩니다. 이 매개변수는 JSON 스키마의 밑줄을 문자 `X`가 아닌 하이픈으로 변환하여 생성된 COBOL 언어 구조의 가독성을 향상시킵니다. 필드 이름 충돌이 발생하면 필드가 고유할 수 있도록 필드에 번호가 지정됩니다. 자세한 정보는 개발 애플리케이션에서 COBOL에 대한 JSON 스키마 맵핑을 참조하십시오.

MINIMUM - RUNTIME -

LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0|CURRENT}

웹 서비스 바인딩 파일이 배치될 수 있는 최소 CICS 런타임 환경을 지정합니다. 지정한 다른 매개변수와 일치하지 않는 레벨을 선택하면 오류 메시지를 받게 됩니다. 다음 옵션을 선택할 수 있습니다.

MINIMUM

선택한 매개변수의 경우 CICS의 런타임 레벨 중 가장 낮은 레벨이 자동으로 할당됩니다.

1.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

1.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

1.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

2.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

2.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

2.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

3.0 생성된 웹 서비스 바인딩 파일이 CICS TS 4.1 이상 리전에 배치됩니다.

참고: JSON 지원은 CICS TS 4.2 이상에서만 사용할 수 있습니다.

4.0 생성된 웹 서비스 바인딩 파일이 CICS TS 5.2 이상 리전에 배치됩니다. 이 런타임 레벨에서는 **MAPPING-LEVEL** 매개변수에 4.0 이하 맵핑 레벨을 사용할 수 있습니다. 이 레벨에서는 선택적 매개변수를 사용할 수 있습니다.

CURRENT

생성된 웹 서비스 바인딩 파일이 웹 서비스 바인딩 파일을 생성하기 위해 사용하는 런타임 레벨과 동일한 런타임 레벨에서 CICS 리전에 배치됩니다.

NAME-TRUNCATION={LEFT|RIGHT}

JSON 이름이 왼쪽 또는 오른쪽부터 잘리는지 여부를 지정합니다. CICS 웹 서비스 지원은 지정된 고급 언어에 적합한 길이로 JSON 이름을 자릅니다. 기본적으로 이름은 오른쪽부터 잘립니다.

PDSCP=value

REQMEM 및 RESPMEM 매개변수에 지정된 파티션된 데이터 세트 멤버에서 사용되는

코드 페이지를 지정합니다. 여기서 *value*는 CCSID 번호 또는 Java 코드 페이지 번호입니다. 이 매개변수가 지정되지 않으면 z/OS UNIX System Services 코드 페이지를 사용합니다. 예를 들어, **PDSCP=037**을 지정할 수 있습니다.

PDSLIB=*value*

생성된 고급 언어를 포함하는 파티션된 데이터 세트의 이름을 지정합니다. 요청 및 응답에 사용되는 데이터 세트 멤버는 **REQMEM** 및 **RESPMEM** 매개변수에 지정됩니다.

PGMINT={CHANNEL|COMMAREA}

서비스 제공자의 경우, CICS가 대상 애플리케이션 프로그램으로 데이터를 전달하는 방식을 지정합니다.

CHANNEL

CICS가 채널 인터페이스를 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

COMMAREA

CICS가 통신 영역을 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

대상 애플리케이션 프로그램이 요청을 처리한 경우에는 동일한 메커니즘으로 응답을 리턴해야 합니다. 즉, 요청이 통신 영역에서 수신된 경우에는 응답도 통신 영역에서 리턴되어야 하고 요청이 컨테이너에서 수신된 경우에는 응답도 컨테이너에서 리턴되어야 합니다. CICS가 대상 애플리케이션 프로그램으로 전달하는 통신 영역 또는 컨테이너의 길이는 요청 통신 영역 또는 컨테이너의 길이와 응답 통신 영역 또는 컨테이너의 길이 중 긴 값입니다.

PGMNAME=*value*

CICS PROGRAM 자원의 이름을 지정합니다.

DFHJS2LS가 서비스 제공자에서 사용되는 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우 이 매개변수를 제공해야 합니다. 이 매개변수는 웹 서비스로 노출되는 애플리케이션 프로그램의 자원 이름을 지정합니다.

DFHJS2LS가 서비스 요청자에서 사용되는 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우에는 이 매개변수를 생략하십시오.

REQMEM=*value*

DFHJS2LS가 웹 서비스 요청(애플리케이션 프로그램의 입력 데이터)에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 사용하는 1 - 6자 접두부를 지정합니다.

DFHJS2LS는 접두부에 01을 추가하여 파티션된 데이터 세트 멤버의 이름을 생성합니다.

RESPMEM=value

DFHJS2LS가 웹 서비스 응답(애플리케이션 프로그램의 출력 데이터)에 대한 고급 언어 구조를 포함하는 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 사용하는 1 - 6자 접두부를 지정합니다.

DFHJS2LS는 접두부에 01을 추가하여 파티션된 데이터 세트 멤버의 이름을 생성합니다.

STRUCTURE=(request,response)

C 및 C++의 경우에만, 요청 및 응답 구조의 이름이 생성되는 방식을 지정합니다.

생성된 요청 및 응답 구조에는 이름 *request01* 및 *response01*이 지정됩니다.

두 이름 중 하나 또는 두 이름이 모두 생략되는 경우, 구조 이름은 사용자가 지정하는 **REQMEM** 및 **RESPMEM** 매개변수에서 생성된 파티션된 데이터 세트 멤버 이름과 같습니다.

SYNCONRETURN={NO|YES}

원격 웹 서비스가 동기점을 발행할 수 있는지 여부를 지정합니다.

NO 원격 웹 서비스가 동기점을 발행할 수 없습니다. 이 값은 기본값입니다. 원격 웹 서비스가 동기점을 발행하면 ADPL 이상 종료와 함께 실패합니다.

YES 원격 웹 서비스가 동기점을 발행할 수 있습니다. YES를 선택하면 원격 웹 서비스에서 제어가 리턴되는 경우 원격 태스크가 개별 작업 단위로 커밋됩니다. 원격 웹 서비스가 복구 가능 자원을 갱신하고 리턴 후 실패하면 해당 자원에 대한 갱신을 백아웃할 수 없습니다.

TRANSACTION=name

서비스 제공자에서 이 매개변수는 파이프라인을 시작할 수 있는 별명 트랜잭션의 1 - 4자 이름을 지정합니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 TRANSACTION 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ # _ < >

URI=value

서비스 제공자에서, 이 매개변수는 클라이언트가 웹 서비스에 액세스하기 위해 사용하는 상대 URI를 지정합니다. CICS는 DFHJS2LS로 작성된 웹 서비스 바인딩 파일에서 URIMAP 자원을 생성할 때 지정되는 값을 사용합니다. 이 매개변수는 URIMAP 정의가 적용되는 URI의 경로 컴포넌트를 지정합니다.

USERID=id

서비스 제공자에서 이 매개변수는 웹 클라이언트가 사용할 수 있는 1 - 8자 사용자 ID를 지정합니다. 애플리케이션 생성 응답 또는 웹 서비스의 경우 이 사용자 ID

아래에 별명 트랜잭션이 첨부됩니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 때 **URIMAP** 자원의 **USERID** 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3={NO|YES}

COBOL에서만 사용됩니다. COBOL 언어 구조에서 팩형 10진수 가변 길이의 최대 크기를 제어합니다.

NO DFHJS2LS가 COBOL 언어 구조 유형 COMP-3을 생성할 때 팩형 10진수 가변 길이를 18로 제한합니다. 팩형 10진수 크기가 18보다 큰 경우, 지정된 유형이 총 18자로 제한됨을 나타내는 DFHP19022W 메시지가 발행됩니다.

YES DFHJS2LS가 COBOL 언어 구조 유형 COMP-3을 생성할 때 최대 크기 31을 지원합니다.

WSBIND=value

웹 서비스 바인딩 파일의 완전한 z/OS UNIX 이름입니다. DFHJS2LS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 기본 파일 확장자는 .wsbind입니다.

기타 정보

- DFHJS2LS가 실행되는 사용자 ID는 UNIX System Services를 사용하도록 구성되어야 합니다. 사용자 ID는 CICS z/OS UNIX 파일 구조 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND**, **WSDL** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다.
- JCL의 최대 매개변수 길이는 100자입니다. 이 길이는 **STDPARM**문을 사용하여 늘릴 수 있습니다. 자세한 정보는 *z/OS UNIX System Services User Guide*를 참조하십시오.

예제

```
//JS2LS JOB 'accounting information',name,MSGCLASS=A
// SET QT='''
//JAVAPROG EXEC DFHJS2LS,
// TMPFILE=&QT.&SYSUID.&QT
/INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
REQMEM=CPYBK1
RESPMEM=CPYBK2
JSON-SCHEMA-REQUEST=example.json
JSON-SCHEMA-RESPONSE=example.json
LANG=COBOL
```

```

LOGFILE=/u/exampleapp/wsbind/example.log
MAPPING-LEVEL=4.0
CHAR-VARYING=NULL
INLINE-MAXOCCURS-LIMIT=2
PGMNAME=DFH0XCMN
URI=exampleApp/example
PGMINT=COMMAREA
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbind/example.wsbind
/*

```

DFHJS2LS: RESTful 서비스를 위한 JSON 스키마 대 고급 언어 변환

DFHJS2LS 프로시저는 JSON 스키마에서 고급 언어 데이터 구조와 웹 서비스 바인딩 파일을 생성합니다. DFHJS2LS는 RESTful JSON 서비스 제공자 작성을 준비할 때 사용할 수 있습니다.

DFHJS2LS와 기호 매개변수, 입력 매개변수와 설명, 작업 예제에 대한 작업 제어 명령문은 이 프로시저를 사용하는 데 도움을 줍니다.

DFHJS2LS JCL 프로시저는 데이터 세트 *HLQ.XDFHINST*에 설치됩니다. 여기서 *HLQ*는 CICS가 설치된 상위 레벨 규정자입니다.

DFHLS2JS 또는 DFHJS2LS 프로시저의 관련 사용 모드는 다음과 같이 요구사항에 따라 다릅니다.

- DFHLS2JS: 링크 가능 인터페이스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 링크 가능 인터페이스를 위한 JSON 스키마 대 고급 언어 변환
- DFHLS2JS: 요청-응답 서비스를 위한 고급 언어 대 JSON 스키마 변환
- DFHJS2LS: 요청-응답 서비스를 위한 JSON 스키마 대 고급 언어 변환
- DFHJS2LS: RESTful 서비스를 위한 JSON 스키마 대 고급 언어 변환

DFHJS2LS의 작업 제어 명령문

JOB 작업을 시작합니다.

EXEC

프로시저 이름(DFHJS2LS)을 지정합니다.

INPUT.SYSUT1 DD

입력을 지정합니다. 입력 매개변수는 일반적으로 입력 스트림에 지정됩니다. 그러나 데이터 세트 또는 파티션된 데이터 세트의 멤버에 정의될 수 있습니다.

기호 매개변수

DFHJS2LS에서는 다음 기호 매개변수가 정의됩니다.

JAVADIR=*path*

DFHJS2LS에서 사용되는 Java 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/에 추가되어 /usr/lpp/*path*의 전체 경로 이름을 생성합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **JAVADIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

PATHPREF=*prefix*

다른 매개변수 또는 ''(빈 문자열)(접두부를 사용하지 않는 경우)에서 사용되는 z/OS UNIX 디렉토리 경로를 확장하는 접두부를 지정합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **PATHPREF** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

SERVICE=*value*

이 매개변수는 IBM 지원 센터에서 지시하는 경우에만 사용합니다.

TMPDIR=*tmpdir*

DFHJS2LS가 z/OS UNIX에서 임시 작업공간으로 사용하는 디렉토리의 위치를 지정합니다. 작업이 실행되는 사용자 ID는 이 디렉토리에 대한 읽기 및 쓰기 권한이 있어야 합니다.

기본값은 /tmp입니다.

TMPFILE=*tmpprefix*

DFHJS2LS가 임시 작업공간 파일의 이름을 구성하기 위해 사용하는 접두부를 지정합니다.

기본값은 JS2LS입니다.

USSDIR=*path*

UNIX 시스템 서비스 파일 시스템에서 CICS TS 디렉토리의 이름을 지정합니다. 이 매개변수의 값은 /usr/lpp/cicsts/에 추가되어 /usr/lpp/cicsts/*path*의 전체 경로 이름을 생성합니다. 기본값을 사용하는 경우 '.'(마침표)로 지정되어야 합니다.

일반적으로 이 매개변수는 지정하지 않습니다. 기본값은 **USSDIR** 매개변수에서 CICS 설치 작업(DFHISTAR)에 제공된 값입니다.

임시 작업공간

DFHJS2LS는 런타임에 다음 세 개 임시 파일을 작성합니다.

tmpdir/tmpprefix.in

tmpdir/tmpprefix.out

tmpdir/tmpprefix.err

여기서,

*tmpdir*은 **TMPDIR** 매개변수에 지정된 값입니다.

*tmpprefix*는 **TMPFILE** 매개변수에 지정된 값입니다.

TMPDIR 및 **TMPFILE**이 지정되지 않은 경우 파일의 기본 이름은 다음과 같습니다.

/tmp/JS2LS.in

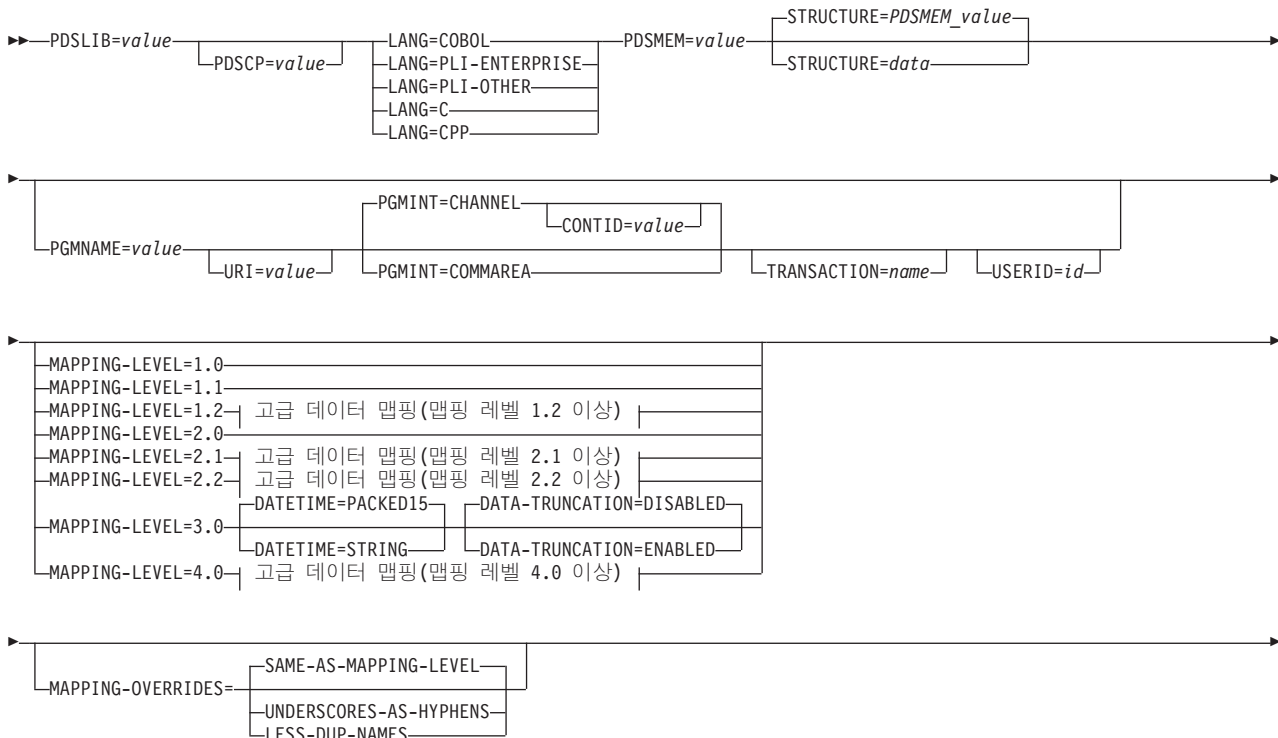
/tmp/JS2LS.out

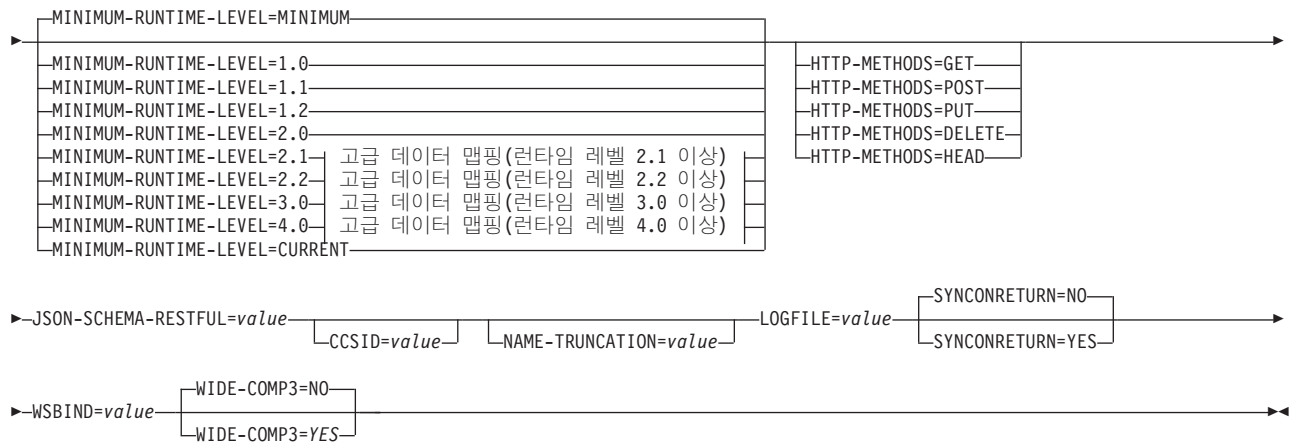
/tmp/JS2LS.err

중요사항: DFHJS2LS는 z/OS UNIX 파일 또는 데이터 세트 멤버에 대한 액세스를 차단하지 않습니다. 따라서 두 개 이상의 DFHJS2LS 인스턴스가 동시에 실행되고 동일한 임시 작업공간 파일을 사용하는 경우, 다른 작업이 작업공간 파일을 사용하는 동안 특정 작업이 해당 파일을 겹쳐쓸 수 있어 예상치 못한 실패를 경험할 수 있습니다.

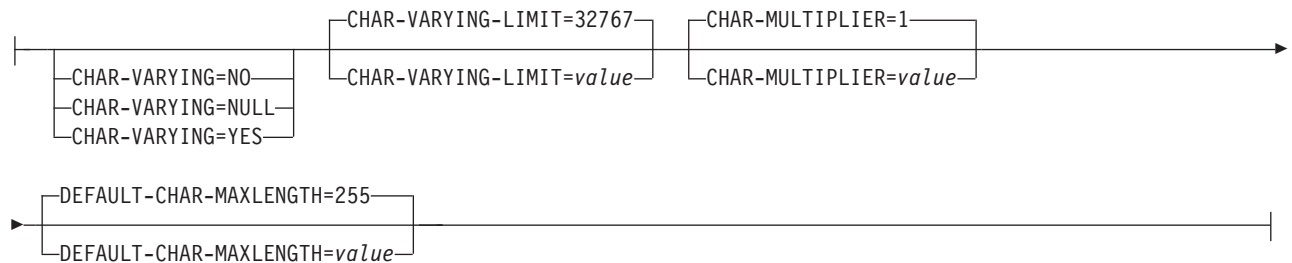
따라서 이러한 상황을 방지하는 이름 지정 규칙과 작동 프로시저를 고안해야 합니다. 예를 들어, 시스템 기호 매개변수 **SYSUID**를 사용하여 개별 사용자에게 고유한 작업공간 파일 이름을 생성할 수 있습니다. 이러한 임시 파일은 작업이 종료되기 전에 삭제됩니다.

DFHJS2LS의 입력 매개변수





고급 데이터 맵핑(맵핑 레벨 1.2 이상):



고급 데이터 맵핑(맵핑 레벨 2.1 이상):



매개변수 사용

- 입력 매개변수는 순서에 관계 없이 지정할 수 있습니다.
- 각 매개변수는 행을 바꿔 시작해야 합니다.
- 매개변수는 사용 시 해당 연속 문자와 함께 열 72를 넘어가서는 안됩니다. 열 73 - 80은 공백이어야 합니다.
- 매개변수가 너무 길어 한 행에 맞지 않는 경우에는 행 끝에서 별표(*) 문자를 사용하여 매개변수가 다음 행에서 연속됨을 표시해야 합니다. 별표 앞의 모든 내용은 공백을 포함하여 매개변수의 일부로 간주됩니다. 예를 들어, 다음과 같습니다.

```
WSBIND=wsbinddir*
/app1
```

구문은 다음과 같습니다.

```
WSBIND=wsbinddir/app1
```

- 행에서 첫 번째 문자 위치의 # 문자는 주석 문자입니다. 이 행은 무시됩니다.

- 행에서 마지막 문자 위치의 쉼표는 선택적 선 구분 기호이며 무시됩니다.

매개변수 설명

CCSID=*value*

런타임에 애플리케이션 데이터 구조에서 문자 데이터를 인코딩하는 데 사용되는 CCSID를 지정합니다. 이 매개변수의 값은 **LOCALCCSID** 시스템 초기화 매개변수의 값을 대체합니다. *value*는 Java 및 z/OS Unicode Services User's Guide and Reference에서 지원하는 EBCDIC CCSID여야 합니다. 이 매개변수를 지정하지 않으면 시스템 초기화 매개변수에 지정된 CCSID를 사용하여 애플리케이션 데이터 구조가 인코딩됩니다.

CHAR-MULTIPLIER={1|*value*}

맵핑 레벨이 1.2 이상인 경우 각 문자에 허용되는 바이트 수를 지정합니다. 이 매개변수의 *value*는 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다. 모든 비숫자 문자 기반 맵핑은 이 승수가 적용됩니다. 2진, 숫자, 존형 및 팩형 10진수 필드는 이 승수의 대상이 아닙니다.

이 매개변수는 예를 들어, DBCS 문자를 사용하려는 경우에 유용할 수 있습니다. 이 경우 런타임 시 모든 2바이트 문자 앞뒤로 잠재적인 SO 및 SI 문자를 위한 공간을 허용하기 위해 승수 3을 선택할 수 있습니다.

CCSID=1200(UTF-16 표시)을 설정하면 **CHAR-MULTIPLIER**에 대한 유일한 올바른 값은 2 또는 4입니다. UTF-16을 사용할 때 기본값은 2입니다. 애플리케이션 데이터가 1개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우 **CHAR-MULTIPLIER=2**를 사용하십시오. 애플리케이션 데이터가 2개의 UTF-16 인코딩 단위를 요구하는 문자를 포함할 것으로 예상되는 경우 **CHAR-MULTIPLIER=4**를 사용하십시오.

참고: **CHAR-MULTIPLIER**를 1로 설정하면 DBCS 문자의 사용이 배제되지 않으며 이를 2로 설정하면 UTF-16 대리 쌍의 사용을 배제하지 않습니다. 그러나 와이드 문자가 주기적으로 사용되면 일부 올바른 값이 할당된 필드에 맞지 않게 됩니다. **CHAR-MULTIPLIER**의 값으로 더 큰 값을 사용할 수록 XML에서 올바른 문자 수보다 더 많은 문자를 할당된 필드에 저장할 수 있습니다. 해당 범위 제한을 주의하여 따라야 합니다.

CHAR-VARYING=NO|NULL|YES

맵핑 레벨이 1.2 이상인 경우 가변 길이 문자 데이터가 맵핑되는 방식을 지정합니다. 가변 길이 2진 데이터 유형은 항상 컨테이너 또는 가변 구조에 맵핑됩니다. 이 매개변수를 지정하지 않은 경우 지정한 언어에 따라 기본 맵핑이 달라집니다. 다음 옵션을 선택할 수 있습니다.

NO 가변 길이 문자 데이터가 고정 길이 문자열로 맵핑됩니다.

NULL

가변 길이 문자 데이터가 널 종료 문자열로 맵핑됩니다.

YES PL/I의 경우 가변 길이 문자 데이터가 CHAR VARYING 데이터 유형에 맵핑됩니다. COBOL, C, C++ 언어에서는 가변 길이 문자 데이터가 두 가지 관련 요소(데이터-길이 및 데이터)로 구성되는 해당 표시에 맵핑됩니다.

CHAR-VARYING-LIMIT=32767|value

맵핑 레벨이 1.2 이상인 경우 언어 구조에 맵핑되는 가변 길이 문자 데이터와 2진 데이터의 최대 크기를 지정합니다. 문자 또는 2진 데이터가 이 매개변수에 지정된 값보다 큰 경우에는 컨테이너에 맵핑되고 생성된 언어 구조에서 컨테이너 이름이 사용됩니다. 값의 범위는 0 - 32,767바이트(기본값)입니다.

CONTID=value

서비스 제공자에서, JSON 메시지를 나타내는 데 사용되는 상위 레벨 데이터 구조를 보유하는 컨테이너의 이름을 지정합니다.

CICS가 대상 애플리케이션 프로그램으로 전달하는 컨테이너의 길이는 요청 컨테이너의 길이와 응답 컨테이너의 길이 중 긴 값입니다.

DATA-TRUNCATION={DISABLED|ENABLED}

가변 길이 데이터가 고정 길이 필드 구조에서 용인되는지 여부를 지정합니다.

DISABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 거부하고 오류 메시지를 발행합니다.

ENABLED

데이터가 CICS가 예상하는 고정 길이보다 짧으면 CICS가 잘린 데이터를 용인하고 누락된 데이터를 널값으로 처리합니다.

DATETIME=PACKED15|STRING

JSON 날짜-시간 요소가 언어 구조에 맵핑되는 방식을 지정합니다.

PACKED15

기본적으로 모든 JSON 날짜-시간 요소는 시간소인으로 처리되고 CICS ABSTIME 형식에 맵핑됩니다.

STRING

JSON 날짜-시간 요소가 텍스트로 처리됩니다.

DEFAULT-CHAR-MAXLENGTH=255|value

맵핑 레벨이 1.2 이상이고 웹 서비스 설명 문서에 길이가 나타나지 않는 경우 맵핑 문자 내 문자 데이터의 기본 배열 길이를 지정합니다. 이 매개변수의 값은 1 - 2,147,483,647 범위의 양의 정수일 수 있습니다.

HTTP-METHODS={GET|POST|PUT|DELETE|HEAD},{GET|POST|PUT|DELETE|HEAD},*
선택적 매개변수입니다.

기본값은 GET, POST, PUT, DELETE를 설정하는 것입니다. 이는 애플리케이션이 네 가지 기본 RESTful 조작을 지원함을 DFHJS2LS에 알려줍니다.

값이 제공되는 경우, DFHJS2LS는 명시적으로 지정된 HTTP 메소드만 허용되는 WSBInd 파일을 빌드합니다.

애플리케이션이 HEAD 메소드를 구현하려는 경우에는 의도적으로 선택해야 합니다. 기본적으로 DFHJS2LS는 애플리케이션이 수신 HTTP HEAD 메소드를 지원하지 않는 것으로 가정합니다.

JSON 클라이언트가 지원되지 않는 HTTP 메소드를 사용하려고 시도하면 CICS가 HTTP 405 응답으로 응답합니다.

INLINE-MAXOCCURS-LIMIT=1|value

인라인 가변 반복 콘텐츠가 maxItems JSON 스키마 키워드를 기반으로 사용되는지 여부를 지정합니다. 인라인으로 맵핑되는 가변 반복 콘텐츠는 생성된 언어 구조의 나머지와 함께 현재 컨테이너에 배치됩니다. 가변 반복 콘텐츠는 두 파트로, 즉 데이터 발생 회수를 저장하는 카운터와 데이터의 각 발생을 저장하는 배열로 저장됩니다. 가변 반복 콘텐츠의 대체 맵핑은 컨테이너 기반 맵핑으로, 데이터 발생 횟수와 데이터가 배치되는 컨테이너의 이름을 저장합니다. 개별 컨테이너에 데이터를 저장하는 경우 성능에 영향을 주어 인라인 맵핑이 선호될 수 있습니다.

INLINE-MAXOCCURS-LIMIT 매개변수는 맵핑 레벨 2.1 이상에서만 사용할 수 있습니다. **INLINE-MAXOCCURS-LIMIT** 값은 0 - 32,767 범위의 양의 정수일 수 있습니다. 값이 0이면 인라인 맵핑이 사용되지 않음을 나타냅니다. 값이 1이면 선택적 요소가 인라인으로 맵핑됩니다. maxOccurs 속성의 value가

INLINE-MAXOCCURS-LIMIT의 value보다 크면 컨테이너 기반 맵핑을 사용하고 그렇지 않으면 인라인 맵핑을 사용합니다.

가변 반복 목록을 인라인으로 맵핑하는지 여부를 결정할 때 반복 데이터의 단일 항목 길이를 고려해야 합니다. 길이가 긴 인스턴스가 거의 없는 경우에는 컨테이너 기반 맵핑이 선호됩니다. 길이가 짧은 인스턴스가 많은 경우에는 인라인 맵핑이 선호됩니다.

JSON-SCHEMA-RESTFUL=value

필수 매개변수입니다.

값은 응답 JSON 스키마가 저장되는 UNIX System Services 위치를 나타냅니다.

LANG=COBOL

고급 언어 구조의 프로그래밍 언어가 COBOL인 것으로 지정합니다.

LANG=PLI-ENTERPRISE

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I인 것으로 지정합니다.

LANG=PLI-OTHER

고급 언어 구조의 프로그래밍 언어가 엔터프라이즈 PL/I 이외의 PL/I 레벨인 것으로 지정합니다.

LANG=C

고급 언어 구조의 프로그래밍 언어가 C인 것으로 지정합니다.

LANG=CPP

고급 언어 구조의 프로그래밍 언어가 C++인 것으로 지정합니다.

LOGFILE=value

DFHJS2LS가 해당 활동 로그 및 추적 정보를 기록하는 파일의 완전한 z/OS UNIX 이름입니다. DFHJS2LS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다.

이 파일은 일반적으로 사용하지 않지만 DFHJS2LS 관련 문제점이 발생하는 경우 IBM 서비스 조직이 요청할 수 있습니다.

MAPPING-LEVEL={1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0}

웹 서비스 바인딩 파일과 언어 구조를 생성할 때 DFHJS2LS가 사용하는 �핑 레벨을 지정합니다. 다음 옵션을 선택할 수 있습니다.

- 1.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.1** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.1** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.2** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 3.0** 사용 가능한 전체 옵션 세트를 사용하여 JSON 스키마를 생성하려면 이 �핑 레벨을 사용하십시오.
- 4.0** UTF-16을 사용하려는 경우 CICS TS 5.2 리전에 이 �핑 레벨을 사용하십시오.

�핑 레벨에 대한 자세한 정보는 애플리케이션 개발에서 CICS JSON 지원자에 대한 �핑 레벨을 참조하십시오.

MAPPING-OVERRIDES={SAME-AS-MAPPING-LEVEL|UNDERSCORES-AS-HYPHENS|LESS-DUP-NAMES}

언어 구조를 생성할 때 지정된 �핑 레벨에 대해 기본 동작이 대체되는지 여부를 지정합니다.

LESS-DUP-NAMES

이 매개변수는 이름 끝에 `_value`가 있는 비구조적 구조 필드 이름을 생성하여 필드를 직접 참조할 수 있습니다. 예를 들어, 다음 PLI 언어 구조에서 `MAPPING-OVERRIDES=LESS-DUP-NAMES`가 지정되는 경우 레벨 12 필드 `streetName`에는 접미부 `_value`가 추가됩니다.

```
09 streetName,
   12 streetName                                CHAR(255) VARYING
   UNALIGNED,
   12 filler                                    BIT (7),
   12 attr_nil_streetName_value                BIT (1),
```

결과적으로 구조는 다음과 같습니다.

```
09 streetName,
   12 streetName_value                          CHAR(255) VARYING
   UNALIGNED,
   12 filler                                    BIT (7),
   12 attr_nil_streetName_value                BIT (1),
```

SAME-AS-MAPPING-LEVEL

이 매개변수는 맵핑 레벨과 동일한 스타일의 언어 구조를 생성합니다. 이는 기본값입니다.

UNDERSCORES-AS-HYPHENS

COBOL에서만 사용됩니다. 이 매개변수는 JSON 스키마의 밑줄을 문자 `X`가 아닌 하이픈으로 변환하여 생성된 COBOL 언어 구조의 가독성을 향상시킵니다. 필드 이름 충돌이 발생하면 필드가 고유할 수 있도록 필드에 번호가 지정됩니다. 자세한 정보는 개발 애플리케이션에서 COBOL에 대한 JSON 스키마 매핑을 참조하십시오.

MINIMUM - RUNTIME -

LEVEL={MINIMUM|1.0|1.1|1.2|2.0|2.1|2.2|3.0|4.0|CURRENT}

웹 서비스 바인딩 파일이 배치될 수 있는 최소 CICS 런타임 환경을 지정합니다. 지정한 다른 매개변수와 일치하지 않는 레벨을 선택하면 오류 메시지를 받게 됩니다. 다음 옵션을 선택할 수 있습니다.

MINIMUM

선택한 매개변수의 경우 CICS의 런타임 레벨 중 가장 낮은 레벨이 자동으로 할당됩니다.

- 1.0** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.1** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 1.2** 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.

- 2.0 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.1 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 2.2 이 옵션은 SOAP 웹 서비스와의 호환성을 위해 유지됩니다. JSON에 대한 용도로는 권장되지 않습니다.
- 3.0 생성된 웹 서비스 바인딩 파일이 CICS TS 4.1 이상 리전에 배치됩니다.

참고: JSON 지원은 CICS TS 4.2 이상에서만 사용할 수 있습니다.

- 4.0 생성된 웹 서비스 바인딩 파일이 CICS TS 5.2 이상 리전에 배치됩니다. 이 런타임 레벨에서는 **MAPPING-LEVEL** 매개변수에 4.0 이하 맵핑 레벨을 사용할 수 있습니다. 이 레벨에서는 선택적 매개변수를 사용할 수 있습니다.

CURRENT

생성된 웹 서비스 바인딩 파일이 웹 서비스 바인딩 파일을 생성하기 위해 사용하는 런타임 레벨과 동일한 런타임 레벨에서 CICS 리전에 배치됩니다.

NAME-TRUNCATION={LEFT|RIGHT}

JSON 이름이 왼쪽 또는 오른쪽부터 잘리는지 여부를 지정합니다. CICS 웹 서비스 지원은 지정된 고급 언어에 적합한 길이로 JSON 이름을 자릅니다. 기본적으로 이름은 오른쪽부터 잘립니다.

PDSCP=value

PDSMEM 매개변수에 지정된 파티션된 데이터 세트 멤버에서 사용되는 코드를 지정합니다. 여기서 *value*는 CCSID 번호 또는 Java 코드 페이지 번호입니다. 이 매개변수가 지정되지 않으면 z/OS UNIX System Services 코드 페이지를 사용합니다. 예를 들어, **PDSCP=037**을 지정할 수 있습니다.

PDSLIB=value

생성된 고급 언어를 포함하는 파티션된 데이터 세트의 이름을 지정합니다. 요청 및 응답에 사용되는 데이터 세트 멤버는 **PDSMEM** 매개변수에 지정됩니다.

PDSMEM=value

DFHJS2LS가 고급 언어 구조를 포함할 파티션된 데이터 세트 멤버의 이름을 생성하기 위해 사용하는 1 - 6자 접두부를 지정합니다.

DFHJS2LS는 접두부에 01을 추가하여 파티션된 데이터 세트 멤버를 생성합니다.

PGMINT=CHANNEL|COMMAREA

서비스 제공자의 경우, CICS가 대상 애플리케이션 프로그램으로 데이터를 전달하는 방식을 지정합니다.

CHANNEL

CICS가 채널 인터페이스를 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

COMMAREA

CICS가 통신 영역을 사용하여 대상 애플리케이션 프로그램에 데이터를 전달합니다.

대상 애플리케이션 프로그램이 요청을 처리한 경우에는 동일한 메커니즘으로 응답을 리턴해야 합니다. 즉, 요청이 통신 영역에서 수신된 경우에는 응답도 통신 영역에서 리턴되어야 하고 요청이 컨테이너에서 수신된 경우에는 응답도 컨테이너에서 리턴되어야 합니다. CICS가 대상 애플리케이션 프로그램으로 전달하는 통신 영역 또는 컨테이너의 길이는 요청 통신 영역 또는 컨테이너의 길이와 응답 통신 영역 또는 컨테이너의 길이 중 긴 값입니다.

PGMNAME=value

CICS PROGRAM 자원의 이름을 지정합니다.

DFHJS2LS가 서비스 제공자에서 사용되는 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우 이 매개변수를 제공해야 합니다. 이 매개변수는 웹 서비스로 노출되는 애플리케이션 프로그램의 자원 이름을 지정합니다.

DFHJS2LS가 서비스 요청자에서 사용되는 웹 서비스 바인딩 파일을 생성하는 데 사용되는 경우에는 이 매개변수를 생략하십시오.

STRUCTURE=name

C 및 C++의 경우에만 구조 이름이 생성되는 방식을 지정합니다.

생성된 구조에는 이름 *name01* 이 지정됩니다.

이름이 생략되는 경우, 구조 이름은 사용자가 지정하는 **PDSMEM** 매개변수에서 생성되는 파티션된 데이터 세트 멤버 이름과 같습니다.

SYNCONRETURN={NO|YES}

원격 웹 서비스가 동기점을 발행할 수 있는지 여부를 지정합니다.

NO 원격 웹 서비스가 동기점을 발행할 수 없습니다. 이 값은 기본값입니다. 원격 웹 서비스가 동기점을 발행하면 ADPL 이상 종료와 함께 실패합니다.

YES 원격 웹 서비스가 동기점을 발행할 수 있습니다. YES를 선택하면 원격 웹 서비스에서 제어가 리턴되는 경우 원격 태스크가 개별 작업 단위로 커밋됩니다. 원격 웹 서비스가 복구 기능 자원을 갱신하고 리턴 후 실패하면 해당 자원에 대한 갱신을 백아웃할 수 없습니다.

TRANSACTION=name

서비스 제공자에서 이 매개변수는 파이프라인을 시작할 수 있는 별명 트랜잭션의 1 - 4자 이름을 지정합니다. 이 매개변수의 값은 **PIPELINE** 스캔 명령을 사용하여 자동으로 작성된 URIMAP 자원의 TRANSACTION 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ # _ < >

URI=value

서비스 제공자에서, 이 매개변수는 클라이언트가 웹 서비스에 액세스하기 위해 사용하는 상대 URI를 지정합니다. CICS는 DFHJS2LS로 작성된 웹 서비스 바인딩 파일에서 URIMAP 자원을 생성할 때 지정되는 값을 사용합니다. 이 매개변수는 URIMAP 정의가 적용되는 URI의 경로 컴포넌트를 지정합니다. URI 끝에 와일드카드 *를 사용하는 경우에는 URI 값 뒤에 쉼표가 와야 합니다.

USERID=id

서비스 제공자에서 이 매개변수는 웹 클라이언트가 사용할 수 있는 1 - 8자 사용자 ID를 지정합니다. 애플리케이션 생성 응답 또는 웹 서비스의 경우 이 사용자 ID 아래에 별명 트랜잭션이 첨부됩니다. 이 매개변수의 값은 PIPELINE 스캔 명령을 사용하여 자동으로 작성된 때 URIMAP 자원의 USERID 속성을 정의하는 데 사용됩니다.

허용 문자:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3=NO|YES

COBOL에서만 사용됩니다. COBOL 언어 구조에서 팩형 10진수 가변 길이의 최대 크기를 제어합니다.

NO DFHJS2LS가 COBOL 언어 구조 유형 COMP-3을 생성할 때 팩형 10진수 가변 길이를 18로 제한합니다. 팩형 10진수 크기가 18보다 큰 경우, 지정된 유형이 총 18자로 제한됨을 나타내는 DFHP19022W 메시지가 발행됩니다.

YES DFHJS2LS가 COBOL 언어 구조 유형 COMP-3을 생성할 때 최대 크기 31을 지원합니다.

WSBIND=value

웹 서비스 바인딩 파일의 완전한 z/OS UNIX 이름입니다. DFHJS2LS가 파일을 작성하지만(존재하지 않는 경우) 디렉토리 구조는 작성하지 않습니다. 기본 파일 확장자는 .wsbind입니다.

기타 정보

- DFHJS2LS가 실행되는 사용자 ID는 UNIX System Services를 사용하도록 구성되어야 합니다. 사용자 ID는 CICS z/OS UNIX 파일 구조 및 PDS 라이브러리에 대한 읽기 권한과 LOGFILE, WSBIND, WSDL 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다.

- JCL의 최대 매개변수 길이는 100자입니다. 이 길이는 **STDPARM**문을 사용하여 늘릴 수 있습니다. 자세한 정보는 *z/OS UNIX System Services User Guide*를 참조하십시오.

예제

```
//JS2LS JOB 'accounting information',name,MSGCLASS=A
// SET QT='''
//JAVAPROG EXEC DFHJS2LS,
// TMPFILE=&QT.&SYSUID.&QT
/INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
PDSMEM=CPYBK2
JSON-SCHEMA-RESTFUL=example.json
LANG=COBOL
LOGFILE=/u/exampleapp/wsbind/example.log
MAPPING-LEVEL=4.0
CHAR-VARYING=NULL
INLINE-MAXOCCURS-LIMIT=2
PGMNAME=DFH0XCMN
URI=exampleApp/example/*,
PGMINT=COMMAREA
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbind/example.wsbind
/*
```

JSON 지원을 사용하여 JSON 서비스 제공자 애플리케이션 작성

JSON 스키마 v4(초안)를 준수하는 JSON 스키마 또는 고급 언어 데이터 구조에서 서비스 제공자 애플리케이션을 작성할 수 있습니다. CICS JSON 지원은 서비스 제공자 설정에서 CICS 애플리케이션을 배치하는 데 도움을 줍니다.

이 태스크 정보

지원을 사용하여 CICS 애플리케이션을 서비스 제공자로 배치할 때 다음 두 가지 옵션이 있습니다.

- JSON 스키마를 시작하고 지원을 사용하여 언어 데이터 구조를 생성합니다.

기존 웹 서비스 설명에 부합되는 서비스 제공자를 구현하려는 경우에는 이 옵션을 사용하십시오.

- 언어 데이터 구조를 시작하고 지원을 사용하여 JSON 스키마를 생성합니다.

기존 프로그램을 JSON 서비스로 노출시키려는 경우에는 이 옵션을 사용하십시오.

관련 정보:

 JSON 스키마 v4(초안)

JSON 스키마에서 서비스 제공자 애플리케이션 작성

CICS JSON 지원을 사용하여 JSON 스키마로부터 서비스 제공자 애플리케이션을 작성할 수 있습니다.

시작하기 전에

서비스 제공자 애플리케이션을 작성하려면 먼저 다음 조건이 만족되어야 합니다.

- 웹 서비스 설명이 z/OS의 UNIX 파일에 있어야 하고 CICS 리전에서 적절한 제공자 모드 파이프라인을 작성해야 합니다.
- OMVS에 DFHJS2LS를 실행하는 사용자 ID를 정의해야 합니다.
- 사용자 ID는 z/OS UNIX 및 PDS 라이브러리에 대한 읽기 권한과 **LOGFILE**, **WSBIND** 및 **JSON-SCHEMA-REQUEST**와 **JSON-SCHEMA-RESPONSE** 및 **JSON-SCHEMA-RESTFUL** 매개변수에 지정된 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- Java를 실행할 수 있도록 사용자 ID에 충분한 저장영역을 할당해야 합니다. 지원되는 모든 Java 버전을 사용할 수 있습니다. 기본적으로 DFHJS2LS는 **JAVADIR** 매개변수에 지정된 Java 버전을 사용합니다.

이 태스크 정보

JSON 지원을 사용하여 서비스 제공자 애플리케이션의 JSON 스키마에서 언어 구조를 작성할 수 있습니다.

프로시저

1. DFHJS2LS 일괄처리 프로그램을 사용하여 웹 서비스 바인딩 파일 및 한 개 이상의 언어 데이터 구조를 생성하십시오. DFHJS2LS에는 사용자 애플리케이션이 필요로 하는 바인딩 파일 및 언어 구조를 작성할 수 있도록 사용자에게 유연성을 제공하는 대형 선택적 매개변수 세트가 포함되어 있습니다. 웹 서비스에 대해 기존 애플리케이션을 사용으로 설정할 때 이러한 옵션을 고려하십시오.
 - 데이터를 서비스 제공자 애플리케이션 프로그램에 전달하기 위해 CICS가 사용하는 메커니즘은? 채널을 사용하여 컨테이너로 데이터를 전달하거나 COMMAREA를 사용할 수 있습니다. 채널 및 컨테이너가 권장됩니다. 이를 **PGMINT** 매개변수로 지정하십시오.
 - 생성할 언어는? DFHJS2LS는 COBOL, C/C++ 또는 PL/I 언어 데이터 구조를 생성할 수 있습니다. **LANG** 매개변수를 사용하여 언어를 지정하십시오.
 - 사용할 맵핑 레벨은? 맵핑 레벨이 높을수록 런타임 시 문자 및 2진 데이터의 핸들링에 사용 가능한 제어 및 지원이 많아집니다. 일부 선택적 매개변수는 높은 맵핑 레벨에서만 사용 가능합니다. 사용 가능한 가장 높은 레벨의 맵핑을 사용하도록 권장됩니다. **MAPPING-LEVEL** 매개변수로 맵핑 레벨을 지정하십시오.

- 웹 서비스 요청자가 사용하게 할 URI는? **URI** 매개변수를 사용하여 상대 URI를 지정하십시오(예: URI=/my/test/webservice). 이 값은 URIMAP 자원을 작성할 때 CICS에서 사용됩니다.
- 웹 서비스 요청 및 응답을 실행할 트랜잭션 및 사용자 ID는? 서비스 요청자에 대한 응답을 구성하기 위해 별명 트랜잭션을 사용하여 애플리케이션을 실행할 수 있습니다. 별명 트랜잭션은 사용자 ID 아래 첨부됩니다. 이를 **TRANSACTION** 및 **USERID** 매개변수로 지정하십시오. 이러한 값은 URIMAP 자원을 작성할 때 사용됩니다. 특정 트랜잭션을 사용하지 않으려는 경우에는 이러한 매개변수를 사용하지 마십시오.

DFHJS2LS를 제출할 때 CICS는 웹 서비스 바인딩 파일을 생성하고 이 파일을 **WSBIND** 매개변수로 지정한 위치에 배치합니다. 언어 구조는 **PDSLIB** 매개변수로 지정한 파티션된 데이터 세트에 배치됩니다.

2. 생성된 웹 서비스 바인딩 파일을 웹 서비스 애플리케이션에 사용할 제공자 모드 PIPELINE 자원의 pickup 디렉토리에 복사하십시오. 바인딩 파일은 2진 모드로 복사해야 합니다.
3. 생성된 언어 구조와 인터페이스할 서비스 제공자 애플리케이션 프로그램을 작성하고 필수 비즈니스 로직을 구현하십시오.
4. **PIPELINE SCAN** 명령을 사용하여 WEBSERVICE 자원 및 URIMAP 자원을 동적으로 작성하십시오.
 - WEBSERVICE 자원은 CICS의 웹 서비스 바인딩 파일을 요약하고 런타임 시 사용됩니다.
 - URIMAP 자원은 CICS에 WEBSERVICE 자원과 특정 URI를 연관시키기 위한 정보를 제공합니다.

또는 직접 자원을 정의할 수 있지만 권장되지 않습니다.

결과

DFHJS2LS 제출에 문제점이 있거나 자원이 올바르게 설치되지 않는 경우 문제점 해결에서 JSON 지원자 문제점 해결의 내용을 참조하십시오.

 JSON 스키마 v4

 JSON 스키마 유효성 검증 v0

데이터 구조에서 서비스 제공자 애플리케이션 작성

CICS 웹 서비스 지원을 사용하여 고급 언어 데이터 구조에서 서비스 제공자 애플리케이션을 작성할 수 있습니다.

시작하기 전에

서비스 제공자 애플리케이션을 작성하기 전에 설정이 다음 전제조건을 준수하는지 확인하십시오.

- 고급 언어 데이터 구조가 다음 기준을 충족시켜야 합니다.
 - 데이터 구조는 소스 프로그램(예: COBOL 카피북)과 별도로 정의되어야 합니다.
 - PL/I 또는 COBOL 애플리케이션 프로그램이 입력 및 출력에 서로 다른 데이터 구조를 사용하는 경우 데이터 구조는 파티션된 데이터 세트에 있는 두 개의 다른 멤버에서 정의되어야 합니다. 입력 및 출력에 같은 구조를 사용하는 경우 이 구조는 단일 멤버에서 정의되어야 합니다.

C 및 C++에서 데이터 구조는 파티션된 데이터 세트에서 같은 멤버에 있을 수 있습니다.

- 언어 구조는 파티션된 데이터 세트에서 사용 가능해야 하며 CICS 리전에서 적절한 PIPELINE 자원을 작성해야 합니다.
- OMVS에 DFHLS2WS가 실행에 사용하는 사용자 ID를 정의해야 합니다.
- 사용자 ID는 z/OS UNIX 및 PDS 라이브러리에 대한 읽기 권한 및 **LOGFILE**, **WSBIND** 및 **JSON-SCHEMA-REQUEST**와 **JSON-SCHEMA-RESPONSE** 출력 매개변수에서 지정되는 디렉토리에 대한 쓰기 권한이 있어야 합니다.
- 사용자 ID는 Java를 실행할 수 있는 충분한 대용량 저장영역이 할당되어야 합니다. 지원되는 모든 Java 버전을 사용할 수 있습니다. 기본적으로 DFHLS2JS는 **JAVADIR** 매개변수에 지정된 Java 버전을 사용합니다.

프로시저

상위 레벨 데이터 구조에서 서비스 제공자 애플리케이션을 작성하려면 다음 단계를 따르십시오.

1. 서비스 제공자 애플리케이션 인터페이스에서 채널 및 다수의 컨테이너를 사용하는 경우 JSON로 인터페이스를 설명하는 채널 설명 문서를 작성하십시오. 채널 설명 문서는 z/OS UNIX의 적절한 디렉토리에 저장해야 합니다. CICS는 이 문서를 사용하여 채널의 여러 컨테이너에서 JSON 메시지를 구성하고 구성 해제합니다. 또는 한 채널에서 한 개의 컨테이너를 사용하고 채널 설명 문서를 작성하지 않을 수 있습니다.

채널 설명 문서를 작성하는 방법에 대한 자세한 정보는 221 페이지의 『채널 설명 문서 작성』의 내용을 참조하십시오.

2. DFHLS2JS 일괄처리 프로그램을 사용하여 언어 구조에서 웹 서비스 설명 및 웹 서비스 바인딩 파일을 생성하십시오. DFHLS2JS 일괄처리 프로그램은 **HLQ.XDFHINST**에서 찾을 수 있으며 여기서 **HLQ**는 CICS를 설치한 위치입니다. DFHLS2JS에는 사용자 애플리케이션이 필요로 하는 바인딩 파일 및 언어 구조를 작성할 수 있도록

사용자에게 유연성을 제공하는 대형 선택적 매개변수 세트가 포함되어 있습니다. 기존 애플리케이션에 대해 웹 서비스를 사용으로 설정할 때 다음 옵션을 고려하십시오.

- 데이터를 서비스 제공자 애플리케이션 프로그램에 전달하기 위해 CICS가 사용하게 하려는 메커니즘은? 채널을 사용하여 컨테이너로 데이터를 전달하거나 COMMAREA를 사용할 수 있습니다. **PGMINT** 매개변수를 사용하여 메커니즘을 지정하십시오. 애플리케이션 인터페이스가 채널 및 다수의 컨테이너를 사용하는 경우 **REQUEST-CHANNEL** 매개변수 및 선택적으로 **RESPONSE-CHANNEL**을 지정하십시오. 이러한 매개변수는 맵핑 레벨이 3.0 이상인 경우에만 사용할 수 있습니다.
- 사용할 맵핑 레벨은? 맵핑 레벨이 높을수록 런타임 시 문자 및 2진 데이터의 핸들링에 사용 가능한 제어 및 지원이 많아집니다. 일부 선택적 매개변수는 높은 맵핑 레벨에서만 사용 가능합니다. **MAPPING-LEVEL** 매개변수에 사용 가능한 가장 높은 레벨의 맵핑을 지정해야 합니다.
- 웹 서비스가 사용하게 할 URI는? **URI** 매개변수를 사용하여 절대 URI를 지정하십시오(예: **URI**=http://www.example.org:80/my/test/webservice). 이 주소의 상대 파트인 /my/test/webservice는 URIMAP 자원을 작성할 때 사용됩니다.

DFHLS2JS를 제출할 때 CICS는 웹 서비스 바인딩 파일을 생성하고 이 파일을 **WSBIND** 매개변수로 지정한 위치에 배치합니다. 생성된 JSON 스키마는 **JSON-SCHEMA-REQUEST** 및 **JSON-SCHEMA-RESPONSE** 매개변수로 지정된 위치에 배치됩니다.

3. 생성된 JSON 스키마를 검토하십시오.

이러한 스키마는 JSON 웹 서비스의 입력 및 출력 데이터 형식을 정의하는 데 사용됩니다. 애플리케이션 개발자는 JSON 웹 서비스와 상호작용하기 위해 애플리케이션을 작성할 때 이러한 스키마를 사용해야 합니다.

참고: 생성된 스키마를 변경하면 생성된 웹 서비스 바인딩 파일 WSBIND를 무효화합니다.

스키마를 변경하려는 경우(예: 스키마 내에서 필드의 이름 바꾸기) DFHJS2LS를 사용하여 새 웹 서비스 바인딩 파일 및 새 언어 구조 세트를 생성해야 합니다. 새 언어 구조를 사용하려면 CICS의 애플리케이션 프로그램을 변경해야 합니다.

4. 웹 서비스 바인딩 파일을 웹 서비스 애플리케이션에 사용할 제공자 모드 파일프라인의 픽업 디렉토리에 복사하십시오. 웹 서비스 바인딩 파일은 2진 모드로 복사해야 합니다.
5. **PIPELINE SCAN** 명령을 사용하여 WEBSERVICE 자원 및 URIMAP 자원을 동적으로 작성하십시오.

- WEBSERVICE 자원은 CICS의 웹 서비스 바인딩 파일을 포함하고 런타임 시 사용됩니다.
- URIMAP 자원은 CICS에 WEBSERVICE 자원과 특정 URI를 연관시키기 위한 정보를 제공합니다.

또는 직접 자원을 정의할 수 있습니다.

결과

서비스 제공자 애플리케이션의 작성이 완료됩니다.

DFHLS2JS 제출에 문제점이 있거나 자원이 올바르게 설치되지 않는 경우 문제점 해결에서 JSON 지원자 문제점 해결의 내용을 참조하십시오.

다음에 수행할 작업

서비스에 액세스할 웹 서비스를 개발하는 모든 사용자가 사용할 수 있도록 웹 서비스 설명을 공개하십시오.

채널 설명 문서 작성

서비스 제공자 애플리케이션이 여러 컨테이너와의 채널 인터페이스를 사용하는 경우 채널 설명 문서를 작성합니다.

이 태스크 정보

XML 편집기를 사용하여 채널 설명 문서를 작성하십시오. 채널 설명에 대한 스키마는 channel.xsd라고 하며 /usr/lpp/cicsts/cicsts53/schemas/channel 디렉토리(여기서, /usr/lpp/cicsts/cicsts53는 z/OS UNIX에서 CICS 파일의 기본 설치 디렉토리임)에 있습니다.

프로시저

1. <채널> 요소 및 CICS 채널 네임스페이스를 사용하여 XML 문서를 작성하십시오.

```
<channel name="myChannel" xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```

2. 애플리케이션 프로그램 인터페이스가 채널에서 사용하는 모든 컨테이너에 <컨테이너> 요소를 추가하십시오. 이름, 유형 및 속성을 사용하여 각 컨테이너를 설명해야 합니다. 다음 예제는 서로 다른 속성 값을 가지는 여섯 개의 컨테이너를 보여줍니다.

```
<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>

```

이 구조 요소는 파티션된 데이터 세트 멤버에 있는 언어 구조로 콘텐츠가 정의됨을 표시합니다.

3. z/OS UNIX에서 XML 문서를 저장하십시오.

채널 스키마

채널 설명 문서는 다음 스키마를 준수해야 합니다.

```

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.ibm.com/xmlns/prod/CICS/channel"
  xmlns:tns="http://www.ibm.com/xmlns/prod/CICS/channel" elementFormDefault="qualified">
  <element name="channel"> 1
    <complexType>
      <sequence>
        <element name="container" maxOccurs="unbounded" "unbounded" minOccurs="0"> 2
          <complexType>
            <sequence>
              <element name="structure" minOccurs="0"> 3
                <complexType>
                  <attribute name="location" type="string" use="required"/>
                  <attribute name="structure" type="string" use="optional"/>
                </complexType>
              </element>
            </sequence>
            <attribute name="name" type="tns:name16Type" use="required"/>
            <attribute name="type" type="tns:typeType" use="required"/>
            <attribute name="use" type="tns:useType" use="required"/>
          </complexType>
        </element>
      </sequence>
      <attribute name="name" type="tns:name16Type" use="optional" />
    </complexType>
  </element>
  <simpleType name="name16Type">
    <restriction base="string">
      <maxLength value="16"/>
    </restriction>
  </simpleType>
  <simpleType name="typeType">
    <restriction base="string">
      <enumeration value="char"/>
      <enumeration value="bit"/>
    </restriction>
  </simpleType>
  <simpleType name="useType">
    <restriction base="string">
      <enumeration value="required"/>
      <enumeration value="optional"/>
    </restriction>
  </simpleType>
</schema>

```

1. 이 요소는 CICS 채널을 나타냅니다.

2. 이 요소는 채널 내 CICS 컨테이너를 나타냅니다.

3. 구조는 '비트' 모드 컨테이너와 함께만 사용할 수 있습니다. '위치' 속성은 컨테이너의 콘텐츠를 맵핑하는 파일의 위치를 표시합니다. '구조' 속성은 C 및 C++에서 사용되어 구조의 이름을 표시할 수 있습니다.

다음에 수행할 작업

DFHLS2JS를 실행하여 웹 서비스 제공자 애플리케이션에 대한 맵핑 및 JSON 스키마를 작성하십시오. DFHLS2JS는 채널 설명 문서에서 컨테이너가 지정되는 순서대로 채널의 맵핑을 JSON 스키마에 넣습니다.

생성된 JSON 스키마 사용자 정의

DFHLS2JS에 의해 생성되는 JSON 스키마에는 공개 전 변경하기에 적합할 수 있는 일부 자동 생성 콘텐츠가 있습니다. JSON 스키마를 사용자 정의하면 웹 서비스 바인딩 파일이 재생성될 수 있으며 일부 경우에는 랩퍼 프로그램이 작성될 수 있습니다.

이 태스크 정보

다음 단계에 따라 생성된 JSON 스키마를 사용자 정의하십시오.

프로시저

1. HTTPS 지원을 광고하려면 DFHLS2JS의 **URI** 매개변수를 사용하여 절대 URI를 설정하십시오.
2. 웹 서비스의 네트워크 위치를 제공하려면 DFHLS2JS의 **URI** 매개변수를 사용하여 절대 URI를 설정하십시오.
3. JSON 스키마에서 자동으로 생성된 이름이 사용자의 용도에 적합한지 여부를 고려해 보십시오. 특성의 이름을 바꿀 수 있습니다.

이러한 값은 클라이언트 프로그램을 코드화할 프로그래밍 방식 인터페이스 일부를 구성합니다. 생성된 이름의 의미가 분명하지 않으면 장기간에 걸친 애플리케이션 코드의 유지보수가 좀 더 어려울 수 있습니다.

이러한 값을 변경하는 경우 DFHJS2LS를 사용하여 웹 서비스 바인딩 파일을 재생성해야 합니다. 생성되는 언어 구조는 기존 애플리케이션과 호환 가능하지 않을 가능성이 높기 때문에 애플리케이션 변경이 필요할 수 있습니다. 생성된 언어 구조를 검토하고 224 페이지의 4 단계에서 논의한 대로 랩퍼 프로그램 작성을 고려해 보십시오.

4. JSON 스키마에서 노출된 COMMAREA 필드가 적합한지 고려하십시오. JSON 클라이언트 개발자에게 유용하지 않은 필드 제거를 고려할 수 있습니다.
 - 출력 값에만 사용되는 필드는 입력 데이터 구조를 맵핑하는 스키마에서 제거할 수 있습니다.
 - 필터 필드입니다.

- 자동으로 생성되는 어노테이션입니다.

이러한 변경사항을 작성하는 경우 DFHJS2LS를 사용하여 웹 서비스 바인딩 파일을 재생성해야 합니다. 생성되는 새 언어 구조가 원래 언어 구조와 호환 가능하지 않기 때문에 랩퍼 프로그램을 작성하여 새 표시에서 이전 표시로 데이터를 맵핑해야 합니다. 이 랩퍼 프로그램은 대상 애플리케이션 프로그램에 대해 **EXEC CICS LINK** 명령을 수행한 다음 리턴된 데이터를 맵핑해야 합니다.

이 사용자 정의 레벨에서는 가장 많은 노력이 필요하지만 JSON 클라이언트 개발자에게 가장 유용한 프로그래밍 방식 인터페이스가 생성됩니다.

결과

비즈니스 요구사항에 맞게 사용자 정의된 JSON 스키마 및 CICS에 비즈니스 요구사항을 구현하는 프로그램을 가지고 있습니다.

RESTful 웹 서비스 제공자 애플리케이션 작성

RESTful JSON 웹 서비스의 CICS 구현은 SOAP 웹 서비스의 CICS 구현과 유사합니다. 대부분의 개념 및 아키텍처는 공유되지만 CICS에서는 JSON 스키마 사용이 필요합니다.

이 태스크 정보

RESTful JSON 웹 서비스 구현에는 다음 태스크가 포함됩니다.

프로시저

1. 애플리케이션 인터페이스를 생성하십시오.

입력:

- RESTful 웹 서비스에 대한 데이터 모델을 정의하는 JSON 스키마

출력:

- JSON 스키마를 맵핑하는 언어 구조(예: COBOL 카피북)
- WSBind 파일

DFHJS2LS 유틸리티를 실행하고 적절한 입력 매개변수를 지정하십시오. 이러한 매개변수는 다음과 같습니다.

- JSON 스키마의 위치
- 지원되는 메소드 목록(GET, PUT, POST 및 DELETE는 기본적으로 사용으로 설정됨)
- 서비스가 배치되는 URI
- 서비스를 구현하는 애플리케이션 PROGRAM의 이름
- 필수 데이터 맵핑 매개변수

2. 이 인터페이스를 사용하는 애플리케이션을 작성하십시오.

입력:

- 224 페이지의 1 단계의 언어 구조
- 구현할 RESTful 조작에 대한 인식

출력:

- CICS에 대한 배치에 적합한 프로그램

다음을 수행하는 프로그램을 작성하십시오.

참고: DFHJS2LS에 대한 **CONTID** 매개변수에 자체 컨테이너의 이름을 제공하는 경우 다음 단계에서 언급될 때마다 DFHWS-DATA 대신 이 컨테이너를 사용해야 합니다.

- a. URI를 검사하여 자원의 ID를 확인하십시오. CICS는 URI의 관련 컴포넌트를 식별하는 데 도움이 되는 여러 컨테이너를 제공합니다. 이러한 컨테이너는 225 페이지의 표 2에 설명되어 있습니다. 다음 예제는 URI에 대한 각 컨테이너의 콘텐츠를 보여줍니다.

`http://www.example.org:10000/JSONServices/CustomerDetails/13388?action=query`

일치한 URIMAP의 경로가 `/JSONServices/CustomerDetails/*`인 경우

표 16. DFHWS-URI 컨테이너. DFHWS-URI 컨테이너

컨테이너	콘텐츠	예제
구성에서 DFHWS-URI	전체 URI	<code>http://www.example.org:10000/JSONServices/CustomerDetails/13388?action=query</code>
	URIMAP과 일치한 URI 경로 부분	<code>/JSONServices/CustomerDetails/*</code>
구성에서 DFHWS-URI-RESID	제거된 URIMAP과 일치하는 부분이 포함된 URI 경로(자원 ID)	13388
구성에서 DFHWS-URI-QUERY	조회 문자열	<code>action=query</code>

- b. URI를 유효성 검증하십시오. 문제점이 있을 경우 CICS에 문제점을 보고하고 종료하십시오.
- c. DFHHTTPMETHOD 컨테이너를 조회하여 구동되고 있는 메소드를 판별하십시오. 자세한 정보는 구성에서 DFHHTTPMETHOD의 내용을 참조하십시오.
- d. POST(작성) 메소드(필요한 경우)의 경우:
- DFHWS-DATA 컨테이너에서 입력 데이터를 읽으십시오.
 - 224 페이지의 1단계에서 생성된 언어 구조를 사용하여 데이터를 해석하십시오.

- 데이터를 유효성 검증하십시오. 문제점이 있을 경우 CICS에 문제점을 보고하고 종료하십시오.
 - 자원 작성에 필요한 애플리케이션 특정 처리를 수행하십시오.
 - 선택적으로 DFHRESPONSE 컨테이너를 작성하여 클라이언트에게 새 자원의 ID를 알리십시오. 이 컨테이너의 콘텐츠는 CICS에 의해 변환되는 것이 아니라 HTTP 응답으로 직접 전송됩니다. DFHWS-DATA 컨테이너는 무시됩니다.
- e. GET(조회) 또는 HEAD(조회) 메소드가 필요한 경우 *resource*를 나타내는 데이터를 DFHWS-DATA 컨테이너에 작성하십시오.
- f. PUT(설정) 메소드가 필요한 경우:
- DFHWS-DATA 컨테이너에서 입력 데이터를 읽으십시오.
 - 224 페이지의 1단계에서 생성된 언어 구조를 사용하여 데이터를 해석하십시오.
 - 데이터를 유효성 검증하십시오. 문제점이 있을 경우 CICS에 문제점을 보고하고 종료하십시오.
 - 자원 갱신에 필요한 애플리케이션 특정 처리를 수행하십시오.
- g. DELETE 메소드가 필요하면 자원을 삭제하는 데 필요한 애플리케이션 특정 처리를 수행하십시오.

참고: CICS에 의해 구현되는 RESTful 데이터 모델은 기본적으로 PUT, POST 또는 DELETE 메소드에 대해 응답 본문을 전송하지 않습니다. RESTful 애플리케이션은 일반적으로 HTTP 상태 코드를 사용하여 성공 또는 실패를 나타내십시오. 애플리케이션이 정상적으로 완료되면 CICS가 HTTP 응답 200(OK)을 전송합니다. 오류 응답 전송에 대한 자세한 정보는 애플리케이션 개발에서 애플리케이션 오류 보고의 내용을 참조하십시오. PUT, POST 또는 DELETE 메소드에 대해 응답 본문을 전송하려는 경우 DFHRESPONSE 컨테이너를 작성해야 합니다. 있는 경우 CICS는 추가 처리 없이 HTTP 본문으로 이 컨테이너의 콘텐츠를 전송합니다. CICS는 이러한 메소드에 대한 응답 처리에서 DFHWS-DATA 컨테이너를 무시합니다.

3. 아티팩트를 배치하십시오.

입력:

- 224 페이지의 1 단계의 WSBind 파일
- 225 페이지의 2 단계의 CICS 프로그램
- JSON에 대해 구성되는 CICS 제공자 모드 파이프라인 자원

출력:

- 배치된 RESTful JSON 웹 서비스

일반적인 방식으로 CICS에 프로그램을 배치하십시오.

다음 중 하나를 수행하십시오.

- WSBind 파일을 파이프라인의 '픽업' 디렉토리에 배치한 다음 **PIPELINE SCAN** 명령을 발행하여 WEBSERVICE 및 URIMAP 자원을 작성하십시오.
- WEBSERVICE 자원 및 연관된 URIMAP 자원을 수동으로 정의하고 설치하십시오. URIMAP은 URI를 PIPELINE 및 WEBSERVICE 둘 다와 연관시켜야 합니다.

4. 서비스를 테스트하십시오.

입력:

- JSON 스키마
- RESTful 웹 서비스의 URI
- 3단계에서 배치된 서비스
- 선택한 JSON 테스트 클라이언트

출력:

- 성공적으로 처리한 요청

선택한 테스트 클라이언트를 사용하여 CICS에 JSON 요청을 전송하십시오.

예상치 못한 응답을 수신하는 경우 문제점 판별을 시도하십시오. 자세한 정보는 JSON 요청 문제점 해결의 내용을 참조하십시오.

관련 정보:

 RESTful 서비스

RESTful 웹 서비스 제공자 애플리케이션에 대한 디자인 고려사항

이 주제에서는 JSON용 RESTful 웹 서비스 제공자 애플리케이션을 계획 및 디자인할 때 고려해야 할 몇 가지 문제를 설명합니다.

자원의 컬렉션

RESTful API를 위한 공통 디자인은 자원의 컬렉션 검색을 지원하는 것입니다. 예를 들어, 다음과 같이 일련의 오브젝트를 리턴하는 서비스가 있을 수 있습니다.

GET /Services/CustomerDetails?Surname=Cooper

이 요청은 성이 "Cooper"인 모든 CustomerDetails 오브젝트에 대한 정보를 리턴할 것으로 예상됩니다. 다음과 같이 좀 더 구체적인 URI를 사용하면 개별 CustomerDetails 오브젝트가 리턴될 수도 있습니다.

GET /Services/CustomerDetails/Customer27

이 예제에서 Customer27은 특정 고객의 기본 키입니다. 이 두 번째 조회의 출력은 CustomerDetails 오브젝트의 인스턴스가 됩니다. 첫 번째 조회의 출력은 덜 명확합니다. 이 조회에서는 CustomerDetails 오브젝트의 목록을 리턴하거나 CustomerDetails 오브젝트에 대한 URI 목록(클라이언트가 개별 검색을 계속할 수 있음)을 리턴할 수 있습니다. 두 규칙 모두 공통입니다.

CICS에서 컬렉션을 구현하려면 데이터 인스턴스의 목록 또는 URI의 목록을 설명하는 JSON 스키마를 작성하십시오. 그런 다음 서비스를 빌드하고 평소와 같이 이 서비스를 구현할 수 있습니다. 이 예제에서 GET 메소드만 구현하도록 선택할 수 있습니다. 클라이언트가 전체 데이터 세트에서 페이지를 앞으로 넘기거나 뒤로 넘길 수 있도록 페이지 매기기 서비스 구현을 고려할 수 있습니다. 예를 들어, 다음과 같습니다.

```
GET /Services/CustomerDetails?startRecord=200&endRecord=225
```

CICS에서 두 개의 URIMAP 자원(및 두 개의 WEBSERVICE 자원)이 필요할 가능성이 높습니다. 컬렉션에 대해 루트 URI 구조를 맵핑하는 URIMAP 및 인스턴스에 대해 와일드카드화된 URIMAP. 예를 들어, 다음과 같습니다.

```
URIMAP1: Path=/Services/CustomerDetails WEBSERVICE=CollectionService
URIMAP2: Path=/Services/CustomerDetails/* WEBSERVICE=InstanceService
```

캐시 관리

RESTful 아키텍처는 표준 HTTP 캐시 관리 기법과의 통합을 권장합니다. 이러한 통합을 통해 GET 요청의 결과가 네트워크에서 캐시될 수 있기 때문에 서버에 대한 부담을 줄일 수 있습니다. 이를 수행하기 위한 메커니즘에는 GET 요청에 대해 리턴되는 데이터의 만기 날짜/시간 설정이 포함됩니다.

CICS에는 애플리케이션 제어 캐시 만기를 지원하기 위한 일반적인 용도의 메커니즘이 없지만 EXEC CICS WEB WRITE HTTPHEADER API를 사용하여 적합한 HTTP 헤더를 추가하도록 파이프라인 핸들러 프로그램을 작성할 수 있습니다. 애플리케이션 프로그램도 이와 유사한 작업을 수행할 수 있지만 HTTP 요청을 수신하는 같은 CICS 리전에서 호스팅되는 경우에만 해당됩니다.

애플리케이션 오류 보고

이 주제를 읽고 RESTful JSON 웹 서비스 제공자 애플리케이션이 클라이언트에게 오류를 보고하는 방법을 이해하십시오.

하향식 시나리오에서는 애플리케이션이 오류 조건을 보고해야 할 가능성이 높습니다. 예를 들어, 오류는 『Account Number not recognized』일 수 있습니다. 이 요구사항은 하향식 시나리오에서 고유합니다. 상향식 개발에서는 애플리케이션에 데이터 필드에서 인코딩되는 오류 보고 메커니즘이 있거나 애플리케이션이 이상 종료됩니다. 하향식 시나리오에서 JSON 스키마는 오류를 보고하는 필드를 거의 정의하지 않기 때문에 대안이 필요합니다.

SOAP 기반 웹 서비스의 경우 **EXEC CICS SOAPFAULT** API를 사용하여 이 문제점을 처리합니다. SOAP 결함 메시지는 JSON에 존재하지 않습니다. 대신 DFHHTTPSTATUS 컨테이너를 사용하여 JSON 애플리케이션에 대한 애플리케이션 발견 오류를 보고할 수 있습니다. 자세한 정보는 구성에서 DFHHTTPSTATUS의 내용을 참조하십시오.

참고: 애플리케이션은 필요한 경우 DFHRESPONSE 컨테이너 또는 기타 제어 컨테이너를 사용하여 좀 더 세부적인 오류 응답을 제공할 수도 있습니다.

JSON 웹 서비스 제한사항

이 참조 자료를 사용하여 JSON 웹 서비스에서 지원되지 않는 기능을 이해할 수 있습니다.

다음 기능은 지원되지 않습니다.

- JSON이 포함된 요청자 모드 파이프라인은 지원되지 않습니다.
- 스키마에 대한 JSON 데이터의 런타임 유효성 검증이 지원되지 않습니다. JSON 페이로드와 함께 사용되는 WEBSERVICE 자원의 VALIDATION 속성 값은 무시됩니다.
- JSON 데이터(Badgerfish 또는 맵핑된 규칙)에서 네임스페이스 사용은 지원되지 않습니다.
- CICS에 전송된 JSON 페이로드는 UTF-8로 인코딩해야 합니다. 다른 인코딩은 지원되지 않습니다. 마찬가지로, CICS에 의해 전송된 JSON은 항상 UTF-8로 인코딩됩니다.
- JSON 파이프라인을 사용한 WebSphere MQ 전송은 지원되지 않습니다.
- 공급업체 변환기 프로그램을 JSON 변환기에 사용하는 것은 지원되지 않습니다.
- SOAP 웹 서비스 애플리케이션을 위해 JSON 파이프라인에서 작성된 WSBind 파일의 재사용은 지원되지 않습니다. JSON 서비스 제공자 애플리케이션에서 사용할 WSBind 파일은 JSON 지원에 의해 생성되어야 합니다.
- CICS가 JSON 페이로드를 변환할 때 JSON 페이로드가 일부 필수 콘텐츠를 잃는 경우 애플리케이션 프로그램에 전달 시 데이터 구조 내 해당 필드가 초기화되지 않습니다.
- CICS는 따옴표로 묶인 정수 값 외에 부호 있는 긴 정수($2^{63} - 1$)의 최대값을 초과하는 정수 값을 변환할 수 없습니다.
- 단순 데이터 유형 또는 배열의 사용은 JSON 스키마의 루트에서 지원되지 않습니다. JSON 오브젝트는 단순 데이터 유형 및 배열로 구성할 수 있지만 JSON 오브젝트를 설명하는 데 JSON 스키마가 필요합니다.
- maxItems 값이 1인 JSON 스키마에서 배열이 선언되는 경우 런타임 시 JSON을 생성할 때 CICS는 이 배열을 단순 문자열 또는 정수로 직렬화합니다.

중요사항: JSON 특성 이름에 대해 지원되는 유일한 문자는 첫 번째 문자에 대해서는 A-Z a-z _ : 이고 모든 후속 문자에 대해서는 A-Z a-z 0-9 _ : . -입니다.

현재 Axis2 웹 서비스 지원에는 애플리케이션 및 사용자 정의의 개발 및 배치를 위한 다수의 옵션이 있습니다. 다음 옵션은 지원되지 않습니다.

- 사용자 제공 애플리케이션 핸들러 - CICS 제공 애플리케이션 핸들러 클래스 `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler`를 사용해야 합니다.
- 사용자 작성 Axis2 Java 애플리케이션
- SOAPFAULT 및 WS-Addressing API는 JSON 파이프라인에서 사용할 수 없습니다.

컨테이너 제한사항

JSON 요청을 처리할 때 일부 파이프라인 컨테이너는 채워지지 않음에 유의하십시오. 자세한 정보는 구성의 파이프라인에서 사용된 컨테이너의 내용을 참조하십시오.

RESTful 웹 서비스의 차이점

INQUIRE PIPELINE에서

- SOAPLEVEL은 NOTSOAP을 리턴함
- MTOMNOXOPST, MTOMST, SENDMTOMST, SOAPRNUM, SOAPVNUM, XOPDIRECTST 및 XOPSUPPORTST 속성은 사용되지 않습니다.

INQUIRE WEBSERVICE에서

- ARCHIVEFILE, BINDING, VALIDATIONST, XOPDIRECTST 및 XOPSUPPORTST 속성은 사용되지 않습니다.
- WSDLFILE은 WEBSERVICE와 연관된 JSON 스키마 파일의 이름을 리턴합니다.

WEBSERVICE 자원에서

- ARCHIVEFILE 및 VALIDATION 매개변수는 사용되지 않으며 해당 값은 무시됩니다.
- WSDLFILE은 WEBSERVICE와 연관된 JSON 스키마 파일의 이름입니다.

제 9 장 웹 서비스를 위한 런타임 처리

웹 서비스 제공자에게 요청을 전달하거나 웹 서비스 요청자로부터 요청을 수신하려면 사용자 애플리케이션(또는 랩퍼 프로그램)이 CICS의 웹 서비스 지원과 올바르게 상호작용해야 합니다. 인바운드 및 아웃바운드 요청이 핸들링되는 방법을 결정하기 위해 파이프라인에서 발생하는 처리를 제어할 수도 있습니다.

CICS가 웹 서비스 지원으로 배치되는 서비스 제공자 프로그램을 호출하는 방법

CICS 웹 서비스 지원을 사용하여 배치되는 서비스 제공자 애플리케이션이 호출될 때, CICS가 COMMAREA 또는 채널로 링크됩니다.

PGMINT 매개변수로 JCL 프로시저 **DFHWS2LS** 또는 **DFHLS2WS**를 실행할 때 사용되는 인터페이스의 종류를 지정합니다. 채널을 지정하면 **CONTID** 매개변수에서 컨테이너의 이름을 지정할 수 있습니다.

- **COMMAREA** 인터페이스로 프로그램이 호출되면, **COMMAREA**에는 SOAP 요청으로부터 CICS가 작성한 최상위 레벨 데이터 구조가 포함됩니다.
- 프로그램이 채널 인터페이스로 호출되면, **DFHWS2LS** 또는 **DFHLS2WS**의 **CONTID** 매개변수에 지정된 컨테이너의 프로그램으로 최상위 레벨 데이터 구조가 전달됩니다. **CONTID** 매개변수를 지정하지 않으면, 데이터가 컨테이너 **DFHWS-DATA**에서 전달됩니다. 채널 인터페이스는 다양한 수의 요소를 가진 배열을 지원하고, 이는 일련의 컨테이너에서 일련의 연결된 데이터 구조로 표시됩니다. 이러한 컨테이너도 존재합니다.

컨테이너에 대해 작업하기 위해 API 명령을 코드화할 때, 모든 컨테이너가 현재 채널(프로그램으로 전달된 채널)과 연관되므로 **CHANNEL** 옵션을 지정할 필요가 없습니다. 채널의 이름을 알아야 하는 경우, **EXEC CICS ASSIGN CHANNEL** 명령을 사용하십시오.

프로그램이 요청을 처리할 때, 이는 응답을 리턴하기 위해 동일한 메커니즘을 사용해야 합니다. 요청이 **COMMAREA**에 수신되면 응답은 **COMMAREA**에서 리턴되어야 합니다. 요청이 컨테이너에 수신되면 응답은 동일한 컨테이너에서 리턴되어야 합니다.

애플리케이션 프로그램이 응답 메시지를 실행할 때 오류가 발생하면 CICS는 애플리케이션이 동기점을 수행하지 않는 경우 모든 변경사항을 롤백합니다.

사용자 프로그램이 제공하는 웹 서비스가 응답을 리턴하도록 설계되지 않은 경우, CICS는 프로그램이 리턴할 때 **COMMAREA** 또는 컨테이너에 있는 것을 무시합니다.

웹 서비스 지원으로 배치되는 애플리케이션으로부터 웹 서비스 호출

웹 서비스 지원으로 배치되는 서비스 요청자 애플리케이션은 **EXEC CICS INVOKE SERVICE** 명령을 사용하여 웹 서비스를 호출합니다. 요청 및 응답이 컨테이너 DFHWS-DATA에서 데이터 구조로 맵핑됩니다. 이 서비스 호출 방법은 JSON에 대해 지원되지 않습니다.

프로시저

1. 채널을 작성하고 이를 컨테이너로 채우십시오. 최소한도로 컨테이너 DFHWS-DATA가 존재해야 합니다. DFHWS-DATA는 CICS가 SOAP 요청으로 변환하는 상위 레벨 데이터 구조를 보유합니다. SOAP 요청에 다수의 요소를 가지는 배열이 포함되는 경우, 일련의 컨테이너에서 일련의 연결된 데이터 구조로 표시됩니다. 이러한 컨테이너는 채널에 존재해야 할 수도 있습니다.
2. 대상 웹 서비스를 호출하십시오. 다음 명령을 사용하십시오.

```
EXEC CICS INVOKE SERVICE(webservice)  
                        CHANNEL(userchannel)  
                        OPERATION(operation)
```

여기서,

- *webservice*는 호출될 웹 서비스를 정의하는 WEBSERVICE 자원의 이름입니다. WEBSERVICE 자원은 웹 서비스와 통신할 때 CICS가 사용하는 웹 서비스 바인딩 파일 및 웹 서비스 설명의 위치를 지정합니다.
- *userchannel*은 애플리케이션의 데이터 구조와 연관되는 기타 컨테이너 및 컨테이너 DFHWS-DATA를 보유하는 채널입니다.
- *operation*은 대상 웹 서비스에서 호출되는 조작의 이름입니다.

자세한 정보는 381 페이지의 『웹 서비스에 대한 로컬 최적화』의 내용을 참조하십시오.

3. 명령이 성공적이면 채널에서 응답 컨테이너를 검색하십시오. 최소한도로 컨테이너 DFHWS-DATA가 존재합니다. SOAP 응답으로부터 CICS가 작성한 상위 레벨 데이터 구조를 보유합니다. 응답에 다수의 요소를 가지는 배열이 포함되는 경우, 일련의 컨테이너에서 일련의 연결된 데이터 구조로 표시됩니다. 이러한 컨테이너는 채널에 존재합니다.
4. 서비스 요청자가 호출된 웹 서비스로부터 SOAP 결함 메시지를 수신하는 경우, 애플리케이션 프로그램이 변경사항을 롤백해야 하는지 여부를 판별해야 합니다. SOAP 결함이 발생하면 6의 RESP2 값을 가진 INVREQ 오류가 애플리케이션 프로그램으로 리턴됩니다. 그러나, 최적화가 실시되면 동일한 트랜잭션이 요청자와 제공자 모두에서 사용됩니다. 오류가 로컬로 최적화된 웹 서비스 제공자에 발생하면 트랜잭션에 의해 수행되는 모든 작업은 제공자와 요청자 모두에서 롤백합니다. INVREQ 오류는 16의 RESP2 값을 가진 요청자로 리턴됩니다.

웹 서비스에 대한 로컬 최적화

WEBSERVICE 자원과 연관되는 웹 서비스 바인딩 파일에서 제공자 애플리케이션 이름을 사용하여 웹 서비스 요청의 로컬 최적화를 사용할 수 있습니다.

INVOKE SERVICE 명령을 사용하여 URIMAP(urimap) 또는 URI(uri)를 지정할 수 있습니다. 여기서, URI는 호출할 웹 서비스의 URI입니다. URIMAP이 지정되면, CICS는 URI를 해결하기 위해 표시된 클라이언트 모드 URIMAP를 사용합니다. 이러한 옵션이 생략되면, CICS는 WEBSERVICE가 생성된 웹 서비스 설명(WSDL)에 지정된 URI를 사용합니다.

표시된 WEBSERVICE가 요청자 모드 PIPELINE에 배치되는 경우, CICS는 원격 웹 서비스를 호출합니다. 이는 가장 일반적인 시나리오입니다.

표시된 WEBSERVICE가 제공자 모드 PIPELINE에 배치되는 경우, CICS는 서비스를 로컬로 호출합니다. 이 최적화를 사용하는 경우, EXEC CICS INVOKE SERVICE 명령이 EXEC CICS LINK 명령에 최적화됩니다. 이는 결과적으로 중요한 성능 이점이 되지만, 다음 제한사항을 소개합니다.

- PIPELINE이 구동되지 않으므로 핸들러 프로그램이 사용되지 않습니다.
- PIPELINE 제어 컨테이너는 채널에 존재하지 않습니다. DFHWS-DATA, DFHWS-OPERATION 및 DFHWS-URI 컨테이너를 포함하여 일부 컨테이너가 존재합니다. 정상적으로 XML을 포함하는 컨테이너가 존재하지 않습니다. 이는 DFH-REQUEST, DFHWS-BODY 및 DFHWS-XMLNS 컨테이너를 포함합니다.
- 제공자 및 요청자 애플리케이션 모두 동일한 카피북을 공유하고 동일한 프로그래밍 언어에 구현되어야 합니다.
- 제공자 및 요청자 애플리케이션 모두는 단일 작업 단위를 공유합니다.
- 웹 서비스가 응답을 리턴할 것으로 예상되지 않으면, EXEC CICS INVOKE SERVICE 명령은 대상 PROGRAM이 종료된 후에 애플리케이션에 제어를 리턴합니다.

최적화된 웹 서비스를 로컬로 사용하지만 PIPELINE을 통해 데이터가 처리되도록 하려는 경우, 386 페이지의 『요청자 파이프라인 처리 제어를 위한 옵션』에 설명된 cics URI 형식을 사용하십시오. 이 메커니즘은 전체적으로 최적화된 접근법을 사용하는 것 보다 덜 효율적이지만, 이는 네트워크로 향하는 처리 비용을 피합니다.

웹 서비스 지원에 의해 생성되는 코드에 대한 런타임 제한사항

런타임 시 CICS는 동등한 데이터 구조에서 웹 서비스 설명(WSDL)을 따르는 유효한 SOAP 메시지를 거의 변형시킬 수 있습니다. 그러나, 웹 서비스 지원 일괄처리 작업을 사용하여 서비스 요청자 또는 서비스 제공자 애플리케이션을 개발할 시기를 인식해야 하는 일부 제한사항이 있습니다.

코드 페이지

CICS는 코드 페이지를 식별하는 적절한 HTTP 또는 WebSphere MQ 헤더가 있는 경우 코드 페이지에서 전송된 SOAP 메시지를 지원할 수 있습니다. CICS는 애플리케이션 프로그램에 의해 요청되는 코드 페이지에 전송하기 전에 파이프라인에서 이를 처리하도록 UTF-8로 SOAP 메시지를 변환합니다. 성능 영향을 최소화하려면 SOAP 메시지를 CICS에 전송할 때 UTF-8 코드 페이지를 사용하는 것이 추천됩니다. CICS는 UTF-8에서 SOAP 메시지를 항상 전송합니다.

애플리케이션 데이터가 EBCDIC 또는 UTF-16으로 인코딩된 경우에만 CICS가 SOAP 메시지를 변환할 수 있습니다. 데이터가 UTF-8, ASCII 및 ISO8859-1과 같은 코드 페이지에서 인코딩될 것으로 예상하는 애플리케이션은 지원되지 않습니다. 사용자 데이터 구조 및 SOAP 메시지 내에서 DBCS 문자를 사용하려는 경우, DBCS를 지원하는 코드 페이지를 지정해야 합니다. 사용자가 선택하는 EBCDIC 코드 페이지는 Java 및 z/OS 변환 서비스 모두에 의해 지원되어야 합니다. z/OS 변환 서비스는 SOAP 메시지의 코드 페이지에서 UTF-8로 변환하는 것을 지원하도록 구성되어야 합니다. UTF-16 지원에 대한 자세한 정보는 애플리케이션 데이터에서 UTF-16에 대한 지원의 내용을 참조하십시오.

적절한 코드 페이지를 설정하기 위해 웹 서비스 지원 작업에서 **LOCALCCSID** 시스템 초기화 매개변수 또는 선택적 **CCSID** 매개변수를 사용할 수 있습니다. **CCSID** 매개변수를 사용하는 경우, 지정하는 값은 해당 특정 웹 서비스에 대한 **LOCALCCSID** 코드 페이지를 대체합니다. **CCSID** 매개변수를 지정하지 않는 경우, **LOCALCCSID** 코드 페이지가 데이터를 변환하기 위해 사용되고 웹 서비스 바인딩 파일은 US EBCDIC(Cp037)에 인코딩됩니다.

컨테이너

서비스 제공자 모드에서 **PGMINT** 매개변수에 CHANNEL의 값이 있다는 것을 지정하는 경우, 애플리케이션 데이터를 보유하는 컨테이너는 2진 모드로 작성되어야 하고 읽어야 합니다. 이 컨테이너는 기본적으로 DFHWS-DATA입니다. **PUT CONTAINER** 명령은 DATATYPE 옵션이 BIT로 설정되도록 하거나 BIT가 기본으로 남아 있도록 FROMCCSID 옵션을 생략해야 합니다. 예를 들어, 다음 코드는 현재 채널의 CUSTOMER-RECORD 컨테이너에 있는 데이터가 2진 모드로 작성되어야 한다는 것을 명백하게 언급합니다.

```
EXEC CICS PUT CONTAINER (CUSTOMER-RECORD)
      FROM (CREC)
      DATATYPE(BIT)
```

비록 컨테이너 자체가 모두 BIT 모드에 있을지라도, 이 데이터를 맵핑하는 언어 구조 내에 텍스트 필드는 **LOCALCCSID** 또는 **CCSID** 매개변수에 지정한 것과 동일한 코드 페이지인 EBCDIC 코드 페이지를 사용해야 합니다. DFHWS2LS를 사용하여 웹 서비스 바인딩 파일을 생성하는 경우, 완료 데이터 구조의 파트를 보유하는 채널에서 많은 컨

데이터가 있을 수 있습니다. 이런 경우에는 각각의 해당 컨테이너에 있는 텍스트 필드를 동일한 코드 페이지 사용에 대해 작성하고 읽어야 합니다.

애플리케이션 프로그램이 SOAP 메시지로 변환될 컨테이너를 채우는 경우, 애플리케이션은 컨테이너에 정확한 양의 콘텐츠가 있다는 것을 확인할 책임이 있습니다. 컨테이너가 예상한 것보다 더 적은 데이터를 보유하는 경우, CICS는 변환 오류를 실행합니다.

애플리케이션 프로그램이 **INVOKE SERVICE** 명령을 사용하는 경우, CICS로 전달하는 컨테이너가 잠재적으로 재사용되고 컨테이너 내의 데이터가 대체됩니다. 이러한 컨테이너에서 데이터를 유지하려는 경우, 새 채널을 작성하고 프로그램을 실행하기 전에 컨테이너를 복사하십시오. 또한 요청자 모드 웹 서비스인 제공자 모드 웹 서비스를 가지는 경우, 원래 첨부된 기본 채널을 사용하는 것보다 **INVOKE SERVICE** 명령을 사용할 때 다른 채널을 사용하는 것이 권장됩니다. 사용자 애플리케이션 프로그램이 **INVOKE SERVICE** 명령을 여러 번 사용 중인 경우, CICS에 대한 각 호출에서 다른 채널을 사용하거나 첫 번째 요청의 중요한 모든 데이터가 두 번째 요청을 하기 전에 저장되는지 확인하는 것이 권장됩니다.

웹 서비스 설명 준수

웹 서비스 설명은 선택적으로 SOAP 메시지의 가능한 일부 콘텐츠에 대해 설명할 수 있습니다. 이 경우, DFHWS2LS는 생성된 언어 구조 내의 필드를 콘텐츠가 존재하는지 여부를 표시하기 위해 할당합니다. 따라서, 런타임 시 CICS는 이러한 필드를 채웁니다. 정보가 존재하지 않는다는 것을 필드(예: 존재 플래그 또는 발생 필드)가 표시하는 경우, 애플리케이션 프로그램은 해당 선택 콘텐츠와 연관된 필드를 처리하려고 시도하지 않아야 합니다.

CICS가 이를 전송할 때 SOAP 메시지가 일부 해당 콘텐츠를 누락하는 경우, 애플리케이션 프로그램으로 전달될 때 데이터 구조 내의 동등한 필드가 초기화되지 않습니다.

웹 서비스 설명은 SOAP 메시지를 읽고 CICS가 런타임 시 이러한 규칙을 구현할 때 사용할 공백 처리 규칙을 지정할 수도 있습니다.

- `xsd:whiteSpace` 패스 값이 `replace`인 경우, 공백 문자(예: 『tab』 및 『carriage return』)가 공백으로 대체됩니다.
- `xsd:whiteSpace` 패스 값이 `collapse`인 경우, SOAP 메시지를 생성할 때 후미 공백 문자가 제거됩니다. 런타임 시, 인바운드 SOAP 메시지는 XML 스키마 스펙에 따라 구문 분석되고 모든 선두, 후미 및 임베디드 공백이 제거됩니다.

SOAP 메시지

CICS는 SOAP 메시지 콘텐츠 도출을 지원하지 않습니다. 예를 들어, SOAP 메시지는 요소를 설명하는 스키마의 위치를 지정하기 위한 `xsi:schemaLocation` 속성과 함께 `xsi:type` 속성을 사용하여 요소가 특정 유형을 가지도록 지정할 수 있습니다. CICS는 스키마의 콘텐츠를 기반으로 요소 값을 전송하고 스키마를 동적으로 검색하는 기능

을 지원하지 않습니다. CICS는 웹 서비스 지원에 설정되는 맵핑 레벨이 1.1 이상이지만, 지원되는 유일한 XML 스키마 인스턴스 속성일 때 `xsi:nil` 속성을 지원합니다.

DFHWS2LS는 SOAP 메시지에서 일부 값의 크기 또는 최대 길이에 대해 가정해야 합니다. 예를 들어, XML 스키마가 `xsd:string`에 대한 최대 길이를 지정하지 않으면 DFHWS2LS는 최대 길이가 255자이고 따라서 언어 구조를 생성한다고 가정합니다. DFHWS2LS에서 **DEFAULT-CHAR-MAXLENGTH** 매개변수를 사용하여 이 값을 변경할 수 있습니다. 런타임 시, CICS가 언어 구조에서 할당되는 공간 보다 큰 값을 가진 SOAP 메시지를 생성하는 경우, CICS가 변환 오류를 실행합니다.

CICS가 서비스 제공자인 경우 SOAP 결함 메시지가 요청자로 리턴됩니다. CICS가 서비스 요청자인 경우, 적당한 RESP2 코드가 **INVOKE SERVICE** 명령으로부터 리턴됩니다.

일부 문자는 < 및 > 문자와 같은 XML에서 특수 의미를 갖습니다. 런타임 시 CICS가 처리하는 문자 배열 내에 이러한 특수 문자가 표시되면 이는 동등한 엔티티로 대체됩니다. 지원되는 XML 엔티티는 다음과 같습니다.

문자	XML 엔티티
&	&
<	<
>	>
"	"
'	'

CICS는 공백 코드에 사용되는 숫자 문자 참조의 정규형을 지원합니다.

문자	XML 엔티티
탭		
캐리지 리턴	

줄 바꾸기	

이 지원은 호출되는 파이프라인 핸들러 프로그램으로 확장되지 않는다는 것을 참고하십시오.

널 문자(`x'00'`)는 XML 문서에서 올바르지 않습니다. 애플리케이션 프로그램이 제공하는 문자 유형 필드에 널 문자가 포함되는 경우, CICS는 해당 위치에서 데이터를 자릅니다. 이를 통해 널 종료 문자열로 문자 배열을 처리할 수 있습니다. `base64Binary`의 DFHWS2LS 또는 `hexBinary` XML 스키마 데이터 유형이 생성하는 문자 유형 필드는 2진 데이터를 나타내고 자르지 않고 널 문자를 포함할 수 있습니다.

주의: CICS는 구조화된 애플리케이션 데이터로부터 XML 및 JSON 데이터를 생성합니다. 해당 애플리케이션 데이터가 사전 형식화된 XML, JSON, HTML, JPEG 이미지 처럼 보이는 비트 패턴 또는 실제로 기타 유의미한 콘텐츠 유형을 포함하는 경우, CICS

가 이 시맨틱 의미를 인식하지 못하고 해당 데이터를 일반 텍스트 또는 2진 데이터처럼 처리합니다. CICS는 데이터 내에서 패턴을 인식하거나 인코딩된 데이터를 다르게 처리하려고 시도하지 않습니다. 예를 들어, 데이터가 사전 형식화된 XML(예: CDATA 인코딩 텍스트)을 포함하는 경우, 해당 데이터가 기타 데이터와 동일한 방식으로 처리됩니다. "An example: <here>"과 같은 애플리케이션 데이터가 있다고 가정해 보십시오. 이 예제 애플리케이션에서 제공하는 데이터는 XML 태그처럼 보이는 내용을 포함하나 원시 텍스트로 처리되어 "An example: < here >"과 같은 XML 표시가 발생합니다. 애플리케이션이 XML 자체를 생성해야 하는 경우, XML 스키마에 xsd:any 구조를 사용하거나 XML-ONLY=TRUE를 보조적으로 사용하는 방법을 고려하십시오.

SOAP 결합 메시지

CICS가 서비스 제공자이고 애플리케이션 프로그램이 SOAP 결합 메시지를 실행하도록 하려는 경우, **SOAPFAULT CREATE** 명령을 사용하십시오. 이 API 명령을 사용하려면 웹 서비스 지원 **PGMINT** 매개변수에 CHANNEL의 값을 갖도록 지정해야 합니다. 이 값을 지정하지 않고 애플리케이션 프로그램이 **SOAPFAULT CREATE** 명령을 호출하는 경우, CICS는 SOAP 응답 메시지를 생성하려고 시도하지 않습니다.

파이프라인 처리 사용자 정의

자체 메시지 핸들러 제공에 추가하여 일련의 글로벌 사용자 종료점(GLUE)을 사용하여 파이프라인에서 인바운드 및 아웃바운드 웹 서비스에 대해 발생하는 처리를 사용자 정의할 수 있습니다.

시작하기 전에

파이프라인을 사용자 정의하기 전에 글로벌 사용자 종료 프로그램을 작성하기 위한 우수 사례를 이해해야 합니다. 서비스 제공자 파이프라인을 사용자 정의하는 경우 파이프라인에서 제공된 DFHPITP 또는 Axis2 애플리케이션 핸들러를 사용하고 있어야 합니다.

이 태스크 정보

파이프라인 도메인 종료를 사용하여 웹 서비스 제공자 파이프라인, 웹 서비스 요청자 파이프라인 또는 보안 핸들러를 포함하는 웹 서비스 요청자 파이프라인의 컨테이너에 액세스할 수 있습니다. 파이프라인 글로벌 사용자 종료에 대한 세부사항은 *CICS 사용자 정의 안내서*에 설명되어 있습니다.

프로시저

1. 사용할 글로벌 사용자 종료점을 선택하십시오.
 - XWSPRRWI, XWSPRROI, XWSPRROO 또는 XWSPRRWO GLUE를 사용하여 웹 서비스 제공자 파이프라인의 컨테이너에 액세스합니다.

- XWSRQRWO, XWSRQROO, XWSROROI 또는 XWSRQRWI GLUE를 사용하여 웹 서비스 요청자 파이프라인의 컨테이너에 액세스합니다.
 - XWSSRRWO, XWSSRROO, XWSSRROI 또는 XWSSRRWI GLUE를 사용하여 보안 웹 서비스 요청자 파이프라인의 컨테이너에 액세스합니다.
2. DFH\$PIEX 샘플 종료 프로그램을 사용하여 자체 글로벌 사용자 종료 프로그램을 작성하십시오. DFH\$PIEX는 SDFHSAMP 라이브러리에 있습니다. 프로그램 threadsafe의 작성이 권장됩니다.
 3. 글로벌 사용자 종료 프로그램을 사용으로 설정하십시오.
 4. 글로벌 사용자 종료 프로그램을 테스트하여 올바르게 작동하는지 확인하십시오.

관련 정보:

파이프라인 도메인 종료

파이프라인 샘플 종료 프로그램

글로벌 사용자 종료 프로그램 작성

종료 프로그램 정의, 사용, 사용 안함

요청자 파이프라인 처리 제어를 위한 옵션

서비스 요청자 파이프라인에서 메시지 핸들러는 웹 서비스 요청이 URI를 변경하여 전송되는 위치를 판별할 수 있습니다. CICS에서는 파이프라인 처리 웹 서비스가 요청하는 방식으로 더 많은 유연성을 가지도록 다른 URI 형식에 대한 지원을 제공합니다.

서비스 요청자 파이프라인이 해당 처리의 끝에 도달할 때 다음 옵션을 갖습니다.

프로그램에 링크

`cics://PROGRAM/program` 형식으로 URI를 변경하는 경우, 여기서 *program*은 대상 애플리케이션 프로그램의 이름이고 CICS는 현재 채널 및 해당 컨테이너 또는 COMMAREA를 **EXEC CICS LINK** 명령을 사용하여 프로그램에 전달합니다.

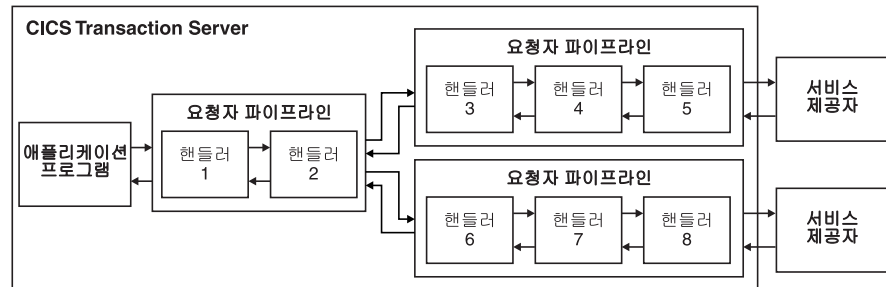
이 처리는 서비스 요청자 및 서비스 제공자 애플리케이션이 동일한 CICS 리전에 있을 때 발생하는 로컬 최적화와 유사합니다. 그러나, 이 URI 형식 사용은 파이프라인 및 사용자 정의 메시지 핸들러를 통해 먼저 요청을 실행하는 이점을 제공합니다. 대상 애플리케이션 프로그램은 컨테이너 또는 COMMAREA의 콘텐츠를 처리할 수 있어야 합니다.

다른 요청자 모드 파이프라인 시작

`cics://PIPELINE/pipeline?targetServiceUri=targetServiceUri` 형식으로 URI를 변경하는 경우, 여기서 *pipeline*은 PIPELINE 자원의 이름이고 *targetServiceUri*는 DFHWS-URI 컨테이너에 넣으려는 URI이고 CICS는 현재 채널 및 해당 컨테이너를 지정된 요청자 파이프라인에 전달합니다. 서비스

제공자에 요청을 전송하기 전에 둘 이상의 요청자 파이프라인에 링크하려고 할 때 이 URI를 사용하십시오. 함께 연결될 수 있는 요청자 파이프라인의 수는 제한되지 않습니다.

다음 예제에서 하나의 일반 요청자 파이프라인은 하나의 애플리케이션을 지원합니다. 메시지 핸들러 1 또는 2는 다른 메시지 핸들러를 포함하는 두 개의 요청자 파이프라인 중 하나에 요청을 전송하여 컨테이너에 있는 애플리케이션 데이터에 따라 각 요청에 대한 URI를 변경할 수 있습니다.



예제가 단지 하나의 서비스 요청자 애플리케이션을 표시한다고 해도, 많은 애플리케이션은 동일한 일반 요청 파이프라인을 사용하고 요청이 마침내 적합한 웹 서비스 제공자로 전송되기 전에 요청이 다른 요청자 파이프라인으로 전송되도록 할 수 있습니다.

제공자 모드 파이프라인에 일직선형으로 요청 보내기

URI를 `cics://SERVICE/service?targetServiceUri=targetServiceUri` 형식으로 변경하는 경우, 여기서 `service`는 대상 서비스의 이름이고 `targetServiceUri`는 서비스에 대한 대상이고 CICS는 URIMAP에 대한 경로를 일치하여 요청을 해결하고 요청을 올바른 제공자 파이프라인에 전달합니다. 네트워크를 사용하지 않고 요청자와 제공자 파이프라인 모두를 통해 요청을 처리하는 데 사용하려고 할 때 이 옵션을 사용하십시오.

이 URI는 요청자와 제공자 애플리케이션이 다른 언어로 작성되거나 다른 맵핑 레벨을 사용하고 다른 2진 데이터를 예상하는 위치에서 유용할 수도 있습니다.

관련 태스크:

388 페이지의 『URI를 사용하여 요청자 파이프라인 처리 제어』

서비스 요청자 파이프라인에서 메시지 핸들러는 URI를 변경하여 웹 서비스 요청을 전송할 위치를 판별할 수 있습니다. URI 형식을 변경하여 특정 최적화 수행(예: 네트워크를 통한 요청 전송 없이 다른 요청자 파이프라인 시작 또는 서비스 제공자 파이프라인 시작)을 선택할 수 있습니다.

관련 참조:

361 페이지의 『DFHWS-URI 컨테이너』

DFHWS-URI는 서비스 URI를 포함하는 DATATYPE(CHAR) 컨테이너입니다.

URI을 사용하여 요청자 파이프라인 처리 제어

서비스 요청자 파이프라인에서 메시지 핸들러는 URI를 변경하여 웹 서비스 요청을 전송할 위치를 판별할 수 있습니다. URI 형식을 변경하여 특정 최적화 수행(예: 네트워크를 통한 요청 전송 없이 다른 요청자 파이프라인 시작 또는 서비스 제공자 파이프라인 시작)을 선택할 수 있습니다.

시작하기 전에

요청자 파이프라인에서 구현할 옵션을 결정하십시오. 세부사항은 386 페이지의 『요청자 파이프라인 처리 제어를 위한 옵션』의 내용을 참조하십시오.

이 태스크 정보

웹 서비스 요청자 애플리케이션은 **EXEC CICS INVOKE SERVICE** 명령을 사용하여 DFHWS-URI 컨테이너를 채울 수 있습니다. 애플리케이션에서 값을 제공하지 않으면 CICS가 웹 서비스 바인딩 파일의 값을 사용하여 컨테이너를 채웁니다. URI를 수정하려면 이 컨테이너의 콘텐츠를 변경하는 메시지 핸들러를 작성해야 합니다.

프로시저

1. 다음 URI 형식 중 하나에 따라 DFHWS-URI 컨테이너를 수정하도록 메시지 핸들러를 작성하십시오.
 - 애플리케이션 프로그램에 링크하려면 URI `cics://PROGRAM/program`를 사용하십시오. 여기서, *program*은 대상 애플리케이션 프로그램입니다. 데이터 변환이 발생하지 않습니다. 따라서 애플리케이션 프로그램이 현재 채널에서 컨테이너의 콘텐츠를 처리할 수 있는지 확인해야 합니다. 애플리케이션 프로그램은 현재 채널 및 컨테이너 또는 COMMAREA를 전달할 수 있습니다.
 - 네트워크를 거치지 않고 제공자 파이프라인을 시작하려면 URI `cics://SERVICE/service?targetServiceUri=targetServiceUri`을 사용하십시오. 여기서, *service*는 서비스의 이름이고 *targetServiceUri*는 서비스의 경로입니다. 전송 핸들러는 서비스 경로를 사용하여 요청을 분석하고 올바른 제공자 파이프라인으로 전달하는 URIMAP 자원을 찾습니다. CICS는 처리 중 서비스의 이름을 사용하지 않습니다.

서비스에 URIMAP 자원이 설치되지 않은 경우 오류가 발생합니다. URIMAP 자원 정의는 USAGE(PIPELINE)도 지정해야 합니다. 전송 핸들러는 **targetServiceUri** 매개변수의 값을 DFHWS-URI 컨테이너에 넣고 제공자 파이프라인을 시작합니다.

- 다른 요청자 파이프라인을 시작하려면 URI `cics://PIPELINE/pipeline?targetServiceUri=targetServiceUri`를 사용하십시오. 여기서, *pipeline*은 시작할 PIPELINE 자원의 이름이고 *targetServiceUri*는 DFHWS-URI 컨테이너에서 다음 파이프라인에 전달할 값입니다.

각 유형의 URI에는 조회 문자열로 추가할 수 있는 추가 매개변수가 있습니다. 이러한 URI의 형식 및 코드화 규칙에 대한 자세한 정보는 361 페이지의 『DFHWS-URI 컨테이너』의 내용을 참조하십시오.

2. XML 편집기를 사용하여 파이프라인 구성 파일에 메시지 핸들러를 추가하십시오.

```
<service>
  <service_handler_list>
    <handler>
      <program>MYPROG</program>
    </handler>
  </service_handler_list>
</service>
```

3. 요청자 파이프라인의 PIPELINE 자원을 사용 안함으로 설정하고, 버리고, 재설치하여 새 메시지 핸들러 프로그램을 파이프라인에 포함시키십시오.
4. CICS 리전에서 메시지 핸들러 프로그램을 설치하십시오.

결과

요청자 파이프라인 전체를 실행하기 위한 다음 서비스 요청은 새 메시지 핸들러에 의해 처리됩니다.

다음에 수행할 작업

요청자 파이프라인에 대한 변경사항을 테스트하여 서비스 요청이 올바른 위치로 이동하는지 및 메시지 핸들러 프로그램이 디자인된 대로 작동하는지 확인하십시오.

관련 개념:

386 페이지의 『요청자 파이프라인 처리 제어를 위한 옵션』

서비스 요청자 파이프라인에서 메시지 핸들러는 웹 서비스 요청이 URI를 변경하여 전송되는 위치를 판별할 수 있습니다. CICS에서는 파이프라인 처리 웹 서비스가 요청하는 방식으로 더 많은 유연성을 가지도록 다른 URI 형식에 대한 지원을 제공합니다.

관련 참조:

361 페이지의 『DFHWS-URI 컨테이너』

DFHWS-URI는 서비스 URI를 포함하는 DATATYPE(CHAR) 컨테이너입니다.

JSON 웹 서비스 제한사항

이 참조 자료를 사용하여 JSON 웹 서비스에서 지원되지 않는 기능을 이해할 수 있습니다.

다음 기능은 지원되지 않습니다.

- JSON이 포함된 요청자 모드 파이프라인은 지원되지 않습니다.

- 스키마에 대한 JSON 데이터의 런타임 유효성 검증이 지원되지 않습니다. JSON 페이로드와 함께 사용되는 WEBSERVICE 자원의 VALIDATION 속성 값은 무시됩니다.
- JSON 데이터(Badgerfish 또는 맵핑된 규칙)에서 네임스페이스 사용은 지원되지 않습니다.
- CICS에 전송된 JSON 페이로드는 UTF-8로 인코딩해야 합니다. 다른 인코딩은 지원되지 않습니다. 마찬가지로, CICS에 의해 전송된 JSON은 항상 UTF-8로 인코딩됩니다.
- JSON 파이프라인을 사용한 WebSphere MQ 전송은 지원되지 않습니다.
- 공급업체 변환기 프로그램을 JSON 변환기에 사용하는 것은 지원되지 않습니다.
- SOAP 웹 서비스 애플리케이션을 위해 JSON 파이프라인에서 작성된 WSBind 파일의 재사용은 지원되지 않습니다. JSON 서비스 제공자 애플리케이션에서 사용할 WSBind 파일은 JSON 지원에 의해 생성되어야 합니다.
- CICS가 JSON 페이로드를 변환할 때 JSON 페이로드가 일부 필수 콘텐츠를 잃는 경우 애플리케이션 프로그램에 전달 시 데이터 구조 내 해당 필드가 초기화되지 않습니다.
- CICS는 따옴표로 묶인 정수 값 외에 부호 있는 긴 정수($2^{63} - 1$)의 최대값을 초과하는 정수 값을 변환할 수 없습니다.
- 단순 데이터 유형 또는 배열의 사용은 JSON 스키마의 루트에서 지원되지 않습니다. JSON 오브젝트는 단순 데이터 유형 및 배열로 구성할 수 있지만 JSON 오브젝트를 설명하는 데 JSON 스키마가 필요합니다.
- maxItems 값이 1인 JSON 스키마에서 배열이 선언되는 경우 런타임 시 JSON을 생성할 때 CICS는 이 배열을 단순 문자열 또는 정수로 직렬화합니다.

중요사항: JSON 특성 이름에 대해 지원되는 유일한 문자는 첫 번째 문자에 대해서는 A-Z a-z _ : 이고 모든 후속 문자에 대해서는 A-Z a-z 0-9 _ : . -입니다.

현재 Axis2 웹 서비스 지원에는 애플리케이션 및 사용자 정의의 개발 및 배치를 위한 다수의 옵션이 있습니다. 다음 옵션은 지원되지 않습니다.

- 사용자 제공 애플리케이션 핸들러 - CICS 제공 애플리케이션 핸들러 클래스 `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler`를 사용해야 합니다.
- 사용자 작성 Axis2 Java 애플리케이션
- SOAPFAULT 및 WS-Addressing API는 JSON 파이프라인에서 사용할 수 없습니다.

컨테이너 제한사항

JSON 요청을 처리할 때 일부 파이프라인 컨테이너는 채워지지 않음에 유의하십시오. 자세한 정보는 구성의 파이프라인에서 사용된 컨테이너의 내용을 참조하십시오.

RESTful 웹 서비스의 차이점

INQUIRE PIPELINE에서

- SOAPLEVEL은 NOTSOAP을 리턴함
- MTOMNOXOPST, MTOMST, SENDMTOMST, SOAPRNUM, SOAPVNUM, XOPDIRECTST 및 XOPSUPPORTST 속성은 사용되지 않습니다.

INQUIRE WEBSERVICE에서

- ARCHIVEFILE, BINDING, VALIDATIONST, XOPDIRECTST 및 XOPSUPPORTST 속성은 사용되지 않습니다.
- WSDLFILE은 WEBSERVICE와 연관된 JSON 스키마 파일의 이름을 리턴합니다.

WEBSERVICE 자원에서

- ARCHIVEFILE 및 VALIDATION 매개변수는 사용되지 않으며 해당 값은 무시됩니다.
- WSDLFILE은 WEBSERVICE와 연관된 JSON 스키마 파일의 이름입니다.

제 10 장 웹 서비스 트랜잭션 지원

웹 서비스 원자적 트랜잭션(또는 WS-AtomicTransaction) 스펙 및 웹 서비스 조정(또는 WS-Coordination) 스펙은 트랜잭션 처리 시스템이 웹 서비스 환경에서 상호 운용될 수 있도록 하는 단기 트랜잭션에 대한 프로토콜을 정의합니다. WS-AtomicTransaction을 사용하는 트랜잭션에는 *ACID* 특성인 원자성, 일관성, 격리 및 내구성이 있습니다.

스펙은 <http://www.ibm.com/developerworks/library/specification/ws-tx/>에서 찾을 수 있습니다.

참고: CICS는 스펙의 2004년 11월 레벨을 지원합니다.

웹 서비스 제공자 또는 요청자로 배치되는 CICS 애플리케이션은 이 스펙을 지원하는 다른 웹 서비스 구현과의 분배 트랜잭션에 참가할 수 있습니다.

등록 서비스

등록 서비스는 조정 프로토콜에 대해 애플리케이션을 등록할 수 있게 하는 WS-Coordination 모델의 해당 파트입니다. 분배 트랜잭션에서 참가 시스템의 등록 서비스는 서로 통신하여 연결된 애플리케이션이 해당 프로토콜에 참가할 수 있게 합니다.

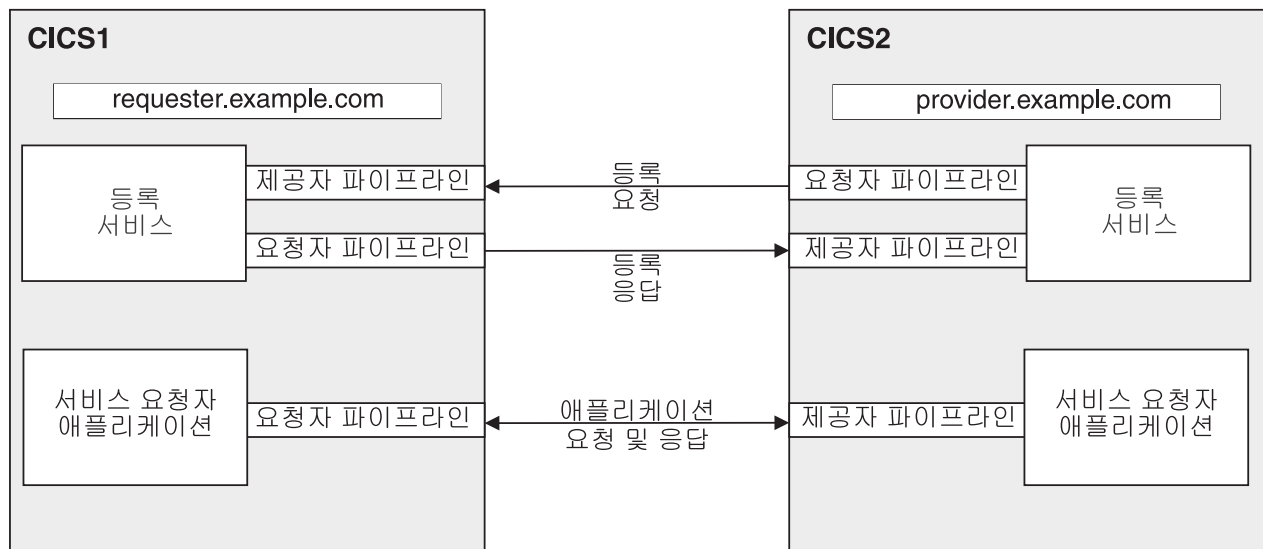


그림 34. 등록 서비스

392 페이지의 그림 29에서는 두 CICS 시스템인 CICS1 및 CICS2를 보여줍니다. CICS1의 서비스 요청자 애플리케이션은 CICS2에서 서비스 제공자 애플리케이션을 호출합니다. 두 CICS 리전 및 애플리케이션은 두 애플리케이션이 WS-Coordination 프로토콜을 사용하여 단일 분배 트랜잭션에 참가할 수 있도록 구성됩니다. 서비스 요청자 애플리케이션은 조정자이고 서비스 제공자 애플리케이션은 참가자입니다.

이러한 프로토콜의 지원에서 두 CICS 리전의 등록 서비스는 트랜잭션 시작 시 상호 작용하고 트랜잭션 종료 중 다시 상호 작용합니다. 이러한 상호작용 중 두 리전의 등록 서비스는 서비스 제공자 및 요청자로 서로 다른 시간에 작동할 수 있습니다. 따라서 각 리전에서 등록 서비스는 서비스 제공자 파이프라인 및 서비스 요청자 파이프라인을 사용합니다. 파이프라인은 PIPELINE 및 연관된 자원을 사용하여 CICS에 정의됩니다.

각 리전의 등록 서비스는 엔드포인트 주소와 연관됩니다. 따라서 이 예제에서 CICS1의 등록 서비스 엔드포인트 주소는 `requester.example.com`이고 CICS2의 등록 서비스 엔드포인트 주소는 `provider.example.com`입니다.

CICSplex에서 등록 서비스 제공자 파이프라인을 다른 리전에 분배할 수 있습니다. 이는 393 페이지의 그림 30에 표시되어 있습니다.

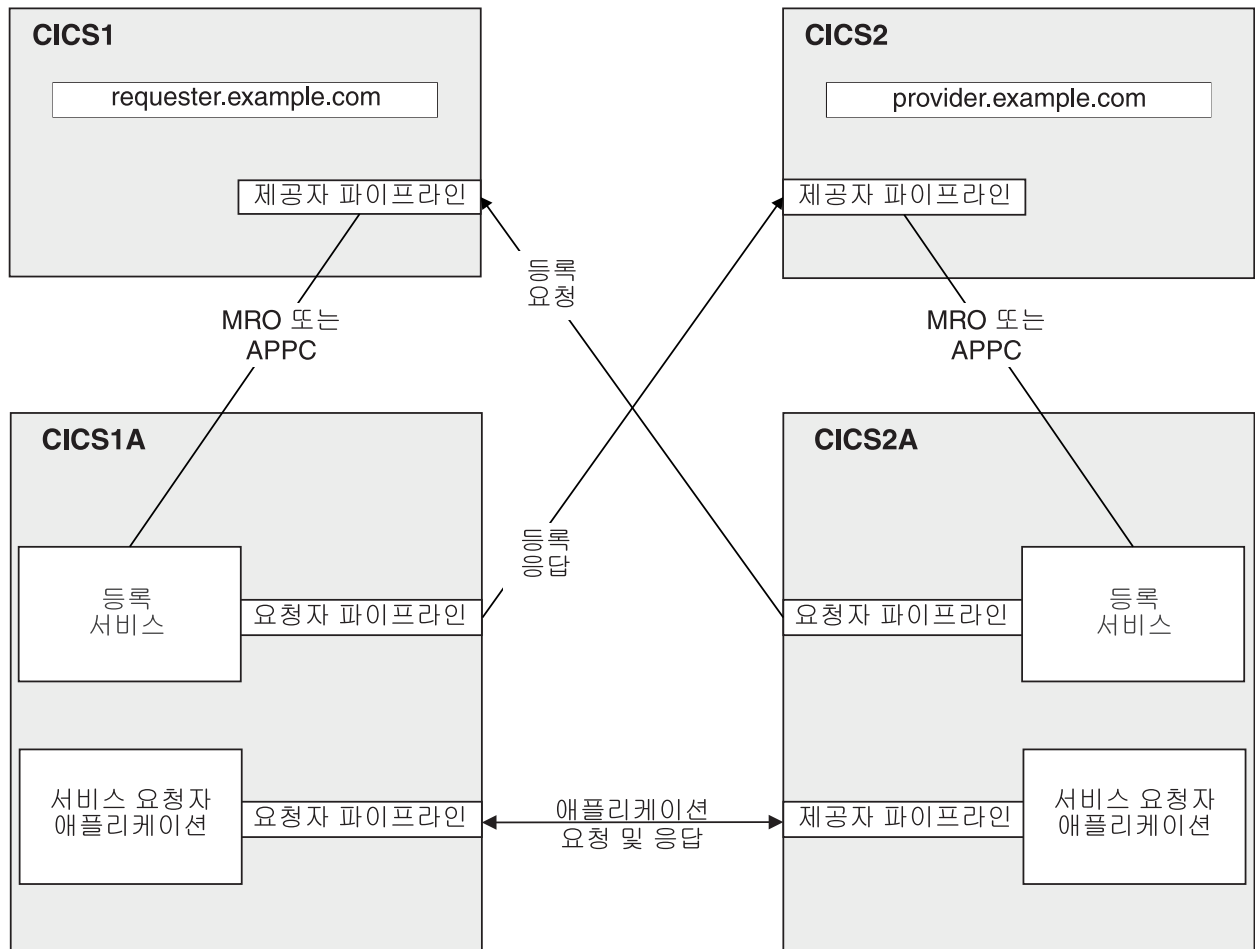


그림 35. CICSplex의 등록 서비스

이 구성에서 제공자 파이프라인은 MPO 또는 APPC를 사용하여 등록 서비스와 통신합니다. 등록 서비스 요청자 파이프라인은 애플리케이션의 요청자 파이프라인과 동일한 리전에 남아 있어야 합니다.

이 구성은 서비스 요청자 및 제공자 애플리케이션이 다수의 리전에 분배되는 경우에 유용합니다. 웹 서비스 트랜잭션에 참가하도록 애플리케이션의 파이프라인을 구성하는 경우 등록 서비스 제공자 파이프라인의 IP 주소 및 포트 번호를 제공하여 등록 서비스 엔드포인트에 대한 정보를 제공해야 합니다. 한 개의 엔드포인트를 가짐으로써 구성을 단순화할 수 있습니다. 모든 파이프라인에 같은 정보가 포함되기 때문입니다. 예를 들어, 393 페이지의 그림 30에서 애플리케이션 요청자 파이프라인에서 지정하는 IP 주소는 `requester.example.com`입니다.

같은 인수가 서비스 제공자 애플리케이션에 적용됩니다. 예제에서 제공자 애플리케이션의 파이프라인은 `requester.example.com`의 IP 주소를 지정합니다.

웹 서비스 트랜잭션을 위한 CICS 구성

웹 서비스 트랜잭션에 웹 서비스 요청자 및 제공자 애플리케이션을 참가시키려면 그에 따라 여러 CICS 자원을 설치하여 CICS를 구성해야 합니다.

시작하기 전에

이러한 자원을 설치하려면 먼저 웹 서비스 트랜잭션의 지원에서 CICS가 제공하는 파이프라인 구성 파일의 위치를 알아야 합니다. 기본적으로 구성 파일은 `/usr/lpp/cicsts/cicsts53/pipeline/configs` 디렉토리에 제공되지만 기본 파일 경로는 CICS 설치 중 변경되었을 수도 있습니다.

이 태스크 정보

웹 서비스 트랜잭션을 위한 CICS 지원은 CICS 제공 등록 서비스를 사용합니다. 이 등록 서비스는 서비스 제공자 및 서비스 요청자로 구성됩니다. 애플리케이션이 모두 서비스 제공자이거나 서비스 요청자인 경우에도 서비스 제공자 및 서비스 요청자 둘 다에 대한 자원을 설치해야 합니다.

또한 서비스 제공자 및 요청자 애플리케이션을 실행할 때 호출되는 헤더 핸들러 프로그램의 프로그램 정의를 설치해야 합니다.

웹 서비스 트랜잭션을 위해 CICS를 구성하는 데 필요한 자원은 모두 DFHWSAT 그룹에서 제공되지만 DFHPIVS, DFHPIVR 또는 DFHPICF 그룹 중 하나에서 제공되는 DFHPIDIR은 제외됩니다. DFHWSAT 그룹은 DFHLIST 목록에 포함되지 않기 때문에 자동으로 설치되지 않습니다. DFHWSAT 그룹에서 CICS 제공 자원은 변경할 수 없습니다.

웹 서비스 트랜잭션을 위해 CICS를 구성하려면 다음을 수행하십시오.

프로시저

1. DFHPIDIR 데이터 세트를 추가하여 JCL을 시작하십시오. DFHPIDIR은 컨텍스트와 태스크 간의 맵핑을 저장합니다.
 - a. DFHPIDIR 데이터 세트의 새 DD 명령문을 CICS 시작 JCL에 추가하십시오.
 - b. DFHDEFDS.JCL의 정보를 사용하여 DFHPIDIR 데이터 세트를 작성하십시오. DFHPIDIR의 기본 RECORDSIZE는 1KB이며 이 값은 대부분의 경우에 적합합니다. 필요한 경우 더 큰 RECORDSIZE를 사용하여 DFHPIDIR을 작성할 수 있습니다.
 - c. CICS 시스템에 DFHPIVS, DFHPIVR 또는 DFHPICF 중 적합한 데이터 세트 그룹을 설치하십시오. 이러한 그룹에 대한 자세한 정보는 WS-AT 데이터 세트 정의를 참조하십시오.

CICS 리전 사이에서 DFHPIDIR 파일을 공유하려는 경우 리전은 MRO를 통해 논리적으로 연결되어 있어야 합니다. 논리 서버 역할을 하는 리전 그룹당 한 개의 데이터 세트를 설치해야 합니다.

팁: 논리적으로 연결되지 않은 리전 간 데이터 세트 공유는 권장되지 않습니다.

2. DFHWSAT 그룹의 콘텐츠를 다른 그룹에 복사하십시오. DFHWSAT 그룹에서 CICS 제공 자원은 변경할 수 없습니다. 그러나 PIPELINE 자원에서 CONFIGFILE 속성은 변경해야 합니다.
3. 등록 서비스의 제공자 PIPELINE 자원을 수정하십시오. PIPELINE은 DFHWSATP로 이름 지정되고 CONFIGFILE 속성에서 파이프라인 구성 파일 /usr/lpp/cicsts/cicsts53/pipeline/configs/registrationservicePROV.xml을 지정합니다.
 - a. 시스템에서 파일의 위치를 반영하도록 CONFIGFILE 속성을 변경하십시오.
 - b. 다른 속성은 변경하지 않는 상태로 두십시오.

제공된 그대로 파이프라인 구성 파일을 사용하고 그 콘텐츠를 변경하지 마십시오.

4. PIPELINE 자원을 설치하십시오. 등록 서비스 제공자 PIPELINE 자원은 서비스 요청자 또는 제공자 애플리케이션과 같은 CICS 리전에 있지 않아도 되지만 적절한 MRO 또는 APPC 연결을 사용하여 해당 리전과 연결되어야 합니다.
5. 등록 서비스 제공자에서 사용되는 URIMAP를 변경하지 않고 PIPELINE과 같은 리전에 설치하십시오. 이 URIMAP은 DFHRSURI로 이름 지정됩니다.
6. 등록 서비스의 요청자 PIPELINE 자원을 수정하십시오. PIPELINE은 DFHWSATR로 이름 지정되고 CONFIGFILE 속성에서 파이프라인 구성 파일 /usr/lpp/cicsts/cicsts53/pipeline/configs/registrationserviceREQ.xml을 지정합니다.
 - a. 시스템에서 파일의 위치를 반영하도록 CONFIGFILE 속성을 변경하십시오.
 - b. 다른 속성은 변경하지 않는 상태로 두십시오.

제공된 그대로 파이프라인 구성 파일을 사용하고 그 콘텐츠를 변경하지 마십시오.

7. PIPELINE 자원을 설치하십시오. 등록 서비스 요청자 PIPELINE 자원은 서비스 요청자 및 제공자 애플리케이션과 같은 CICS 리전에 설치해야 합니다.
8. 등록 서비스 제공자 파이프라인에서 사용되는 프로그램을 PIPELINE 자원과 같은 리전에 설치하십시오. 이러한 프로그램은 DFHWSATX, DFHWSATR 및 DFHPIRS입니다. 두 PIPELINE 자원이 서로 다른 리전에 있는 경우 이러한 프로그램을 두 리전 모두에 설치해야 합니다.
9. 헤더 핸들러 프로그램의 PROGRAM 자원 정의를 설치하십시오. 이 프로그램은 DFHWSATH로 이름 지정됩니다. 서비스 제공자 및 요청자 애플리케이션이 실행되는 리전에 PROGRAM을 설치하십시오.

결과

서비스 제공자 및 요청자 애플리케이션이 WS-AtomicTransaction 및 WS-Coordination 프로토콜을 사용하여 분배 트랜잭션에 참가할 수 있도록 CICS가 구성되었습니다.

다음에 수행할 작업

이제 각 참가 애플리케이션을 개별적으로 구성해야 합니다.

웹 서비스 트랜잭션을 위한 서비스 제공자 구성

서비스 제공자 애플리케이션이 웹 서비스 트랜잭션에 참가하는 경우 파이프라인 구성 파일은 <headerprogram> 요소 및 <service_parameter_list> 요소를 지정해야 합니다.

시작하기 전에

서비스 제공자 애플리케이션을 웹 서비스 트랜잭션에 참가시키려는 경우 이 애플리케이션은 SOAP 프로토콜을 사용하여 서비스 요청자와 통신해야 하고 파이프라인이 CICS 제공 SOAP 메시지 핸들러 중 하나를 사용하도록 구성해야 합니다. 서비스 제공자 애플리케이션을 올바르게 구성했다고 해도 이 애플리케이션은 요청자 애플리케이션이 참가하도록 설정된 경우에만 서비스 요청자와의 웹 서비스 트랜잭션에 참가합니다.

이 태스크 정보

사용자 애플리케이션에 특정한 파이프라인 구성 정보 외에도 구성 파일은 애플리케이션이 웹 서비스 트랜잭션에 참가하도록 하기 위해 CICS가 사용하는 정보를 포함해야 합니다.

CICS는 파일 /usr/lpp/cicsts/cicsts53/samples/pipelines/wsatprovider.xml 디렉토리(여기서, /usr/lpp/cicsts/cicsts53는 z/OS UNIX에서 CICS 파일의 기본 설치 디렉토리임)에 이 정보를 포함하는 파이프라인 구성 파일의 예제를 제공합니다.

프로시저

1. 터미널 핸들러의 정의에서 <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소의 <headerprogram> 요소를 코드화하십시오. <program_name>, <namespace>, <localname> 및 <mandatory> 요소를 이 예제에 표시된 그대로 코드화하십시오.

```
<terminal_handler>
  <cics_soap_1.1_handler>
    <headerprogram>
      <program_name>DFHWSATH</program_name>
      <namespace>http://schemas.xmlsoap.org/ws/2004/10/wscoord</namespace>
```

```

        <localname>CoordinationContext</localname>
        <mandatory>false</mandatory>
    </headerprogram>
</cics_soap_1.1_handler>
</terminal_handler>

```

애플리케이션이 필요로 하는 경우 다른 <headerprogram> 요소를 포함시키십시오.

2. <service_parameter_list>에 있는 <registration_service_endpoint> 요소를 코드화하십시오. 다음과 같이 <registration_service_endpoint>를 코드화하십시오.

```

<registration_service_endpoint>
http://address:port/cicswsat/RegistrationService
</registration_service_endpoint>

```

*address*는 등록 서비스 제공자 파이프라인이 있는 CICS 리전의 IP 주소입니다.

*port*는 등록 서비스 제공자 파이프라인에서 사용되는 포트 번호입니다.

그 나머지도 표시된 그대로 코드화하십시오. 문자열 *cicswsat/*

*RegistrationService*는 URIMAP DFHRSURI의 PATH 속성과 일치합니다.

```

<registration_service_endpoint>
http://provider.example.com:7160/cicswsat/RegistrationService
</registration_service_endpoint>

```

웹 서비스 트랜잭션을 위한 서비스 요청자 구성

서비스 요청자 애플리케이션이 웹 서비스 트랜잭션에 참가하는 경우 파이프라인 구성 파일은 <headerprogram> 요소 및 <service_parameter_list> 요소를 지정해야 합니다.

시작하기 전에

서비스 요청자 애플리케이션을 웹 서비스 트랜잭션에 참가시키려는 경우 이 애플리케이션은 SOAP 프로토콜을 사용하여 서비스 제공자와 통신해야 하고 파이프라인이 CICS 제공 SOAP 메시지 핸들러 중 하나를 사용하도록 구성해야 합니다. 서비스 요청자 애플리케이션을 올바르게 구성했다고 해도 이 애플리케이션은 제공자 애플리케이션이 참가하도록 설정된 경우에만 서비스 제공자와의 웹 서비스 트랜잭션에 참가합니다.

이 태스크 정보

사용자 애플리케이션에 특정한 파이프라인 구성 정보 외에도 구성 파일은 애플리케이션이 웹 서비스 트랜잭션에 참가하도록 하기 위해 CICS가 사용하는 정보를 포함해야 합니다.

CICS는 파일 */usr/lpp/cicsts/cicsts53/samples/pipelines/wsatrequester.xml* 디렉토리(여기서, */usr/lpp/cicsts/cicsts53*는 z/OS UNIX

에서 CICS 파일의 기본 설치 디렉토리임)에 이 정보를 포함하는 파이프라인 구성 파일의 예제를 제공합니다.

프로시저

1. <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소의 <headerprogram> 요소를 코드화하십시오. <program_name>, <namespace>, <localname> 및 <mandatory> 요소를 다음 예제에 표시된 그대로 코드화하십시오.

```
<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>
```

애플리케이션이 필요로 하는 경우 다른 <headerprogram> 요소를 포함시킬 수 있습니다.

2. <service_parameter_list>에 있는 <registration_service_endpoint> 요소를 코드화하십시오. 다음과 같이 <registration_service_endpoint>를 코드화하십시오.

```
<registration_service_endpoint>
http://address:port/cicswsat/RegistrationService
</registration_service_endpoint>
```

*address*는 등록 서비스 제공자 파이프라인이 있는 CICS 리전의 IP 주소입니다.

*port*는 등록 서비스 제공자 파이프라인에서 사용되는 포트 번호입니다.

시작 <registration_service_endpoint> 요소, 해당 콘텐츠 및 종료 <registration_service_endpoint> 요소 사이에는 공백이 없어야 합니다. 이 예제에서는 명료성을 위해 공백이 포함되어 있습니다.

3. CICS가 같은 작업 단위에 있는 요청에 같은 트랜잭션 컨텍스트를 사용하는 대신 각 요청에 대해 새 트랜잭션 컨텍스트를 작성하게 하려는 경우

```
<service_parameter_list>에 있는 비어 있는 요소
<new_tx_context_required/>를 파이프라인 구성 파일에 추가하십시오.

<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> 요소의 시작, 해당 콘텐츠 및 <registration_service_endpoint> 요소의 종료 사이에는 공백이 없어야 합니다. 이 예제에서는 명료성을 위해 공백이 포함되었습니다.

<new_tx_context_required/> 설정은 CICS의 기본값이 아니며 예제 파이프라인 구성 파일 wsatprovider.xml에 포함되지 않습니다. <service_parameter_list>에 있는 <new_tx_context_required/>를 파이프라인 구성 파일에 추가하면 CICS에 대한 루프백 호출이 허용되어 이 상황에서 교착 상태가 발생할 수 있음을 알 수 있습니다.

SOAP 메시지가 원자적 트랜잭션의 파트인지 판별

원자적 트랜잭션 파이프라인에서 CICS 웹 서비스가 호출되면 SOAP 메시지는 원자적 트랜잭션의 파트가 아니어도 됩니다.

이 태스크 정보

SOAP 메시지가 원자적 트랜잭션의 파트인 경우 <soapenv:Header> 요소에는 특정 정보가 포함됩니다. SOAP 메시지가 원자적 트랜잭션의 파트인 경우 다음 중 하나를 수행할 수 있습니다.

프로시저

- 추적을 사용하여 <soapenv:Header> 요소의 콘텐츠 내부를 보십시오.
 - 컴포넌트 PI를 사용하여 보조 추적을 수행하고 추적 레벨을 2로 설정하십시오.
 - 요청 컨테이너에 대한 정보를 포함하는 추적점 PI 0A31을 찾으십시오. 특히 <cicswsa:Action> 요소 바로 앞에 표시되는 PIIS EVENT - REQUEST_CNT를 찾으십시오.
- DFHWSATP 파이프라인에서 사용자 작성 메시지 핸들러 프로그램을 사용하여 DFHREQUEST 컨테이너가 RECEIVE-REQUEST 데이터를 포함할 경우 이 컨테이너의 콘텐츠를 표시하십시오. 이 접근 방식을 선택하는 경우 파이프라인 구성 파일에서 메시지 핸들러 프로그램을 정의해야 합니다.

예

다음 예제는 원자적 트랜잭션의 SOAP 엔벨로프 헤더에서 볼 수 있는 정보를 표시합니다.

```
<soapenv:Header>
  <wscoor:CoordinationContext soapenv:mustUnderstand="1"> 1
    <wscoor:Expires>500</wscoor:Expires>
    <wscoor:Identifier>com.ibm.ws.wstx:
      0000010a2b5008c800000000200000019a75aab901a1758a4e40e2731c61192a10ad6e921
    </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. CoordinationContext는 SOAP 메시지가 원자적 트랜잭션에 참가하도록 계획됨을 표시합니다. CoordinationContext는 조정 서비스의 일부가 될 웹 서비스 제공자의 필요 정보를 포함하고 제공자가 헤더를 인식하고 처리하도록 구성되는 것으로 가정합니다.
2. CoordinationType은 조정 컨텍스트가 준수하는 WS-AT 스펙의 버전을 표시합니다.
3. 조정 RegistrationService는 조정자의 등록점이 있는 위치 및 참가 웹 서비스가 원자적 트랜잭션의 컴포넌트로서 등록을 시도할 때 조정자에게 리턴해야 하는 정보를 설명합니다.

원자적 트랜잭션의 진행상태 확인

CICS 웹 서비스가 원자적 트랜잭션의 파트로 호출되면 트랜잭션은 여러 상태를 통과합니다. 이러한 상태는 트랜잭션이 성공적이었는지 및 롤백해야 되는지 여부를 표시합니다.

이 태스크 정보

이 정보에 액세스해야 하는 경우 다음을 수행할 수 있습니다.

프로시저

- 추적을 사용하여 <cicswsa:Action> 요소의 콘텐츠 내부를 보십시오.
 1. 컴포넌트 PI를 사용하여 보조 추적을 수행하고 추적 레벨을 2로 설정하십시오.
 2. 요청 컨테이너에 대한 정보를 포함하는 추적점 PI 0A31을 찾으십시오. 특히 <cicswsa:Action> 요소 바로 앞에 표시되는 PIIS EVENT - REQUEST_CNT를 찾으십시오.
- DFHWSATR 및 DFHWSATP 파이프라인에서 사용자 작성 메시지 핸들러 프로그램을 사용하여 DFHWS-SOAPACTION 컨테이너의 콘텐츠를 표시하십시오. 이 접근 방식을 선택하는 경우 파이프라인 구성 파일에서 메시지 핸들러 프로그램을 정의해야 합니다.

예

성공적으로 완료되고 커밋된 트랜잭션의 상태는 다음과 같습니다.

```
"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 "
```

롤백되는 트랜잭션의 상태는 다음과 같습니다.

```
"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"
```

제 11 장 2진 데이터의 MTOM/XOP 최적화 지원

표준 SOAP 메시지에서 2진 오브젝트는 base64로 인코딩되며 메시지 본문에 포함됩니다. 메시지 본문에서는 2진 오브젝트 크기가 33% 증가합니다. 아주 큰 2진 오브젝트의 경우에는 이러한 크기 증가로 인해 전송 시간이 큰 영향을 받습니다. MTOM/XOP 구현은 이 문제점의 솔루션을 제공합니다.

SOAP 메시지 전송 최적화 메커니즘(MTOM) 및 XML-2진 최적화 패키징(XOP) 스펙은 MTOM/XOP라고도 하며 SOAP 메시지 내 큰 base64Binary 데이터 오브젝트의 전송을 최적화하는 방법을 정의합니다.

- MTOM 스펙은 2진 데이터를 구분하여(그렇지 않으면 base64로 인코딩됨) SOAP 메시지를 최적화한 후 MIME 멀티파트/관련 메시지를 사용하여 개별 2진 첨부 파일로 보내는 방법을 개념적으로 정의합니다. 이러한 유형의 MIME 메시지를 *MTOM 메시지*라고 합니다. 데이터를 2진 형식으로 보내면 크기가 크게 감소하여 SOAP 메시지 전송이 최적화됩니다.
- XOP 스펙은 MIME 메시지를 포함하되 이에 한정되지 않는 패키징 형식의 2진 첨부 파일을 사용하여 XML 메시지를 최적화하는 구현을 정의합니다.

CICS는 전송 프로토콜이 WebSphere MQ, HTTP 또는 HTTPS인 경우 요청자 및 제공자 파이프라인에서 모두 이 스펙에 대한 지원을 구현합니다. SOAP 메시지에 직접 base64Binary 데이터를 포함하지 않고 웹 서비스 제공자 또는 요청자로 배치된 CICS 애플리케이션이 이 지원을 사용하여 2진 첨부 파일을 포함하는 MTOM 메시지를 송수신할 수도 있습니다.

이 지원은 파이프라인 구성 파일에서 추가 옵션을 사용하여 구성할 수 있습니다.

MTOM/XOP 및 SOAP

MTOM/XOP가 SOAP 메시지를 최적화하기 위해 사용되는 경우 XOP 처리를 사용하여 MIME 멀티파트/관련 메시지로 직렬화됩니다. base64Binary 데이터는 이메일 첨부 파일과 유사한 방식으로 SOAP 메시지에서 추출된 후 MIME 메시지 내 개별 2진 첨부 파일로 패키징화됩니다.

첨부 파일이 2진 형식으로 인코딩되므로 base64Binary 데이터 크기가 크게 감소합니다. SOAP 메시지의 XML은 XOP 형식으로 변환되며 이 때 base64Binary 데이터가 URI를 사용하여 관련 MIME 첨부 파일을 참조하는 특수 `<xop:Include>` 요소로 바뀝니다.

수정된 SOAP 메시지를 *XOP* 문서라고 하며 메시지 내에서 루트 문서를 구성합니다. *XOP* 문서와 2진 첨부 파일이 함께 *XOP* 패키지를 구성합니다. SOAP MTOM 스펙에 적용되는 경우, *XOP* 패키지는 MTOM 형식의 MIME 메시지입니다.

루트 문서는 MIME 메시지의 전체 content-type 헤더에서 해당 Content-ID를 참조하여 식별됩니다. 다음은 content-type 헤더의 예입니다.

```
Content-Type: Multipart/Related; boundary=MIME_boundary;
type="application/soap+xml"; start="<claim@insurance.com>"
```

start 매개변수에는 *XOP* 문서의 Content-ID가 포함됩니다. 이 매개변수가 content-type 헤더에 포함되지 않으면 MIME 메시지의 첫 번째 파트를 *XOP* 문서로 가정합니다.

MIME 메시지에서 첨부 파일의 순서는 중요하지 않습니다. 예를 들어, 일부 메시지에서 2진 첨부 파일이 *XOP* 문서 앞에 나타날 수 있습니다. MIME 메시지를 처리하는 애플리케이션은 특정 순서로 나타나는 첨부 파일에 의존해서는 안 됩니다. 자세한 정보는 MTOM/XOP 스펙을 읽으십시오.

다음 예제는 *XOP* 처리를 사용하여 JPEG 이미지를 포함하는 단순 SOAP 메시지가 최적화되는 방법을 보여줍니다. SOAP 메시지는 다음과 같습니다.

```
<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>
```

이 SOAP 메시지의 MTOM/XOP 버전은 다음과 같습니다.

```
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** 매개변수는 MIME 메시지에서 루트 XOP 문서 파트를 나타냅니다.
2. Content-ID 값은 MIME 메시지의 일부를 식별합니다. 이 경우 루트 XOP 문서입니다.
3. <xop:Include> 요소는 JPEG 2진 첨부 파일을 참조합니다.
4. Content-ID는 2진 첨부 파일에서 JPEG를 식별합니다.

CICS의 MTOM 메시지 및 2진 첨부 파일

CICS는 웹 서비스 제공자 및 요청자 파이프라인에서 모두 MTOM 핸들러 프로그램과 XOP 처리를 사용하여 MTOM 메시지 핸들링을 지원하고 제어합니다.

파이프라인 구성 파일을 사용하여 MTOM 지원을 구성하고 사용하도록 설정할 수 있습니다. 파이프라인에 MTOM 지원을 사용하도록 설정되면 CICS가 인바운드 MTOM 메시지를 자동으로 풀고 아웃바운드 메시지를 패키징화합니다. 파이프라인에 MTOM 지원을 사용하도록 설정되지 않고 CICS가 MTOM 메시지를 수신하면 Java 기반 파이프라인이 인바운드 MTOM 메시지를 승인하지만 다른 SOAP 파이프라인은 SOAP 결함과 함께 인바운드 MTOM 메시지를 거부합니다.

Java 기반 파이프라인에 대한 구성 옵션

제공자 파이프라인이 다음 태스크를 수행하도록 구성할 수 있습니다.

- MTOM 메시지를 승인하지만 MTOM 응답 메시지는 보내지 않습니다.
- MTOM 메시지를 승인하고 MTOM 응답 메시지를 항상 보냅니다.
- XOP 문서와 2진 첨부 파일을 Axis2 모드로 처리합니다.

요청자 파이프라인이 다음 태스크를 수행하도록 구성할 수 있습니다.

- MTOM 메시지를 보내지 않고 MTOM 응답 메시지는 승인합니다.
- MTOM 메시지를 항상 보내고 MTOM 응답 메시지를 승인합니다.
- XOP 문서와 2진 첨부 파일을 Axis2 모드로 처리합니다.

Java를 지원하지 않는 파이프라인에 대한 구성 옵션

제공자 파이프라인이 다음 태스크를 수행하도록 구성할 수 있습니다.

- MTOM 메시지를 승인하지만 MTOM 응답 메시지는 보내지 않습니다.
- MTOM 메시지를 승인하고 동일한 유형의 응답 메시지를 보냅니다.
- MTOM 메시지를 승인하지만 2진 첨부 파일이 있는 경우에만 MTOM 메시지를 보냅니다.
- MTOM 메시지를 승인하고 MTOM 응답 메시지를 항상 보냅니다.

- XOP 문서와 2진 첨부 파일을 직접 또는 호환 모드로 처리합니다.

요청자 파이프라인이 다음 태스크를 수행하도록 구성할 수 있습니다.

- MTOM 메시지를 보내지 않고 MTOM 응답 메시지는 승인합니다.
- 2진 첨부 파일이 있는 경우에만 MTOM 메시지를 보내고 MTOM 응답 메시지를 승인합니다.
- MTOM 메시지를 항상 보내고 MTOM 응답 메시지를 승인합니다.
- XOP 문서와 2진 첨부 파일을 직접 또는 호환 모드로 처리합니다.

지원 모드

파이프라인에는 XOP 문서와 연관된 2진 첨부 파일을 처리하기 위한 세 가지 지원 모드가 제공됩니다.

Axis2 모드

Axis2 모드는 웹 서비스 파이프라인의 터미널 핸들러가

<cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 메시지 핸들러인 경우 사용됩니다.

직접 모드

직접 모드에서는 인바운드 또는 아웃바운드 MTOM 메시지와 연관된 2진 첨부 파일이 파이프라인을 통해 컨테이너에서 전달되며 데이터 변환을 수행할 필요 없이 애플리케이션으로 직접 처리됩니다.

호환 모드

호환 모드는 파이프라인 처리 시 메시지가 표준 XML 형식이어야 하는 경우 사용되며 2진 데이터가 메시지 내에서 base64Binary 필드로 저장됩니다. 인바운드 메시지의 경우, XOP 문서와 2진 첨부 파일이 파이프라인 시작(웹 서비스 보안을 사용하는 경우) 또는 파이프라인 끝(웹 서비스 유효성 검증을 사용하는 경우)에서 표준 XML 메시지로 재구성됩니다. 아웃바운드 메시지의 경우에는 표준 XML 메시지가 작성되어 파이프라인을 따라 전달됩니다. 이 XML 메시지는 CICS가 보내기 바로 전에 MTOM 핸들러에 의해 XOP 형식으로 변환됩니다.

호환 모드는 2진 데이터가 base64 형식으로 변환되었다가 다시 2진 형식으로 변환되므로 직접 모드보다 훨씬 비효율적입니다. 그러나 애플리케이션을 변경하지 않고 웹 서비스가 다른 MTOM/XOP 웹 서비스 요청자 및 제공자와 상호 운용될 수 있습니다.

Java를 지원하지 않는 파이프라인에 대한 인바운드 MTOM 메시지 처리

MTOM 핸들러를 Java를 지원하지 않는 파이프라인에서 사용하는 경우, DFHREQUEST 또는 DFHRESPONSE 컨테이너에서 인바운드 메시지의 헤더를 확인하여 전송 핸들링 처리 중에 메시지 형식을 판별합니다.

MIME Multipart/Related 메시지가 수신되면 MTOM 핸들러가 다음과 같이 메시지 패키지를 해제합니다.

1. 각 2진 첨부 파일의 헤더와 2진 데이터는 개별 컨테이너에 둡니다.
2. 컨테이너 목록은 DFHWS-XOP-IN 컨테이너에 둡니다.
3. 메시지 루트를 구성한 XOP 문서를 다시 DFHREQUEST 또는 DFHRESPONSE 컨테이너에 두어 원래 메시지를 바꿉니다.

2진 첨부 파일이 없는 경우에는 XOP 문서가 보통 XML 메시지로 처리되며 XOP 처리가 필요하지 않습니다. 2진 첨부 파일이 있는 경우에는 메시지에 XOP 처리를 사용합니다.

XOP 처리를 사용하는 경우, MTOM 핸들러가 파이프라인 특성을 확인하여 현재 메시지가 직접 또는 호환 모드에서 처리되어야 하는지 여부를 판별하고 이 정보를 DFHWS-MTOM-IN 컨테이너에 둡니다.

제공자 모드의 경우 MTOM 핸들러가 또한 DFHWS-MTOM-OUT 컨테이너를 작성하여 아웃바운드 응답 메시지 처리 방법을 판별합니다.

직접 모드

CICS 웹 서비스 지원을 사용하는 경우, 즉 서비스 제공자 파이프라인이 DFHPITP 애플리케이션 핸들러를 사용하거나 서비스 요청자 파이프라인이 **INVOKE WEBSERVICE**를 사용하여 호출되는 경우에는 파이프라인이 XOP 문서와 2진 첨부 파일을 직접 모드로 처리할 수 있습니다.

이 모드에서는 XOP 문서와 연관된 컨테이너가 MTOM 핸들러에 의해 파이프라인 내 다음 메시지 핸들러로 전달되어 처리됩니다. CICS 웹 서비스 지원은 <xop:Include> 요소를 해석합니다. base64Binary 필드가 애플리케이션 데이터 구조에서 컨테이너로 표시되는 경우에는 첨부 파일 컨테이너 이름이 해당 구조에 저장됩니다. 필드가 가변 또는 고정 길이 문자열로 표시되는 경우에는 컨테이너의 콘텐츠가 관련 애플리케이션 데이터 구조 필드에 복사됩니다. 데이터 구조는 그런 다음 애플리케이션 프로그램으로 전달됩니다.

호환 모드

파이프라인이 사용자 정의 애플리케이션 핸들러를 사용하도록 구성되거나 웹 서비스 보안 또한 사용하도록 설정되는 경우, 메시지가 호환 모드에서 처리됩니다. 이 모드에서는 XOP 처리를 사용하여 XOP 문서와 2진 첨부 파일이 즉시 SOAP 메시지로 재구성되어 파이프라인에서 콘텐츠를 처리할 수 있습니다. XOP 처리는 다음 태스크를 수행합니다.

1. <xop:Include> 요소의 XOP 문서를 스캔하고 각 발생을 base64 인코딩 형식으로 참조되는 첨부 파일의 2진 데이터로 바꿉니다.

2. DFHWS-XOP-IN 컨테이너와 모든 첨부 컨테이너를 버립니다.

재구성된 SOAP 메시지는 파이프라인의 다음 핸들러로 전달되어 정상적으로 처리됩니다.

웹 서비스 유효성 검증을 사용하도록 설정된 경우에는 메시지가 애플리케이션 핸들러에 도달하면 파이프라인이 호환 모드로 전환됩니다. 메시지는 SOAP 메시지로 재구성되고 유효성을 검증한 후 애플리케이션으로 전달됩니다.

Java를 지원하지 않는 파이프라인에 대한 아웃바운드 MTOM 메시지 처리

Java를 지원하지 않는 파이프라인이 아웃바운드 MTOM 메시지를 보내도록 구성되면 웹 서비스 및 파이프라인 특성을 확인하여 메시지를 처리, 전송하는 방법을 판별합니다.

이 특성은 두 컨테이너(DFHWS-MTOM-OUT 및 DFHWS-XOP-OUT)에 저장됩니다. 요청자 모드 파이프라인의 경우, 이 컨테이너는 애플리케이션이 **EXEC CICS INVOKE WEBSERVICE** 명령을 실행할 때 CICS로 작성됩니다. 제공자 모드 파이프라인의 경우에는 DFHWS-MTOM-OUT 컨테이너가 인바운드 메시지가 수신될 때 판별된 옵션으로 이미 초기화되어 있습니다.

아웃바운드 메시지가 직접 모드에서 처리될 수 있으면 메시지 최적화가 즉시 수행됩니다. 아웃바운드 메시지를 호환 모드로 처리해야 하는 경우, 파이프라인 처리 맨 끝에서 최적화가 수행됩니다.

CICS 웹 서비스 지원을 사용하여 웹 서비스 제공자 또는 요청자 애플리케이션을 배치하지 않았거나 파이프라인에서 웹 서비스 보안 또는 웹 서비스 유효성 검증을 사용하도록 설정한 경우에는 아웃바운드 메시지가 호환 모드로 처리됩니다.

직접 모드

직접 모드에서는 다음 처리가 수행됩니다.

1. 컨테이너 DFHWS-DATA의 애플리케이션 데이터 구조에서 XOP 문서가 구성됩니다. 크기가 1500바이트 이상인 2진 필드가 식별되며 2진 첨부 파일을 설명하는 MIME 헤더와 2진 데이터는 개별 컨테이너에 포함됩니다. 2진 데이터가 이미 컨테이너에 있는 경우에는 해당 컨테이너가 직접 첨부 파일로 사용됩니다. 그런 다음 `<xop:Include>` 요소가 생성된 Content-ID를 사용하여 일반 base64 인코딩 2진 데이터 대신 XML에 삽입됩니다. 예를 들어, 다음과 같습니다.

```
<xop:Include href="cid:generated-content-ID-value"
xmlns:xop="http://www.w3.org/2004/08/xop/include">
```

2. 모든 컨테이너가 DFHWS-XOP-OUT 컨테이너의 첨부 파일 목록에 추가됩니다.
3. SOAP 핸들러가 DFHWS-DATA를 처리하면 XOP 문서와 SOAP 엔벨로프가 DFHREQUEST 또는 DFHRESPONSE 컨테이너에 저장되며 파이프라인을 통해 처리됩니다.

4. 마지막 메시지 핸들러가 완료되면 MTOM 핸들러가 XOP 문서와 2진 첨부 파일을 MIME 멀티파트/관련 메시지로 패키징하여 웹 서비스 요청자 또는 제공자로 보냅니다. DFHWS-XOP-OUT 컨테이너와 연관된 컨테이너는 버려집니다.

호환 모드

파이프라인이 XOP 문서를 직접 핸들링할 수 없는 경우에는 다음과 같은 처리가 수행됩니다.

1. SOAP 본문이 애플리케이션 데이터 구조에서 DFHWS-DATA에 구성되고 파이프라인에서 정상적으로 처리됩니다.
2. 최종 핸들러가 메시지 처리를 완료하면 MTOM 핸들러가 DFHWS-MTOM-OUT 컨테이너의 옵션을 확인하고 선택적으로 2진 첨부 파일이 있는지 여부를 고려하여 MTOM을 사용해야 하는지 여부를 판별합니다. MTOM 핸들러가 MTOM이 필요하지 않은 것으로 판별하면 XOP 처리가 수행되지 않고 CICS가 SOAP 메시지를 정상적으로 보냅니다.
3. MTOM 핸들러가 아웃바운드 메시지를 MTOM 형식으로 보내야 하는 것으로 판별하면 XOP 처리 시 올바른 필드에 대한 메시지를 스캔하여 데이터를 2진 첨부 파일로 분할합니다. 올바른 필드는 요소에 대해 MIME **contentType** 속성이 지정되어야 하며 연관된 2진 값이 표준 양식의 올바른 base64Binary 데이터로 구성되어야 합니다. 데이터 크기는 1500바이트보다 커야 합니다. XOP 처리는 2진 첨부 파일과 첨부 파일 목록을 작성한 다음 필드를 <xop:Include> 요소로 바꿉니다.
4. MTOM 핸들러는 XOP 문서와 2진 첨부 파일을 MIME 멀티파트/관련 메시지로 패키징하고 CICS는 해당 패키지를 웹 서비스 요청자 또는 제공자로 보냅니다.

MTOM/XOP 사용 시 제한사항

MTOM/XOP를 지원하기 위해 파이프라인 구성 파일에서 <mtom> 요소를 지정하거나 파이프라인에서 MTOM 핸들러를 사용하도록 설정할 수 있습니다. 그러나 각 방법과 연관된 제한사항이 있습니다.

Java 기반 파이프라인에 대한 제한사항

파이프라인 구성 파일에서 <mtom> 요소를 지정하면 Java 기반 파이프라인에 MTOM/XOP 지원을 사용할 수 있습니다. 그러나 이 MTOM/XOP 구현에는 제한사항이 있습니다.

DFHPITP 애플리케이션 핸들러

MTOM/XOP 지원의 Axis2 모드는 DFHPITP를 애플리케이션 핸들러로 지정하는 파이프라인에 사용할 수 없습니다.

WS-보안

MTOM/XOP 지원의 Axis2 모드는 XML 서명이 필요한 WS-보안 구성을 사용하는 파이프라인에 사용할 수 없습니다.

INQUIRE PIPELINE 명령 사용

MTOM/XOP 지원의 Axis2 모드를 사용하는 Java 기반 파이프라인에 대해 **INQUIRE PIPELINE** 명령이 실행되는 경우 **Mtomst**, **Sendmtomst**, **Mtomnoxopst**, **Xopsupportst**, **Xopdirectst** 속성은 Nomtom으로 보고됩니다. 자세한 정보는 **INQUIRE PIPELINE**을 참조하십시오.

기타 SOAP 파이프라인에 대한 제한사항

파이프라인에서 MTOM 핸들러를 사용한다는 것은 MTOM/XOP 최적화를 사용하는 웹 서비스 구현을 지원할 수 있음을 의미합니다. 호환 모드 옵션은 웹 서비스 애플리케이션을 변경하지 않고 이 웹 서비스를 상호 운용할 수 있음을 의미합니다. 그러나 MTOM/XOP를 사용할 수 없거나 사용이 제한되는 특정 상황이 있습니다.

CICS 웹 서비스 지원 사용

MTOM/XOP에 대한 직접 모드 최적화는 DFHWS2LS를 맵핑 레벨 1.2 이상에서 사용하고 WSDL 문서에 xsd:base64Binary 유형의 필드가 최소 하나 포함되어 있는 경우에만 사용할 수 있습니다. DFHLS2WS를 사용하여 사용할 수 있는 웹 서비스는 XOP 최적화에 적합하지 않습니다.

CHAR-VARYING=BINARY를 지정한 DFHLS2WS를 사용하여 생성된 웹 서비스는 MTOM/XOP 최적화에 적합할 수도 있습니다. DFHLS2WS를 사용하여 생성된 기타 웹 서비스에는 2진 데이터가 포함되지 않고 MTOM/XOP 최적화에 적합하지 않지만 MTOM/XOP를 지원하는 PIPELINE에서 정상적으로 작동합니다.

제공자 파이프라인

CICS는 제공자 파이프라인에서 구성될 수 있는 기본 애플리케이션 핸들러 DFHPITP를 제공합니다. 이 애플리케이션 핸들러는 XOP 문서를 핸들링할 수 있으며 파이프라인 처리를 직접 및 호환 모드에서 모두 지원하는 데 필요한 컨테이너를 작성할 수 있습니다. 제공자 파이프라인에서 고유한 애플리케이션 핸들러를 사용하고 MTOM/XOP를 사용하도록 설정하려는 경우에는 파이프라인에 호환 모드에서 실행되도록 구성해야 합니다.

요청자 파이프라인

애플리케이션이 **INVOKE WEBSERVICE** 명령을 사용하는 경우 CICS가 직접 및 호환 모드에서 둘 다 SOAP 메시지 최적화를 처리합니다. 프로그램 DFHPIRT를 사용하여 파이프라인을 시작하는 경우에는 호환 모드에서만 MIME 멀티파트/관련 메시지를 송수신할 수 있습니다.

웹 서비스 보안

파이프라인 구성 파일의 MTOM 핸들러가 직접 모드에서 실행되도록 설정하고

웹 서비스 보안 메시지 핸들러도 사용하도록 설정하려는 경우에는 파이프라인이 호환 모드에서의 MTOM 메시지 핸들링만 지원합니다.

2진 데이터 핸들링

대용량 2진 데이터(예를 들어, JPEG와 같은 그래픽 파일)를 웹 서비스에 포함하는 경우 MTOM/XOP를 사용하여 서비스 제공자 또는 요청자로 보내는 메시지의 크기를 최적화할 수 있습니다. MTOM/XOP를 사용하여 최적화될 수 있는 2진 데이터의 최소 크기는 1500바이트입니다. 필드의 2진 데이터가 1500바이트 미만이면 CICS가 필드를 최적화하지 않습니다.

XOP 스펙에 명시된 대로 base64Binary 데이터에는 공백이 없어야 합니다. base64Binary 데이터를 생성하는 애플리케이션 프로그램은 표준 양식을 사용해야 합니다. 아웃바운드 메시지의 base64Binary 데이터에 공백이 포함되면 CICS가 데이터를 2진 첨부 파일로 변환하지 않습니다. base64Binary 데이터가 CICS로 생성되면 필드가 표준 양식으로 제공되므로 공백이 포함되지 않습니다.

아웃바운드 메시지에 대해 XOP 처리가 호환 모드로 발생하려면 base64Binary 필드에 **contentType** 속성이 있어야 합니다. **contentType** 속성이 hexBinary 필드에 있어서는 안 됩니다.

웹 서비스 유효성 검증

웹 서비스 유효성 검증을 활성화하면 다음 파이프라인 처리가 실행됩니다.

- 직접 모드로 파이프라인을 통해 인바운드 XOP 문서가 전달되면 CICS가 자동으로 호환 모드로 전환되며 CICS 웹 서비스 지원이 문서 유효성을 검증하려고 할 때 표준 XML로 다시 변환합니다.
- 아웃바운드 SOAP 메시지가 표준 XML로 생성되고 호환 모드로 처리됩니다.

유효성 검증 처리는 XOP 문서의 콘텐츠를 처리할 수 없으므로 추가 파이프라인 처리가 필요합니다.

CICS에서 MTOM/XOP를 지원하도록 구성

CICS에서 MTOM 메시지를 지원하려면 파이프라인 구성 파일에서 파이프라인 유형에 올바른 MTOM/XOP 지원을 지정해야 합니다.

Java 기반 파이프라인에 대한 MTOM/XOP 지원 구성

Java 기반 파이프라인에 대한 MTOM/XOP 지원을 구성하려면 파이프라인 구성 파일에 `<mtom>` 요소를 추가해야 합니다.

시작하기 전에

이 작업을 수행하기 전에 MTOM/XOP의 구성 정보를 추가할 파이프라인 구성 파일을 식별하거나 작성해야 합니다.

이 작업 정보

<mtom> 요소가 파이프라인 구성 파일에 정의된 경우 모든 인바운드 및 아웃바운드 메시지에 MTOM 지원을 사용할 수 있습니다. 그러나 이 요소가 파이프라인 구성 파일에 지정되지 않으면 인바운드 메시지에만 MTOM 지원을 사용할 수 있습니다.

프로시저

파이프라인 구성 파일에 <mtom> 요소를 추가하십시오. 이 요소는 선택적 <addressing> 요소 뒤와 선택적 <headerprogram> 요소 앞에 정의되어야 합니다.

예제

제공자 또는 요청자 모드 파이프라인의 경우 다음을 지정할 수 있습니다.

```
<cics_soap_1.2_handler_java>
  <jvmserver>JVMSERV1</jvmserver>
  <addressing></addressing>
  <mtom></mtom>
  <headerprogram>
    <program_name>HDRPROG4</program_name>
    <namespace>http://mynamespace</namespace>
    <localname>myheaderblock</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler_java>
```

다른 SOAP 파이프라인에 대한 MTOM/XOP 구성

<cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 핸들러를 사용하지 않는 파이프라인에 대한 MTOM/XOP 지원을 구성하려면 파이프라인 구성 파일에 MTOM 핸들러를 추가해야 합니다.

시작하기 전에

이 작업을 수행하기 전에 MTOM/XOP의 구성 정보를 추가할 파이프라인 구성 파일을 식별하거나 작성해야 합니다.

프로시저

1. 파이프라인 구성 파일에 <cics_mtom_handler> 요소를 추가하십시오. 이 요소는 <provider_pipeline> 요소의 첫 번째 요소이자 <requester_pipeline> 요소에서 <service_parameter_list> 앞의 마지막 요소여야 합니다. 다음 요소를 코딩하십시오.

```
<cics_mtom_handler>
  <dfhmtom_configuration version="1">
    </dfhmtom_configuration>
  </cics_mtom_handler>
```

<dfhmtom_configuration> 요소는 구성 내 다른 요소의 컨테이너입니다. MTOM/XOP 처리의 기본 설정을 승인하려는 경우 다음과 같이 비어 있는 요소를 지정할 수 있습니다.

```
<cics_mtom_handler/>
```

2. 옵션: <mtom_options> 요소를 코드화하십시오. 서비스 제공자 및 서비스 요청자 파이프라인에서 모두 이 요소는 아웃바운드 메시지를 MTOM 메시지로 패키징해야 하는지 여부를 지정합니다.
 - a. 아웃바운드 메시지가 MTOM 메시지로 전송되어야 하는지 여부를 정의하도록 **send_mtom** 속성을 코드화하십시오. 이 속성에 대한 세부사항은 309 페이지의 『<mtom_options> 요소』를 참조하십시오.
 - b. 2진 첨부 파일이 없는 경우 아웃바운드 메시지가 MTOM 메시지로 전송되어야 하는지 여부를 정의하도록 **send_when_no_xop** 속성을 코드화하십시오. 이 속성에 대한 세부사항은 309 페이지의 『<mtom_options> 요소』를 참조하십시오.
3. 옵션: <xop_options> 요소를 **apphandler_supports_xop** 속성으로 코드화하십시오. 애플리케이션 핸들러가 XOP 문서를 직접 핸들링할 수 있는지 여부를 지정합니다. 이 속성을 포함하지 않는 경우 기본값은 <apphandler> 요소가 DFHPITP 또는 다른 프로그램을 지정하는지 여부에 따라 다릅니다. 이 속성에 대한 세부사항은 310 페이지의 『<xop_options> 요소』를 참조하십시오.
4. 옵션: <mime_options> 요소를 **content_id_domain** 속성으로 코드화하십시오. 2진 첨부 파일을 식별하는 데 사용되는 MIME content-ID 값을 생성할 때 사용되어야 하는 도메인 이름을 지정합니다. 이 속성에 대한 세부사항은 312 페이지의 『<mime_options> 요소』를 참조하십시오.

예제

다음 예제는 선택적 요소가 모두 있는 완료된 <cics_mtom_handler> 요소를 보여줍니다.

```
<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>
```

제 12 장 WS-Addressing에 대한 지원

CICS는 W3C(Worldwide Web Consortium) 웹 서비스 주소 지정(WS-Addressing) 스펙을 사용하는 서비스를 지원합니다. 이 스펙의 계열은 웹 서비스의 주소를 지정하고 엔드 투 엔드 주소 지정을 용이하게 하는 전송 독립 메커니즘을 제공합니다.

CICS는 기존 웹 서비스 애플리케이션이 WS-Addressing을 사용하는 웹 서비스의 요청을 승인할 수 있도록 보장합니다. 또한 엔드포인트 참조 및 SOAP 메시지의 메시지 주소 지정 특성을 사용하는 새 웹 서비스를 작성할 수 있습니다.

WS-Addressing은 MAP(Message Addressing Properties) 양식의 주소 지정 정보를 SOAP 메시지 헤더에 추가합니다. MAP은 메시징 정보(예:고유 메시지 ID 및 메시지 생성 위치, 메시지를 보낼 대상 및 응답 또는 결함 메시지를 보낼 대상에 대해 자세히 설명하는 엔드포인트 참조)를 포함합니다. 엔드포인트 참조(EPR)는 특정 유형의 MAP이며 메시지의 대상 주소, 애플리케이션에서 사용할 선택적 참조 매개변수 및 선택적 메타데이터를 포함합니다.

WS-Addressing 지원의 기능

CICS에는 다음과 같은 WS-Addressing 지원 기능이 있습니다.

- 웹 서비스 요청자 및 제공자 애플리케이션이 사용자에게 재배치를 요구하지 않고도 WS-Addressing을 사용 중인 다른 서비스와 상호 작용할 수 있습니다. 파이프라인의 새 메시지 핸들러인 주소 지정 메시지 핸들러 DFHWSADH는 WS-Addressing 정보를 포함하는 메시지를 지정된 웹 서비스로 라우팅합니다.
- WS-Addressing API 명령을 사용하는 애플리케이션을 작성하여 엔드포인트 참조를 작성하고 주소 지정 컨텍스트를 작성, 갱신, 삭제 및 조회할 수 있습니다.
- 응답 메시지를 요청자 엔드포인트 외의 엔드포인트로 라우팅할 수 있습니다. 예를 들어, 결함 메시지를 전용 결함 핸들러로 라우팅할 수 있습니다.
- 애플리케이션에 참조 매개변수를 SOAP 헤더의 MAP 일부로 전달할 수 있습니다.

WS-Addressing 스펙 및 상호 운용성에 대한 지원

기본적으로 CICS는 권장 스펙을 지원합니다.

- W3C WS-Addressing 1.0 - 코어
- W3C WS-Addressing 1.0 - SOAP 바인딩
- W3C WS-Addressing 1.0 - 메타데이터

이러한 스펙은 <http://www.w3.org/2005/08/addressing> 네임스페이스로 식별합니다. 별도로 명시되지 않는 한 이 문서에 설명된 WS-Addressing 시맨틱은 권장 스펙을 참조합니다.

상호 운용성의 경우 CICS는 제출 스펙도 지원합니다.

- W3C WS-Addressing - 제출

이 스펙은 <http://schemas.xmlsoap.org/ws/2004/08/addressing> 네임스페이스로 식별합니다. 제출 스펙은 제출 스펙을 구현하는 웹 서비스 제공자 또는 클라이언트와 상호 운용해야 하는 경우에만 사용하십시오.

WS-Addressing 개요

웹 서비스 주소 지정(WS-Addressing)은 SOAP 메시지의 엔드포인트를 지정하기 위한 표준 프레임워크를 제공합니다. 이 프레임워크는 전송 중립이고 다른 전송 메커니즘을 사용하는 웹 서비스의 상호 운용성을 개선합니다. WS-Addressing 스펙은 메시지 주소 지정 특성 및 엔드포인트 참조를 소개합니다.

웹 서비스 주소 지정(WS-Addressing)은 웹 서비스의 주소를 지정하고 SOAP 메시지에 주소 지정 정보를 제공하는 표준 방식을 정의하여 웹 서비스 간 상호 운용성을 향상시키는 W3C(Worldwide Web Consortium) 스펙입니다. SOAP 메시지는 각각 서로 다른 방식으로 메시지의 대상 정보를 저장하는 HTTP 및 WebSphere MQ를 포함하여 다양한 전송 메커니즘을 통해 보낼 수 있습니다.

WS-Addressing을 사용하도록 구성되는 파이프라인에 배치되는 기존 CICS 웹 서비스는 변경 없이 기본 WS-Addressing 설정을 사용할 수 있습니다. WS-Addressing 기능을 최대한 활용하려면 WS-Addressing API 명령을 사용하십시오. WS-Addressing 구현은 각 WSDL 조작에 대해 한 개의 SOAP 결함을 지원합니다.

메시지 주소 지정 특성

메시지 주소 지정 특성(MAP)은 일련의 잘 정의된 WS-Addressing 특성이며 SOAP 헤더의 요소로 나타낼 수 있습니다. MAP는 메시지 응답이 경로 지정되는 엔드포인트 또는 메시지와 다른 메시지 간 관계 정보와 같은 표준 정보 전달 방식을 제공합니다. WS-Addressing 스펙에 정의된 MAP는 다음 테이블에 요약됩니다.

표 17. WS-Addressing 스펙에 의해 정의되는 메시지 주소 지정 특성

추상 WS-Addressing MAP 이름	SOAP WS-Addressing MAP 이름	MAP 콘텐츠 유형	다중성	설명
[action]	<wsa:Action>	xs:anyURI	1..1	메시지의 시맨틱을 고유하게 식별하는 절대 URI입니다. 이 값은 필수입니다.
[destination]	<wsa:To>	SOAP 메시지의 xs:anyURI 주소 지정 컨텍스트에서 EndpointReference	0..1	메시지의 계획된 수신인 주소를 지정하는 절대 URI입니다. 이 값을 지정하지 않으면 스펙에서 정의되는 익명 URI(http://www.w3.org/2005/08/addressing/anonymous)로 기본값 설정됩니다. 주소 지정 컨텍스트에서 <wsa:To> MAP는 EPR로 표시됩니다. <wsa:To>가 SOAP 메시지의 일부로 보내지는 경우 스키마에서 정의된 대로 해당 주소 및 해당 참조 매개변수로 분할됩니다.

표 17. WS-Addressing 스펙에 의해 정의되는 메시지 주소 지정 특성 (계속)

추상 WS-Addressing MAP 이름	SOAP WS-Addressing MAP 이름	MAP 콘텐츠 유형	다중성	설명
[reference parameters] *	[참조 매개변수]*	xs:any	0..무제한	메시지가 주소 지정되는 엔드포인트 참조의 <wsa:ReferenceParameters> 특성에 해당하는 매개변수입니다. 이 값은 선택사항입니다.
[source endpoint]	<wsa:From>	EndpointReference	0..1	메시지가 시작되는 엔드포인트에 대한 참조입니다. 이 값은 선택사항입니다.
[reply endpoint]	<wsa:ReplyTo>	EndpointReference	0..1	이 메시지에 대한 응답의 계획된 수신인에 대한 엔드포인트 참조입니다. 이 값은 선택사항입니다. 이 값을 지정하지 않으면 http://www.w3.org/2005/08/addressing/anonymous로 기본값 설정됩니다.
[fault endpoint]	<wsa:FaultTo>	EndpointReference	0..1	이 메시지에 관련된 결함의 계획된 수신인에 대한 엔드포인트 참조입니다. 이 값은 선택사항이고 <wsa:ReplyTo> MAP의 값으로 기본값 설정됩니다.
[relationship] *	<wsa:RelatesTo>	xs:anyURI에 xs:anyURI 유형의 선택적 속성 추가	0..무제한	이 메시지가 다른 메시지에 관련된 방식을 나타내는 값 쌍입니다. 이 요소의 콘텐츠는 관련 메시지의 <wsa:MessageID>를 전달합니다. 선택적 속성은 관계 유형을 전달합니다. 이 값은 선택사항입니다. 이 값을 지정하지 않으면 http://www.w3.org/2005/08/addressing/reply로 기본값 설정됩니다.
[message id]	<wsa:MessageID>	xs:anyURI		메시지를 고유하게 식별하는 절대 URI입니다. 이 값은 선택사항입니다. 값을 제공하지 않으면 CICS가 아웃바운드 요청 및 응답에 대한 값을 생성합니다.

다음 예제 SOAP 메시지는 WS-Addressing MAP를 포함합니다.

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://w3.org/2005/08/addressing"
  xmlns:example="http://example.ibm.com/namespace">
  <S:Header>
    ...
    <wsa:To>http://example.ibm.com/enquiry</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://example.ibm.com/enquiryReply</wsa:Address>
    </wsa:ReplyTo>
    <wsa:Action>...</wsa:Action>
    <example:AccountCode wsa:IsReferenceParameter='true'>123456789</example:AccountCode>
    <example:DiscountId wsa:IsReferenceParameter='true'>ABCDEFG</example:DiscountId>
    ...
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

엔드포인트 참조

엔드포인트 참조는 특정 엔드포인트에 대한 정보를 캡슐화하기 위한 표준 메커니즘을 제공하는 특정 유형의 MAP입니다. 엔드포인트 참조는 다른 당사자에게 보낼 수 있는

며 엔드 포인트 참조가 나타내는 웹 서비스 엔드포인트를 대상화하는 데 사용할 수 있습니다. 다음 테이블은 엔드포인트 참조에 대한 정보 모델을 요약합니다.

표 18. 엔드포인트 참조에 대한 정보 모델

추상 특성 이름	특성 유형	다중성	설명
[address]	xs:anyURI	1..1	엔드포인트의 주소를 지정하는 절대 URI입니다.
[reference parameters] *	xs:any	0..무제한	엔드포인트와 인터페이스하는 데 필요한 네임스페이스로 규정된 요소 정보 항목입니다.
[metadata]	xs:any	0..무제한	엔드포인트의 동작, 정책 및 기능에 대한 설명입니다.

다음 XML 단편은 엔드포인트 참조를 나타냅니다. <wsa:EndpointReference> 요소는 URI <http://example.ibm.com/enquiry>에서 엔드포인트를 참조하고 엔드포인트 참조가 참조하는 인터페이스를 지정하는 메타데이터 및 일부 애플리케이션별 참조 매개변수를 포함합니다.

```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
  xmlns:example="http://example.ibm.com/namespace">
  <wsa:Address>http://example.ibm.com/enquiry</wsa:Address>
  <wsa:Metadata
    xmlns:wsdli="http://www.w3.org/ns/wsdli-instance"
    wsdl:wsdlLocation="http://example.ibm.com/wsdli/wsdli-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>
```

wsa:EndpointReferenceType 유형의 WS-Addressing MAP는 <wsa:From>, <wsa:ReplyTo> 및 <wsa:FaultTo>입니다. 그러나 <wsa:To> MAP는 유형이 xs:anyURI 인 것으로 WS-Addressing 1.0 표준에서 정의됩니다. 단순성을 위해 CICS는 주소 지정 컨텍스트에서 <wsa:To> MAP를 EPR로 다룹니다. <wsa:To> MAP가 SOAP 메시지의 일부로 전송되면 CICS는 표준에서 요구하는 대로 이를 해당 주소 및 참조 매개변수로 분할합니다.

기본 네임스페이스

다음 접두부 및 해당 네임스페이스는 WS-Addressing 문서 전체에서 참조됩니다.

표 19. 접두부 및 해당 네임스페이스

기본 접두부	네임스페이스
xs	http://www.w3.org/2001/XMLSchema

표 19. 접두부 및 해당 네임스페이스 (계속)

기본 접두부	네임스페이스
wsa	http://www.w3.org/2005/08/addressing(권장 스키마) http://schemas.xmlsoap.org/ws/2004/08/addressing(제출 스키마)
wsam	http://www.w3.org/2007/05/addressing/metadata

관련 참조:

361 페이지의 『DFHWS-URI 컨테이너』

DFHWS-URI는 서비스 URI를 포함하는 DATATYPE(CHAR) 컨테이너입니다.

WS-Addressing을 위한 요청자 파이프라인 구성

웹 서비스 주소 지정(WS-Addressing)을 지원하도록 요청자 파이프라인을 구성하려면 파이프라인 구성 파일에 주소 지정 핸들러를 추가해야 합니다.

시작하기 전에

WS-Addressing에 대한 구성 정보를 추가하려면 파이프라인 구성 파일을 식별하거나 작성해야 합니다. 사용할 WS-Addressing 스펙도 결정해야 합니다. 가능한 경우 W3C WS-Addressing 1.0 코어 스펙을 사용하십시오.

이 태스크 정보

다음 두 가지 방식 중 하나로 WS-Addressing에 대한 지원을 추가할 수 있습니다.

- SOAP 파이프라인이 Java를 사용하는 경우 SOAP 처리는 Axis2에 의해 처리되고 이 기술에서 제공하는 지원을 사용하여 WS-Addressing을 사용하는 요청을 처리할 수 있습니다. 모든 헤더 핸들링은 Axis2에 의해 처리되며 파이프라인에 DFHWSADH 헤더 처리 프로그램을 추가하지 않는 것이 중요합니다. 자체 헤더 처리 프로그램을 사용할 수 있습니다. SOAP 헤더를 처리하려는 경우 더 나은 성과를 얻으려면 Java로 Axis2 핸들러를 작성하십시오.
- SOAP 파이프라인이 Java를 사용하지 않는 경우 CICS 제공 헤더 처리 프로그램인 DFHWSADH를 추가하여 요청을 처리해야 합니다.

프로시저

- SOAP 파이프라인이 <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소를 사용하는 경우 파이프라인 구성 파일에 <addressing> 요소를 추가하십시오. 요청 메시지에 사용할 스펙을 포함하는 한 개의 <namespace> 요소를 포함시키십시오. 이 스펙은 응답 메시지에 대한 스펙과 다를 수 있습니다. 예를 들어, 응답 메시지가 제출 스펙을 사용한다고 해도 W3C 코어 스펙을 준수하는 요청을 항상 보낼 수 있습니다. Axis2는 인바운드 메시지에 대해 두 WS-Addressing 스펙 모두 지원합니다.

다음 예제는 요청자 파이프라인을 구성하는 방법을 보여줍니다.

```
<requester_pipeline>
  <service>
    <service_handler_list>
      <cics_soap_1.1_handler_java>
        <jvmserver>JVMSERV1</jvmserver>
        <addressing>
          <namespace>http://www.w3.org/2005/08/addressing</namespace>
        </addressing>
      </cics_soap_1.1_handler_java>
    </service_handler_list>
  </service>
</requester_pipeline>
```

<jvmserver> 요소는 Axis2를 지원하는 JVMSERVER 자원의 이름을 포함합니다.

- SOAP 파이프라인이 Java를 사용하지 않는 경우 <cics_soap_1.1_handler> 또는 <cics_soap_1.2_handler>에서 CICS 주소 지정 헤더 프로그램을 파이프라인 구성 파일에 추가하십시오. 다음 예제는 요청자 파이프라인을 구성하는 방법을 보여줍니다.

```
<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>
```

<program_name>, <localname> 및 <mandatory> 요소를 표시된 그대로 코드화하십시오. <namespace>를 <http://www.w3.org/2005/08/addressing>으로 설정하여 W3C WS-Addressing 1.0 코어 스펙을 사용하거나 <http://schemas.xmlsoap.org/ws/2004/08/addressing>으로 설정하여 W3C WS-Addressing 제출 스펙을 사용하십시오.

헤더 처리 프로그램의 순서는 보장되지 않습니다. 다른 헤더 처리 프로그램을 정의하는 경우 정의한 프로그램을 <service_handler_list> 요소에 있는 후속 CICS SOAP 핸들러 요소에서 추가하십시오. DFHWSADH 헤더 핸들러는 첫 번째 SOAP 핸들러 요소에 있어야 합니다.

결과

요청자 파이프라인이 WS-Addressing을 지원하도록 구성되었습니다.

다음에 수행할 작업

구성 파일을 가리키는 PIPELINE 자원을 작성하십시오. Java 기반 SOAP 파이프라인을 사용하는 경우 JVMSERVER 자원이 Axis2 처리를 처리할 수 있는지 확인하십시오.

WS-Addressing을 위한 제공자 파이프라인 구성

웹 서비스 주소 지정(WS-Addressing)을 지원하도록 제공자 파이프라인을 구성하려면 파이프라인 구성 파일에 주소 지정 핸들러를 추가해야 합니다.

시작하기 전에

WS-Addressing에 대한 구성 정보를 추가하려면 파이프라인 구성 파일을 식별하거나 작성해야 합니다. 사용할 WS-Addressing 스펙도 결정해야 합니다. 가능한 경우 W3C *WS-Addressing 1.0* 코어 스펙을 사용하십시오.

이 태스크 정보

다음 두 가지 방식 중 하나로 WS-Addressing에 대한 지원을 추가할 수 있습니다.

- SOAP 파이프라인이 Java를 사용하는 경우 SOAP 처리는 Axis2에 의해 처리되고 이 기술에서 제공하는 지원을 사용하여 WS-Addressing을 사용하는 요청을 처리할 수 있습니다. 모든 헤더 핸들링은 Axis2에 의해 처리되며 파이프라인에 DFHWSADH 헤더 처리 프로그램을 추가하지 않는 것이 중요합니다. 자체 헤더 처리 프로그램을 사용할 수 있습니다. SOAP 헤더를 처리하려는 경우 더 나은 성과를 얻으려면 Java로 Axis2 핸들러를 작성하십시오.
- SOAP 파이프라인이 Java를 사용하지 않는 경우 CICS 제공 헤더 처리 프로그램인 DFHWSADH를 추가하여 요청을 처리해야 합니다.

프로시저

- SOAP 파이프라인이 <cics_soap_1.1_handler_java> 또는 <cics_soap_1.2_handler_java> 요소를 사용하는 경우 파이프라인 구성 파일에 <addressing> 요소를 추가하십시오. 선택적으로 한 개 이상의 <namespace> 요소를 포함시킬 수 있습니다. 이 요소에는 아웃바운드 메시지에 사용할 스펙이 포함되며 이 스펙은 인바운드 메시지에 대한 스펙과 다를 수 있습니다. 예를 들어, 인바운드 메시지가 제출 스펙을 사용한다고 해도 W3C 코어 스펙을 준수하는 아웃바운드 응답을 항상 보낼 수 있습니다. 이 요소를 제외시키면 Axis2는 인바운드 메시지와 같은 스펙을 아웃바운드 메시지에 사용합니다. Axis2는 인바운드 메시지에 대해 두 WS-Addressing 스펙 모두 지원합니다.

다음 예제는 제공자 파이프라인을 구성하는 방법을 보여줍니다.

```

<provider_pipeline>
  <terminal_handler>
    <cics_soap_1.1_handler_java>
      <jvmserver>JVMSESV1</jvmserver>
      <addressing>
        <namespace>http://www.w3.org/2005/08/addressing</namespace>
      </addressing>
    </cics_soap_1.1_handler_java>
  </terminal_handler>
</provider_pipeline>

```

<jvmserver> 요소는 Axis2를 지원하는 JVMSESV1 자원의 이름을 포함합니다.

- SOAP 파이프라인이 Java를 사용하지 않는 경우 CICS 주소 지정 헤더 프로그램인 DFHWSADH를 파이프라인 구성 파일의 SOAP 핸들러에 추가하십시오. 다음 예제는 제공자 파이프라인을 구성하는 방법을 보여줍니다.

```

<provider_pipeline>
  <terminal_handler>
    <cics_soap_1.1_handler>
      <headerprogram>
        <program_name>DFHWSADH</program_name>
        <namespace>http://www.w3.org/2005/08/addressing</namespace>
        <localname>*</localname>
        <mandatory>true</mandatory>
      </headerprogram>
    </cics_soap_1.1_handler>
  </terminal_handler>
</provider_pipeline>

```

<program_name>, <localname> 및 <mandatory> 요소를 표시된 그대로 코드화하십시오. <namespace>를 http://www.w3.org/2005/08/addressing으로 설정하여 W3C WS-Addressing 1.0 코어 스펙을 사용하거나 http://schemas.xmlsoap.org/ws/2004/08/addressing으로 설정하여 W3C WS-Addressing 제출 스펙을 사용하십시오.

헤더 처리 프로그램의 순서는 보장되지 않습니다. 다른 헤더 처리 프로그램을 정의하는 경우 정의한 프로그램을 <service_handler_list> 요소에 있는 다른 CICS SOAP 핸들러 요소에서 추가하십시오. DFHWSADH 헤더 핸들러는 마지막 SOAP 핸들러 요소에 있어야 합니다.

결과

제공자 파이프라인이 WS-Addressing을 지원하도록 구성되었습니다.

다음에 수행할 작업

구성 파일을 가리키는 PIPELINE 자원을 작성하십시오. Java 기반 SOAP 파이프라인을 사용하는 경우 JVMSESV1 자원이 Axis2 처리를 처리할 수 있는지 확인하십시오.

WS-Addressing을 사용하는 웹 서비스 작성

WSDL에서 웹 서비스 주소 지정(WS-Addressing)을 사용하는 웹 서비스를 작성하려면 웹 서비스 지원에서 매개변수를 사용하여 XML에서 언어 구조로의 변환을 처리하십시오.

이 태스크 정보

웹 서비스 지원 작업 DFHWS2LS를 사용하여 엔드포인트 참조(EPR)가 WSDL 문서에서 처리되는 방식을 제어하고 CICS가 기본 입력, 출력 및 결합 조치를 구성하는지 여부를 판별할 수 있습니다.

프로시저

1. 웹 서비스 지원 DFHWS2LS에서 **MINIMUM-RUNTIME** 매개변수를 3.0 이상으로 설정하십시오. 최소 3.0 이상의 런타임 레벨은 생성된 웹 서비스 바인딩이 WS-Addressing을 지원하고 다른 웹 서비스 플랫폼과 상호 운용할 수 있도록 합니다.
2. 웹 서비스 지원 DFHWS2LS에서 **MAPPING-LEVEL** 매개변수를 3.0 이상으로 설정하십시오.
3. 요청 또는 응답 메시지에서 `wsa:EndpointReferenceType` 유형 요소를 사용하면 **WSADDR-EPR-ANY** 매개변수를 TRUE로 설정하십시오. 엔드포인트 참조는 애플리케이션 데이터에 포함될 수 있고 **WSACONTEXT BUILD**와 같은 API 명령에서 EPR을 사용하는 옵션이 있습니다. **WSADDR-EPR-ANY** 매개변수를 TRUE로 설정하는 것은 CICS가 런타임 시 EPR을 언어 구조로 변환하지 말아야 하며 대신 CICS가 EPR 데이터를 `<xsd:any>` 요소로 처리한 후 이름 지정된 컨테이너에 저장해야 함을 나타냅니다.

이 예제 WSDL 단편은 `wsa:EndpointReferenceType` 유형의 요소로 전달되고 있는 `<wsa:To>` MAP를 보여줍니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="exampleEPR" targetNamespace="http://example.ibm.com/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:s0="http://example.ibm.com/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata">
  <types>
    <xs:schema targetNamespace="http://test.org/"
      xmlns:s="http://www.w3.org/2001/XMLSchema"
      xmlns:s0="http://example.ibm.com/"
      xmlns:wsa="http://www.w3.org/2005/08/addressing">
      ...
      <xs:element name="exampleResponse" type="s0:typeResponse"/>
      <xs:complexType name="typeResponse">
        <xs:sequence>
          <xs:element name="myEpr" type="wsa:EndpointReferenceType"/> 1
        </xs:sequence>
      </xs:complexType>
      ...
    </xs:schema>
  </types>
</definitions>
```

```

</types>
...
<message name="msgResponse">
  <part element="s0:exampleResponse" name="response"/>
</message>
...
</definitions>

```

<xs:element name="myEpr" type="wsa:EndpointReferenceType"/> **1** 요소가 **WSADDR-EPR-ANY** 매개변수가 TRUE로 설정된 DFHWS2LS에 의해 처리될 때 myEpr 요소 데이터는 <xsd:any> 요소이자 생성된 언어 구조에 추가된 컨테이너에 대한 포인터로 이름 지정된 컨테이너에 저장됩니다.

예를 들어, myEpr 요소에 대해 DFHWS2LS에서 생성되는 COBOL 언어 구조는 다음과 같습니다.

```

09 myEpr.
   12 myEpr-xml-cont          PIC X(16).
   12 myEpr-xmlns-cont       PIC X(16).

```

myEpr-xml-cont 컨테이너는 myEpr 데이터를 포함하는 컨테이너의 이름을 저장합니다. myEpr-xmlns-cont는 범위 내 XML 네임스페이스 선언으로 채워지는 선택적 컨테이너입니다.

4. DFHWS2LS 작업을 저장하고 제출하십시오.

결과

CICS는 웹 서비스 바인딩을 작성하여 서비스 요청자 또는 제공자 애플리케이션을 작성하는 데 사용할 수 있는 언어 구조 및 데이터 변환을 처리합니다.

다음에 수행할 작업

웹 서비스를 사용으로 설정하려면 파이프라인 스캔을 수행하여 필수 CICS 자원을 작성하십시오.

기본 엔드포인트 참조

대부분의 WSDL 문서에는 웹 서비스가 호스팅되는 주소가 포함되어 있습니다. WS-Addressing에서 WSDL 문서는 웹 서비스의 엔드포인트 참조(EPR)도 포함할 수 있습니다. 이 EPR은 요청자와 제공자 애플리케이션 사이의 통신을 용이하게 하는 추가 메타데이터도 포함할 수 있습니다.

DFHWS2LS를 사용하여 WSDL을 처리하는 경우 EPR은 웹 서비스 바인딩에 저장되어 요청 및 응답 메시지를 보내기 위해 CICS에서 사용됩니다. EPR에서 정의되는 모든 참조 매개변수 <wsa:ReferenceParameters>는 SOAP 메시지에 포함됩니다. 이 EPR은 애플리케이션에 의해 대체될 수 있기 때문에 기본 EPR이라고 합니다. 애플리케이션이 명시적 EPR을 제공하지 않는 경우 WSDL의 기본 EPR이 사용됩니다.

다음 WSDL 1.1 단편에는 기본 EPR(<soap:address location="http://example.ibm.com:12345/exampleTest" />.)이 포함됩니다. <port> 요소는 하위 요소인 <wsa:EndpointReference>를 포함하여 하위 요소 **2**에 의해 지정되는 주소는 상위 요소 **1**에서 지정된 주소와 일치해야 합니다.

```
<service name="exampleService">
  <port name="examplePort" binding="s0:createBinding">
    <soap:address location="http://example.ibm.com:12345/exampleTest" /> 1

    <wsa:EndpointReference
      xmlns:example="http://example.ibm.com/namespace"
      xmlns:wsdl="http://www.w3.org/2006/01/wsdl-instance"
      wsdl:wsdlLocation="http://example.ibm.com/location "
      title="http://example.ibm.com/example/example_wsdl"
      class="http://example.ibm.com/example/example_wsdl">
        <wsa:Address>http://example.ibm.com:12345/exampleTest</wsa:Address> 2
        <wsa:Metadata>
          <wsam:InterfaceName>example:Inventory</wsam:InterfaceName>
        </wsa:Metadata>
        <wsa:ReferenceParameters>
          <example:AccountCode>123456789</example:AccountCode>
          <example:DiscountId>ABCDEFGH</example:DiscountId>
        </wsa:ReferenceParameters>
      </wsa:EndpointReference>
    </port>
  </service>
```

명시적 조치

WSDL 문서는 <wsa:Action> 특성의 값을 명시적으로 정의할 수 있습니다. WSDL 문서에 명시적으로 정의된 <wsa:Action> 특성이 없을 경우 CICS는 WSDL이 DFHWS2LS에 의해 처리될 때 기본 조치를 빌드합니다.

WSDL 1.1

다음 WSDL 1.1 단편은 명시적으로 정의된 <wsa:Action> 특성을 포함하는 예약 시스템을 나타냅니다.

```
<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>
```

이 예제에서 makeBooking 조작의 입력 조치는 http://example.ibm.com/namespace/makeBooking으로 명시적으로 정의되고 출력 조치는 http://example.ibm.com/namespace/bookingResponse로 명시적으로 정의됩니다.

WSDL 2.0

다음 WSDL 2.0 단편은 명시적으로 정의된 <wsa:Action> 특성을 포함하는 예약 시스템을 나타냅니다.

```
<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>
```

이 예제에서 makeBooking 조작의 입력 조치는 http://example.ibm.com/namespace/makeBooking으로 명시적으로 정의되고 출력 조치는 http://example.ibm.com/namespace/makeBookingResponse로 정의됩니다.

자세한 정보는 W3C WS-Addressing 1.0 메타데이터 스펙을 참조하십시오.

WSDL 1.1의 기본 조치

WSDL 1.1 문서에 명시적으로 지정된 <wsa:Action> 특성이 포함되지 않은 경우 CICS는 WSDL이 DFHWS2LS에 의해 처리될 때 기본 입력, 출력 및 기본 조치를 빌드합니다.

WSDL 1.1의 기본 입력 및 출력 조치

다음 패턴은 권장 스키마 또는 제출 스키마에 따라 기본 입력 또는 출력 조치를 구성하는 WSDL 1.1 문서에서 CICS에 의해 사용됩니다.

```
[target namespace]/[port type name]/[input|output name]
```

WSDL 1.1의 기본 결함 조치

권장 스키마를 따르는 경우 CICS가 기본 결함 조치를 빌드하는 방식은 스키마에서 설명한 동작과는 다릅니다. 다음 패턴은 권장 스키마에 따라 기본 결함 메시지를 구성하는 WSDL 1.1 문서에서 CICS에 의해 사용됩니다. 결함 이름은 생략됨에 유의하십시오.

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

제출 스키마를 따르는 경우 CICS가 기본 결함 조치를 빌드하는 방식은 스키마에서 설명한 동작을 따릅니다. 다음 패턴은 제출 스키마에 따라 기본 결함 메시지를 구성하는 WSDL 1.1 문서에서 CICS에 의해 사용됩니다.

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

CICS에 의해 생성되는 WSDL 1.1 문서의 기본 조치 예제

이 예약 시스템 예제는 CICS가 WSDL 1.1 문서에서 기본 조치를 구성하는 방법을 보여줍니다.

```
<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>
```

WSDL 단편은 다음과 같은 주소 지정 특성을 가집니다.

특성 이름	값
[대상 네임스페이스]	http://example.ibm.com/namespace
[portType name]	bookingInterface
[조작 이름]	makeBooking
[입력 이름]	MakeBooking
[출력 이름]	BookingResponse
[결함 이름]	InvalidBooking

이러한 값에서 다음 조치가 작성됩니다.

조치	값
입력 조치	http://example.ibm.com/namespace/bookingInterface/MakeBooking [입력 이름]을 지정하지 않으면 "요청"이 추가된 [조작 이름]의 값이 대신 사용됩니다. 예를 들어, 이 경우에 입력 조치는 http://example.ibm.com/namespace/bookingInterface/makeBookingRequest입니다. http://example.ibm.com/namespace/bookingInterface/BookingResponse
출력 조치	[출력 이름]을 지정하지 않으면 "응답"이 추가된 [조작 이름]의 값이 대신 사용됩니다. 예를 들어, 이 경우에 출력 조치는 http://example.ibm.com/namespace/bookingInterface/makeBookingResponse입니다.
결함 조치 (권장 스키마)	http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/ [결함 이름]은 생략됨에 유의하십시오.
결함 조치 (제출 스키마)	http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/ InvalidBooking

자세한 정보는 W3C WS-Addressing 1.0 메타데이터 스펙을 참조하십시오.

WSDL 2.0의 기본 조치

WSDL 2.0 문서에 명시적으로 지정된 <wsa:Action> 특성이 포함되지 않은 경우 CICS는 WSDL이 DFHWS2LS에 의해 처리될 때 기본 입력, 출력 및 기본 조치를 빌드합니다.

WSDL 2.0의 기본 입력 및 출력 조치

다음 패턴은 권장 스키마에 따라 입력 및 출력에 대한 기본 조치를 구성하는 WSDL 2.0 문서에서 CICS에 의해 사용됩니다.

```
[target namespace]/[interface name]/[operation name][direction token]
```

WSDL 2.0의 기본 결함 조치

권장 스키마를 따르는 경우 CICS가 WS-Addressing 결함에 대한 기본 조치를 빌드하는 방식은 스키마에서 설명한 동작과는 다릅니다. 제출 스키마를 따르는 경우 CICS가 WS-Addressing 결함에 대한 기본 조치를 빌드하는 방식은 스키마에서 설명한 동작을 따릅니다.

다음 패턴은 권장 스키마에 따라 결함에 대한 기본 조치를 구성하는 WSDL 2.0 문서에서 CICS에 의해 사용됩니다. 결함 이름은 생략됨에 유의하십시오.

```
[target namespace]/[interface name]/
```

다음 패턴은 제출 스키마에 따라 결함에 대한 기본 조치를 구성하는 WSDL 2.0 문서에서 CICS에 의해 사용됩니다.

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

CICS에 의해 생성되는 WSDL 2.0 문서의 기본 조치 예제

이 예제는 CICS가 권장 스키마에 따라 WSDL 2.0 문서의 기본 조치를 구성하는 방법을 보여줍니다.

```
<description targetNamespace="http://example.ibm.com/namespace" ...>
...
<interface name="bookingInterface">
  <operation name="makeBooking" pattern="http://www.w3.org/ns/wsd1/in-out">
    <input element="tns:makeBooking" messageLabel="In"/>
    <output element="tns:bookingResponse" messageLabel="Out"/>
  </operation>
</interface>
...
</definitions>
```

WSDL 단편은 다음과 같은 주소 지정 특성을 가집니다.

특성 이름	값
[대상 네임스페이스]	http://example.ibm.com/namespace
[인터페이스 이름]	bookingInterface
[조작 이름]	makeBooking
[방향 토큰]	Request 또는 Response.

이러한 값에서 다음 입력 및 출력조치가 작성됩니다.

조치	값
입력 조치	http://example.ibm.com/namespace/bookingInterface/makeBookingRequest
출력 조치	http://example.ibm.com/namespace/bookingInterface/makeBookingResponse

자세한 정보는 W3C WS-Addressing 1.0 메타데이터 스펙을 참조하십시오.

메시지 교환

웹 서비스 주소 지정(WS-Addressing)은 다음의 메시지 교환을 지원합니다. 단방향, 양방향요청- 응답, 동기 요청-응답 및 비동기 요청-응답.

WS-Addressing 메시지 교환에는 메시지 주소 지정 특성(MAP) 및 엔드포인트 참조(EPR)가 포함됩니다.

런타임 시 CICS는 요청 메시지의 SOAP 헤더에 관련 WS-Addressing 메시지 정보가 포함되는지 확인하고 요청자 애플리케이션은 WS-Addressing 헤더를 설정하지 않아도 되며 WS-Addressing을 사용 중인지 인식하지 못할 수도 있습니다.

단방향

이 단순한 단방향 메시지는 입력 전용 조작으로 정의됩니다. 이 조작에 대한 WSDL(Web Services Description Language)의 양식은 다음과 같습니다.

```
<operation name="myOperation">
  <input message="tns:myInputMessage"/>
</operation>
```

WS-Addressing을 사용 중인 경우 CICS는 런타임 시 <wsa:Action> MAP 및 <wsa:MessageID> MAP를 WS-Addressing 요청 메시지의 SOAP 메시지 헤더에 추가하여 WS-Addressing 스펙 준수를 보장합니다.

<wsa:MessageID> MAP는 고유 ID이며 지정하지 않은 경우 CICS가 이 ID를 자동으로 생성합니다.

<wsa:Action> MAP는 WSDL에서 파생되고 WSBind 파일로 저장됩니다.

CICS WS-Addressing API 명령을 사용하여 이러한 MAP의 값을 대체할 수 있습니다.

양방향 요청-응답

이 양방향 교환에는 요청 메시지 및 응답 메시지가 포함됩니다. 조작의 응답 파트는 출력 메시지, 결함 메시지 또는 둘 다로 정의할 수 있습니다. 요청-응답 조작에 대한 WSDL 정의의 양식은 다음과 같습니다.

```
<operation name="myOperation">
  <input message="tns:myInputMessage"/>
  <output message="tns:myOutputMessage"/>
  <fault="tns:myFaultMessage"/>
</operation>
```

응답 대상 또는 결함 생성 위치, 엔드포인트로 경로 지정된 요청은 응답 유형이 보통인 지 또는 결함인지에 따라 <wsa:ReplyTo> MAP 또는 the <wsa:FaultTo> MAP를 대상으로 합니다.

요청 메시지에서 <wsa:ReplyTo> 또는 <wsa:FaultTo> MAP를 지정하여 응답을 전송해야 하는 위치를 표시하십시오.

권장 스펙을 사용하고 <wsa:ReplyTo> MAP의 값을 지정하지 않으면 <wsa:ReplyTo> MAP는 익명 URI(<http://www.w3.org/2005/08/addressing/anonymous>)를 포함하는 엔드포인트 참조로 기본값 설정되고 이로 인해 CICS는 요청자에게 응답을 다시 보냅니다.

권장 스펙을 사용하고 <wsa:FaultTo> MAP의 값을 지정하지 않으면 <wsa:FaultTo> MAP가 <wsa:ReplyTo> MAP의 값으로 기본값 설정됩니다.

요청자가 올바르지 않은 MAP를 빌드하고 이로 인해 유효성 검증이 실패하면 CICS는 <wsa:FaultTo> MAP에 의해 지정된 주소 대신 요청자에게 다시 결함 메시지를 보냅니다.

동기 요청-응답

기본적으로 양방향 메시지의 응답 파트는 사용 중인 기본 프로토콜에 따라 리턴됩니다. HTTP 요청의 경우 응답은 HTTP 응답에서 동시에 리턴됩니다.

비동기 요청-응답

비동기 응답은 다른 웹 서비스를 대상으로 하고 원래 요청자 애플리케이션에 다시 돌아오지 않습니다. HTTP 요청의 경우 요청 클라이언트와의 연결은 HTTP 202 응답과 함께 닫힙니다. 웹 서비스 제공자가 CICS 시스템을 실행 중인 경우 요청자 애플리케이션은 비어 있는 응답 메시지를 수신합니다. 웹 서비스 제공자가 WebSphere MQ 시스템에서 실행되고 있는 경우 요청자 애플리케이션은 어떠한 응답도 수신하지 않습니다.

양방향 메시지 응답 파트의 대상을 변경하려면 <wsa:ReplyTo> MAP 또는 <wsa:ReplyTo> 및 <wsa:FaultTo>, MAP에 적절한 주소를 지정해야 합니다.

WSDL 1.1과 WSDL 2.0에서 필수인 MAP의 전체 목록은 428 페이지의 『WS-Addressing의 필수 메시지 주소 지정 특성』의 내용을 참조하십시오.

관련 개념:

36 페이지의 『WSDL 및 메시지 교환 패턴』

WSDL 2.0 문서에는 웹 서비스 요청자와 웹 서비스 제공자 간에 SOAP 1.2 메시지가 교환되는 방식을 정의하는 메시지 교환 패턴(MEP)이 포함됩니다.

관련 참조:

428 페이지의 『WS-Addressing의 필수 메시지 주소 지정 특성』

WS-Addressing 1.0 메타데이터 스펙은 WSDL 1.1 및 WSDL 2.0 문서에 포함시켜야 하는 메시지 주소 지정 특성(MAP)을 명시합니다. WS-Addressing의 CICS 구현은 이러한 필수 MAP에 대한 값을 자동으로 제공하여 WS-Addressing 스펙을 준수할 수 있도록 돕습니다.

WS-Addressing의 필수 메시지 주소 지정 특성

WS-Addressing 1.0 메타데이터 스펙은 WSDL 1.1 및 WSDL 2.0 문서에 포함시켜야 하는 메시지 주소 지정 특성(MAP)을 명시합니다. WS-Addressing의 CICS 구현은 이러한 필수 MAP에 대한 값을 자동으로 제공하여 WS-Addressing 스펙을 준수할 수 있도록 돕습니다.

제공하는 WSDL에서 MAP에 사용자 고유 값을 지정하고 CICS WS-Addressing API 명령을 사용하여 이러한 값을 주소 지정 컨텍스트에서 갱신할 수 있습니다. 필수 MAP에 대한 값을 제공하지 않으면 CICS가 값을 생성하게 됩니다.

다음 테이블은 WSDL 1.1 및 WSDL 2.0에서 지원되는 여러 메시지 교환 패턴(MEP)에 대해 필수인 MAP를 나열합니다.

표 20. WS-Addressing의 필수 메시지 주소 지정 특성

WS-Addressing MAP 이름	설명	WSDL 1.1에서 필수	WSDL 2.0에서 필수
<wsa:To>	메시지의 계획된 수신인 주소입니다.	아니오	아니오
<wsa:Action>	WS-Addressing 조치: 입력, 출력 또는 결합	다음 MEP에 대해 필수임: 단방향 양방향(요청) 양방향(응답)	다음 MEP에 대해 필수임: In-only Robust In-only(In) Robust In-only(결합) In-out(In) In-out(Out) In-optional-out(In) In-optional-out(Out)

표 20. WS-Addressing의 필수 메시지 주소 지정 특성 (계속)

WS-Addressing MAP 이름	설명	WSDL 1.1에서 필수	WSDL 2.0에서 필수
<wsa:From>	메시지가 시작되는 엔드포인트입니다.	아니오	아니오
이 값은 필수가 아님			
<wsa:ReplyTo>	메시지에 대한 응답의 계획된 수신인의 엔드포인트입니다.	아니오	아니오
<wsa:FaultTo>	메시지에 관련된 결함에 대한 계획된 수신인의 엔드포인트입니다.	아니오	아니오
<wsa:MessageID>	고유 메시지 ID입니다.	다음 MEP에 대해 필수임: 양방향(요청)	다음 MEP에 대해 필수임: Robust In-only(In) In-out(In) In-optional-out(In)
<wsa:RelatesTo>	이 메시지가 다른 메시지에 관련된 방식을 나타내는 값 쌍입니다. 이 요소에는 관련된 메시지의 <wsa:MessageID>가 포함되고 선택적 속성은 관계 유형을 전달합니다.	다음 MEP에 대해 필수임: 양방향(응답)	다음 MEP에 대해 필수임: Robust In-only(결함) In-out(Out) In-optional-out(Out)

자세한 정보는 W3C WS-Addressing 1.0 메타데이터 스펙(<http://www.w3.org/TR/ws-addr-metadata>)을 참조하십시오.

참고:

- <wsa:ReplyTo> MAP의 주소 요소에 대해 값이 설정되지 않은 경우 주소는 익명 URI <http://www.w3.org/2005/08/addressing/anonymous>로 설정됩니다. 익명 URI는 응답이 요청자에게로 다시 전송되는 것을 나타냅니다.
- <wsa:FaultTo> MAP의 주소 요소에 대해 값을 지정하지 않으면 CICS는 이 주소를 <wsa:ReplyTo> MAP의 주소 요소와 같은 값으로 설정합니다.

요청자가 올바르지 않은 MAP를 빌드하고 이로 인해 유효성 검증이 실패하면 CICS는 <wsa:FaultTo> MAP에 의해 지정된 주소 대신 요청자에게 다시 결함 메시지를 보내는 점에 유의하십시오.

- <wsa:To> MAP의 값을 지정하지 않으면 CICS가 주소를 익명 URI <http://www.w3.org/2005/08/addressing/anonymous>로 설정합니다. 익명 URI는 요청이 DFHWS-URI 컨테이너에서 지정된 주소로 전송됨을 나타내며 자세한 정보는 361 페이지의 『DFHWS-URI 컨테이너』의 내용을 참조하십시오.
- <wsa:Action> MAP를 WSDL 문서에서 명시적으로 정의하거나 CICS가 자동으로 생성하게 할 수 있습니다.
- CICS는 런타임 시 응답을 예상하는 요청 메시지 및 응답 메시지에 대해 <wsa:MessageID> MAP에 자동으로 고유 값을 설정합니다.

- <wsa:RelatesTo> MAP는 응답 메시지에 대해 필수입니다. 메시지의 관계 유형은 선택적이고 <http://www.w3.org/2005/08/addressing/reply>로 기본값 설정됩니다.

관련 개념:

426 페이지의 『메시지 교환』

웹 서비스 주소 지정(WS-Addressing)은 다음의 메시지 교환을 지원합니다. 단방향, 양방향요청-응답, 동기 요청-응답 및 비동기 요청-응답.

관련 참조:

361 페이지의 『DFHWS-URI 컨테이너』

DFHWS-URI는 서비스 URI를 포함하는 DATATYPE(CHAR) 컨테이너입니다.

WS-Addressing 보안

웹 서비스 주소 지정(WS-Addressing)을 사용하여 공용 네트워크에서 이동하는 통신은 적절하게 보호되어야 하고 통신 당사자 간 충분한 수준의 신뢰가 확립되어야 합니다. 통신 보안을 위해 SSL 또는 HTTPS와 같은 전송 레벨 보안을 사용하도록 권장됩니다.

SSL 또는 HTTPS와 같은 전송 레벨 보안은 WS-Addressing 통신 보안을 보장하는 가장 단순한 방법입니다. 전송 레벨 보안을 사용할 수 없는 경우에는 WS-Addressing 메시지 주소 지정 특성에 서명하거나 엔드포인트 참조를 암호화하여 메시지를 보호할 수 있습니다.

CICS는 WS-Addressing 메시지 주소 지정 특성을 포함하는 헤더에 서명할 수 없으며 엔드포인트 참조를 암호화할 수 없습니다. 그러나 CICS는 수신 메시지에 대한 서명을 확인하고 암호화된 헤더를 복호화할 수 있습니다. 서명 및 암호화를 사용하여 통신을 보호하려는 경우 IBM WebSphere DataPower XML Security Gateway와 같은 외부 보안 게이트웨이를 사용해야 합니다. 자세한 정보는 IBM WebSphere DataPower XML Security Gateway를 참조하십시오.

WS-Addressing 예제

이 예제는 WS-Addressing을 사용하여 메시지를 보내는 회사에 고객이 주문을 할 때 발생하는 프로세스에 대한 상위 레벨 개요를 제공합니다.

전자 컴포넌트를 판매하는 국제 기업은 비즈니스에 WS-Addressing을 사용합니다. 이러한 회사의 인프라는 주문 클라이언트, 분배 서비스 그룹, 이행 서비스 및 구성 서비스로 구성됩니다.

WS-Addressing 사용은 회사에 다음과 같은 이점을 제공합니다.

- WS-Addressing은 메시지 전송을 위한 전송 독립 메커니즘을 제공하고 이 메커니즘은 여러 플랫폼에서 실행되는 웹 서비스 간 상호 운용성을 향상시킵니다. 이 예제에서 회사가 소유하는 분배 서비스는 다양한 플랫폼에서 실행됩니다. WS-Addressing

은 간단하게 여러 플랫폼 간의 상호 운용성을 제공합니다. 웹 서비스 요청자 및 제공자는 메시지를 교환하고 있는 서비스가 실행되고 있는 플랫폼에 대해서 알지 못해도 됩니다.

- WS-Addressing은 <wsa:ReplyTo> MAP에서 EPR을 갱신하여 응답 메시지의 대상을 변경하는 데 사용할 수 있습니다. 이 예제에서 이행 서비스는 메시지가 우회되는 분배 서비스를 선택할 때 응답 메시지의 대상을 수정합니다.

회사는 다수의 국가에 여러 분배 센터를 가지고 있으며 이 예제에서 각 분배 센터는 분배 서비스로 표시되고 구성 서비스에 등록됩니다.

이행 서비스는 요청된 항목의 가용성 및 고객과 분배 센터와의 거리를 포함한 다양한 요소를 기준으로 하여 주문을 처리하기에 가장 적합한 분배 서비스를 선택합니다.

주소 지정 정보는 구성 서비스에(서) 전달됩니다. 구성 서비스는 사용 가능한 서비스의 주소를 엔드포인트 참조 양식으로 저장합니다. **WSAEPR CREATE** 명령을 사용하여 EPR을 작성하고 이 EPR을 구성 서비스에 전송하여 새 서비스를 구성 서비스에 등록합니다. 구성 서비스에는 XML의 블록으로 EPR이 필요하기 때문에 DFHWS2LS의 **WSADDR-EPR-ANY** 매개변수를 TRUE로 설정해야 합니다. **WSADDR-EPR-ANY=TRUE** 옵션을 사용하여 CICS에 EPR을 <xsd:any> 요소로 처리하도록 지시합니다. CICS는 런타임 시 이 EPR을 언어 구조로 변환하는 대신 컨테이너에 배치해야 합니다.

이러한 서비스가 상호 작용하는 방식은 다음 다이어그램에 표시되어 있습니다. 이 다이어그램은 비즈니스 애플리케이션에 관련되었을 수 있지만 태스크에서는 제외된 다른 서비스를 보여줍니다.

- 주문의 상태로 기타 서비스 각각에 의해 갱신할 수 있는 추적 서비스
- 발생하는 결함 메시지를 처리하기 위한 문제점 해결 서비스
- 주문 클라이언트에게 향하는 응답 메시지를 처리하는 주문 클라이언트 콜백 서비스

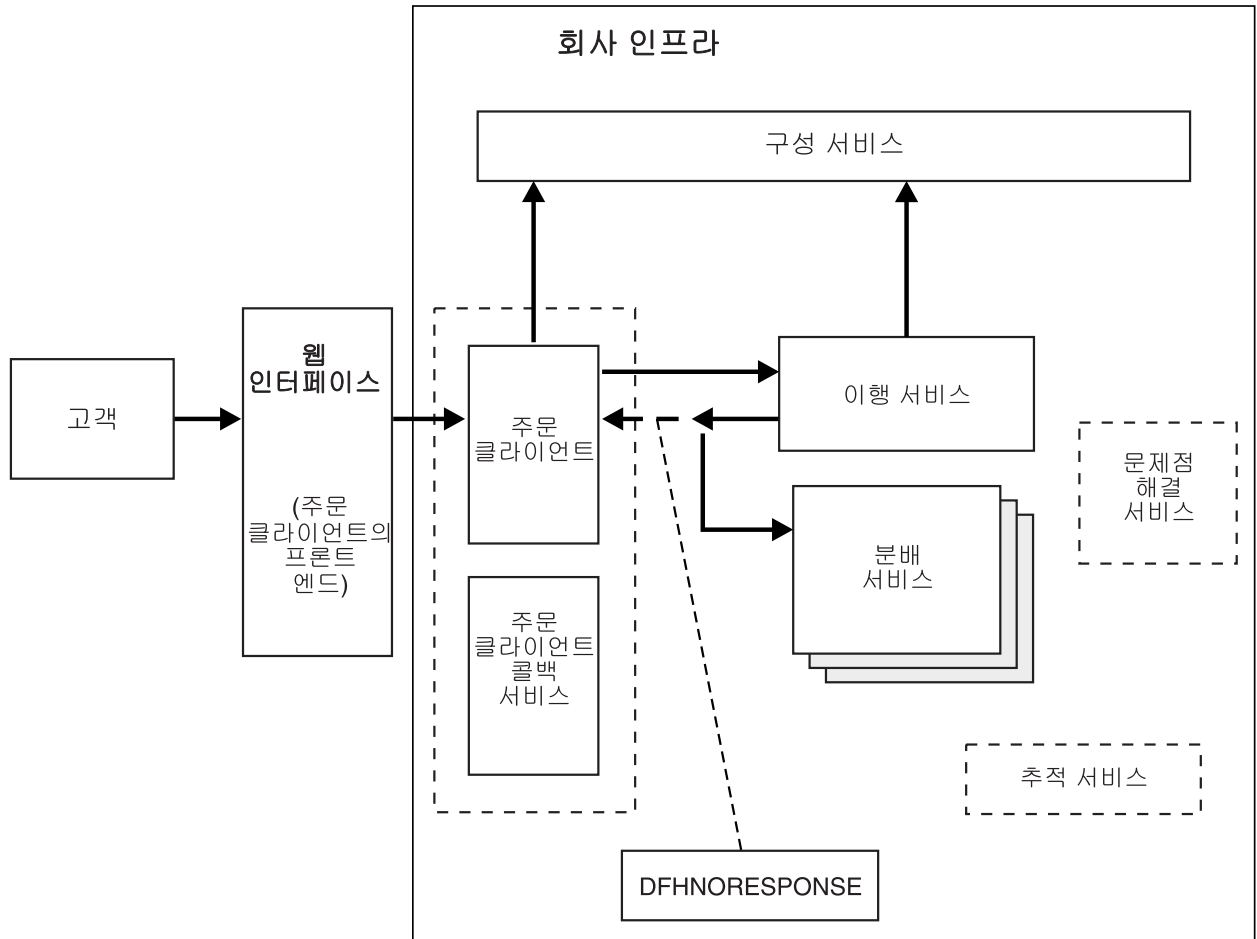


그림 36. 회사 인프라

다음 단계는 고객이 주문을 하는 순간부터 주문이 처리되는 시점까지 발생하는 프로세스를 설명합니다.

1. 고객이 회사에 주문합니다.
 - a. 고객이 주문 클라이언트의 프론트 엔드인 회사 웹 사이트에서 주문합니다.
 - b. 주문 클라이언트가 고객의 접속 세부사항을 주문의 일부로 가져옵니다.
 - c. 주문 클라이언트는 웹 인터페이스를 통해 고객에게 확인 및 고유 주문 참조를 리턴합니다.
2. 주문 클라이언트는 이행 서비스에 주문 요청을 보냅니다.
 - a. 이행 서비스의 EPR을 이미 알고 있지 않으면 주문 클라이언트는 구성 서비스에 이를 요청합니다. 주문 클라이언트가 구성 서비스에 이행 서비스의 EPR을 요청할 때 포함되는 프로세스는 <wsa:To>의 예제 섹션에 자세히 설명되어 있습니다.

- b. 주문 클라이언트는 이행 서비스에 대해 **INVOKE SERVICE** 명령을 발행합니다. WS-Addressing은 메시지를 TO EPR에서 요청 주소 지정 컨텍스트로 지정된 주소로 라우팅합니다.
3. 이행 서비스는 주문을 처리할 분배 서비스를 선택하고 응답 메시지를 해당 서비스로 경로 재지정합니다.
 - a. 이행 서비스는 **WSACONTEXT GET** 명령을 사용하여 주소 지정 컨텍스트로부터 주문 참조 및 기타 주소 지정 특성을 추출합니다.
 - b. 이행 서비스는 구성 서비스에서 가장 적합한 분배 서비스를 선택합니다.
 - c. <wsa:ReplyTo> EPR이 주소 지정 컨텍스트에 추가됩니다.


```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Address>http://www.example.ibm.com/DistributionService</wsa:Address>
</wsa:EndpointReference>
```

이행 서비스는 **WSACONTEXT BUILD** 명령을 사용하여 선택한 분배 서비스의 ReplyTo EPR을 요청 주소 지정 컨텍스트에 추가합니다.
 - d. 이행 서비스는 **WSACONTEXT BUILD** 명령을 반복적으로 사용하여 주문 참조 및 기타 정보를 요청 주소 지정 컨텍스트에 추가합니다.
 - e. DFHNORESPONSE 컨테이너는 주문 클라이언트 파이프라인에 추가되어 응답을 수신하지 않으며 응답 메시지가 요청 메시지의 양식으로 분배 서비스에 경로 재지정됨을 주문 클라이언트에 표시합니다.
4. 분배 서비스는 경로 재지정된 응답 메시지를 수신하여 주문을 처리합니다.
 - a. 분배 서비스는 **WSACONTEXT GET** 명령을 사용하여 요청 주소 지정 컨텍스트로부터 주문 참조 및 주소 지정 세부사항을 추출합니다.
 - b. 분배 서비스가 주문을 처리합니다.

<wsa:To>의 예제

1. 주문 클라이언트가 메시지를 보낼 서비스의 EPR을 구성 서비스에 요청합니다. 이 예제에서 주문 클라이언트는 이행 서비스의 EPR을 요청합니다.
2. 구성 서비스는 다음과 같이 응답 메시지를 작성하고 보냅니다.
 - a. 구성 서비스는 이행 서비스의 요청된 <wsa:To> EPR을 **WSAEPR CREATE** API 명령 EXEC CICS WSAEPR CREATE를 사용하여 작성합니다.
 - b. 구성 서비스는 **WSAEPR CREATE** 명령의 출력을 컨테이너 EXEC CICS PUT CONTAINER(work-cont)에 기록합니다.
 - c. 구성 서비스는 컨테이너 이름을 myEpr-xml-cont 요소인 MOVE work-cont TO myEpr-xml-cont에 복사합니다.

- d. 구성 서비스는 주문 클라이언트로 응답 메시지를 보내고 이 메시지에는 myEpr-xml-cont 컨테이너로 이름 지정된 컨테이너의 콘텐츠가 포함됩니다. 이 예제에서 work-cont 컨테이너의 콘텐츠는 <wsa:myEpr> 요소 내 주문 클라이언트에게로 전송됩니다.

```
<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>
```

434 페이지의 그림 32에서는 주문 클라이언트와 구성 서비스 간의 요청-응답 메시지 교환을 보여줍니다. 이 메시지 교환에는 두 개의 일반적인 웹 서비스 파이프라인이 포함됩니다.

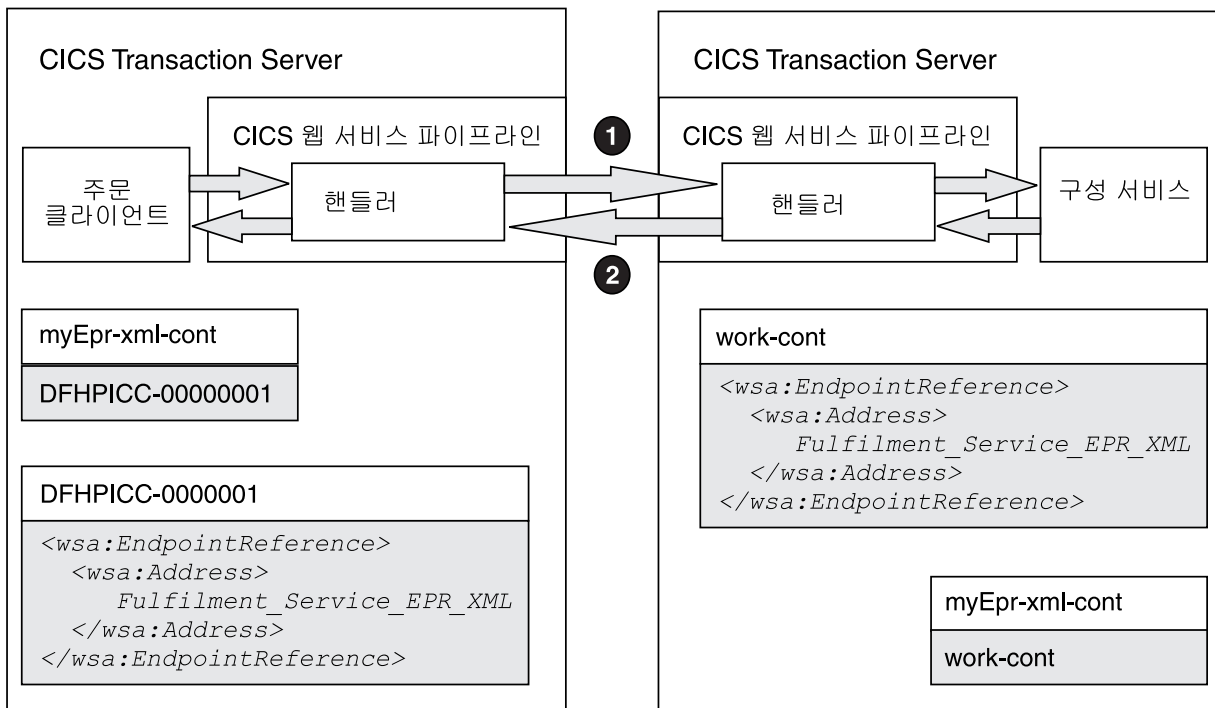


그림 37. 주문 클라이언트와 구성 서비스 간 요청-응답 메시지 교환

- 주문 클라이언트는 응답 메시지를 수신하고 <wsa:To> EPR을 빌드한 다음 이행 서비스에 요청을 보냅니다.

- a. 주문 클라이언트는 응답 메시지에서 <wsa:To> EPR 데이터를 추출합니다.
- b. CICS는 고유 컨테이너(이 예제에서는 DFHPICC-00000001 컨테이너임)를 <wsa:To> EPR 데이터로 채웁니다.
- c. CICS는 컨테이너의 이름(이 예제에서는 DFHPICC-00000001임)을 myEpr-xml-cont 요소에 복사합니다.
- d. 주문 클라이언트는 myEpr-xml-cont 요소로 지정된 컨테이너의 콘텐츠를 읽고 이 콘텐츠를 **WSACONTEXT BUILD** API 명령에 대한 입력으로 제공합니다. **WSACONTEXT BUILD** 명령은 이 입력을 사용하여 이행 서비스의 <wsa:To> EPR을 빌드합니다.
- e. 주문 클라이언트는 파이프라인 처리를 시작하는 **INVOKE SERVICE** 명령을 발행합니다.
- f. CICS WS-Addressing 핸들러인 DFHWSADH는 아웃바운드 파이프라인에서 <wsa:To> EPR을 주소 및 선택적 참조 매개변수 세트로 변환하고 이행 서비스로 보내고 있는 SOAP 요청 메시지의 헤더에 넣습니다.

```
<env:Header>
  <wsa:To>http://example.ibm.com/Fulfilment_Service</wsa:To>
</env:Header>
```

435 페이지의 그림 33에서는 주문 클라이언트에서 이행 서비스로 보내는 요청을 보여줍니다. 이 요청에는 CICS WS-Addressing 핸들러인 DFHWSADH를 포함하는 웹 서비스 파이프라인이 포함됩니다.

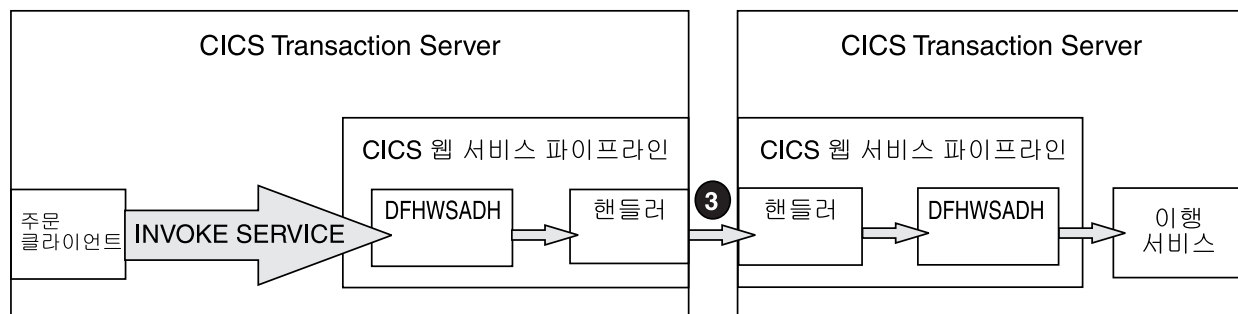


그림 38. 주문 클라이언트에서 이행 서비스로의 요청

관련 참조:

435 페이지의 『WS-Addressing 용어』

웹 서비스 주소 지정(WS-Addressing) 지원을 설명하는 데 사용되는 용어입니다.

WS-Addressing 용어

웹 서비스 주소 지정(WS-Addressing) 지원을 설명하는 데 사용되는 용어입니다.

주소 지정 컨텍스트

SOAP 요청 메시지로 보내기 전 및 SOAP 요청 및 응답 메시지에서 수신된 후 WS-Addressing 메시지 주소 지정 특성(MAP)을 저장하는 XML 문서입니다.

엔드포인트 참조(EPR)

메시지를 웹 서비스로 라우팅하는 데 사용되는 주소 지정 정보를 포함하는 XML 구조입니다. 이 주소 지정 정보에는 메시지의 대상 주소, 애플리케이션에서 사용할 선택적 참조 매개변수 및 선택적 메타데이터가 포함됩니다.

메시지 주소 지정 특성(MAP)

고유 메시지 ID, 메시지 대상 및 메시지의 엔드포인트 참조와 같이 특정 웹 서비스 메시지에 대한 주소 지정 정보를 전달하는 XML 요소입니다.

제 13 장 웹 서비스 보안 지원

z/OS용 CICS Transaction Server는 SOAP 및 JSON 메시지 보안에 사용할 수 있는 다수의 관련 기술에 대한 지원을 제공합니다.

이러한 기술 중 일부는 HTTP 프로토콜의 일부로 사용 가능하고 SOAP 및 JSON 둘 다에 동일하게 적용 가능합니다. 일부는 웹 서비스 보안(WSS): SOAP 메시지 보안 1.0 스펙을 사용하고 SOAP에 대해서만 사용 가능합니다. 공유 TCP/IP 및 HTTP 보안 옵션에 대한 정보는 TCP/IP 클라이언트에 대한 보안 및 보안에서 CICS 웹 지원을 위한 보안의 내용을 참조하십시오.

SAML 어설션 사용에 대한 정보는 SAML 지원의 개요의 내용을 참조하십시오.

SOAP 웹 서비스 보안

웹 서비스 보안(WSS): SOAP 메시지 보안 1.0은 SOAP 메시지 보호 및 인증을 위해 보안 토큰 및 디지털 서명의 사용에 대해 설명합니다. 자세한 정보는 WSS: Soap 메시지 보안 1.0 스펙을 참조하십시오.

웹 서비스 보안은 각각 권한 없는 공개로부터 메시지를 보호하고 권한이 없고 발견되지 않는 수정을 방지하여 SOAP 메시지의 개인정보 보호정책 및 무결성을 보호합니다. WSS는 메시지의 XML 요소를 디지털로 서명하고 암호화하여 이러한 보호를 제공합니다. 보호할 수 있는 요소는 본문 또는 본문이나 헤더의 모든 요소입니다. SOAP 메시지의 여러 다른 요소에 여러 레벨의 보호를 제공할 수 있습니다.

웹 서비스 신뢰 언어 스펙은 보안 토큰 요청 및 발행을 위한 프레임워크를 제공하고 웹 서비스 요청자와 제공자 사이의 신뢰 관계를 관리하여 웹 서비스 보안을 더욱 강화합니다. SOAP 메시지 인증에 대한 이러한 확장을 통해 웹 서비스는 신뢰 씨드파티를 사용하여 여러 유형의 보안 토큰을 유효성 검증하고 교환할 수 있습니다. 이 씨드파티를 STS(Security Token Service)라고 합니다. 웹 서비스 신뢰 언어에 대한 자세한 정보는 WS-Trust 언어 스펙을 참조하십시오.

z/OS용 CICS Transaction Server는 파이프라인에서 CICS 제공 보안 핸들러를 사용하여 이러한 스펙에 대한 지원을 제공합니다.

- 아웃바운드 메시지의 경우 CICS는 전체 SOAP 본문의 디지털 서명 및 암호화에 대한 지원을 제공합니다. CICS는 다른 유형의 보안 토큰에 대한 사용자 이름 토큰을 STS와 교환할 수도 있습니다.
- 인바운드 메시지의 경우 CICS는 본문 및 헤더의 본문 또는 요소가 암호화되거나 디지털로 서명되는 메시지를 지원합니다. CICS는 보안 토큰을 유효성 검증하고 STS와 교환할 수도 있습니다.

CICS는 CICS 보안 핸들러를 사용하지 않고도 STS와 상호작용할 수 있도록 개별 신뢰 클라이언트 인터페이스도 제공합니다.

참고: 웹 서비스 보안은 SP800-131A를 준수하지 않을 수 있습니다. 웹 서비스 보안은 파이프라인에 핸들러를 추가하여 구성되고 CICS는 고객 작성 핸들러에서의 처리를 제어할 수 없습니다. 디지털 서명을 사용하는 경우 `dsa-sha1` 및 `rsa-sha1` 알고리즘만 지정할 수 있습니다. 이러한 알고리즘은 SP800-131A를 준수하지 않습니다. SOAP 본문을 암호화하는 데 사용할 수 있는 두 개의 키 3-DES 암호화 알고리즘도 비준수입니다.

웹 서비스 보안을 위한 필수 소프트웨어

웹 서비스 보안을 구현하려면 다음 갱신을 CICS 리전에 적용해야 합니다. z/OS용 IBM XML 툴킷 v1.10을 설치하고 APAR OA14956을 적용한 후 DFHRPL 연결에 3개의 라이브러리를 추가합니다.

이 태스크 정보

웹 서비스 보안을 구현하기 전에 다음 단계를 완료하십시오.

프로시저

1. 무료 z/OS용 IBM XML 툴킷 v1.10을 설치하십시오. <http://www.ibm.com/servers/eserver/zseries/software/xml/> 사이트에서 다운로드할 수 있습니다. 버전 1.10을 설치해야 합니다. 이후 버전은 CICS의 웹 서비스 보안 지원에서 작동하지 않습니다.
2. z/OS에 이미 설치되지 않은 경우 ICSF APAR OA14956을 적용하십시오.
3. DFHRPL 연결에 다음 라이브러리를 추가하십시오.
 - `hlq.SIXMLOD1`, 여기서 `hlq`는 XML 툴킷의 상위 레벨 규정자입니다.
 - `hlq.SCEERUN`, 여기서 `hlq`는 Language Environment의 상위 레벨 규정자입니다.
 - `hlq.SDFHWSLD`, 여기서 `hlq`는 CICS 설치의 상위 레벨 규정자입니다(예: `CICSTS53`).

처음 두 개 라이브러리는 런타임 시 보안 핸들러가 필요로 하는 DLL을 포함합니다. `IXM4C57`은 XML 툴킷에 의해 제공되고 `hlq.SIXMLOD1`에 있습니다. `C128N`은 Language Environment 런타임에 의해 제공되고 `hlq.SCEERUN`에 있습니다.

`hlq.SDFHWSLD` 라이브러리를 사용하면 CICS가 `DFHWSSE1` 및 `DFHWSXXX` 웹 서비스 보안 모듈을 찾을 수 있습니다.

4. **EDSALIM** 시스템 초기화 매개변수의 값을 늘려야 할 수도 있습니다. 로드되는 세 개의 DLL에는 대략 15MB의 EDSA 저장영역이 필요합니다.

결과

라이브러리가 지정되어 있지 않은 경우 다음 메시지를 참조하십시오.

```
CEE3501S The module module_name was not found.
```

*module_name*은 누락된 라이브러리에 따라 달라집니다.

SOAP 웹 서비스 보안 계획

웹 서비스를 보안화하기 위한 최선의 방법을 결정할 수 있습니다. CICS는 구성 가능한 보안 메시지 핸들러 및 개별 신뢰 클라이언트 인터페이스를 포함한 다수의 옵션을 지원합니다.

이 태스크 정보

CICS는 각 웹 서비스에 대해서가 아닌 파이프라인 레벨에서 웹 서비스 보안 (WS-Security 또는 WSS)을 구현합니다. 다음 질문에 응답하여 보안을 구현하는 최선의 방법을 결정하십시오.

프로시저

1. 파이프라인 처리의 성능이 중요합니까? 웹 서비스 보안을 위해 WSS를 사용하면 상당한 성능 영향이 발생합니다.

WSS 구현의 주요 장점은 SOAP 메시지의 일부를 암호화하여 중간 노드의 체인을 통해 메시지를 전송할 수 있다는 점입니다. 모든 중간 노드에는 라우팅 또는 처리 결정을 내리기 위해 SOAP 헤더를 확인할 정당한 이유가 있지만 메시지의 콘텐츠를 보는 것은 허용되지 않습니다. 기밀이어야 하는 섹션만 암호화하여 다음과 같은 이점을 얻을 수 있습니다.

- 중간 프로세스의 체인에 있는 모든 모드에서 암호화 및 복호화의 오버헤드를 발생시키지 않습니다.
- 비신뢰 노드의 공용 네트워크를 통해 데이터의 최종 수신인만 이해할 수 있는 기밀 메시지를 라우팅할 수 있습니다.

WSS 사용에 대한 대안으로 SSL을 사용하여 전체 데이터 스트림을 암호화할 수 있습니다.

2. WSS를 사용하려는 경우 원하는 보안 레벨은 무엇입니까? 옵션에는 메시지 헤더에 사용자 이름 및 비밀번호가 포함되는 기본 인증부터 메시지에 디지털 서명과 암호화를 결합하는 것까지 있습니다. CICS 보안 핸들러가 지원하는 옵션은 678 페이지의 『SOAP 메시지 보안 옵션』에 설명되어 있습니다.
3. CICS 제공 보안 핸들러가 요구사항을 충족시킵니까? 고급 보안 처리를 수행하려는 경우 자체 사용자 정의 보안 핸들러를 작성해야 합니다. 이 핸들러는 직접 RACF를 사용하거나 보안 토큰 서비스를 사용하여 필요한 메시지 인증을 수행하고 디지

털 인증서 및 암호화된 요소를 처리해야 합니다. 세부사항은 695 페이지의 『사용자 정의 보안 핸들러 작성』의 내용을 참조하십시오.

4. 파이프라인이 MTOM 핸들러를 포함합니까? 파이프라인 구성 파일에서 MTOM 핸들러 및 보안 핸들러 둘 다 사용으로 설정할 계획인 경우 모든 MIME 다중 파트 또는 관련 메시지는 호환성 모드로 처리됩니다. 보안 핸들러는 메시지의 본문에 있는 XOP 요소를 구문 분석할 수 없기 때문입니다. 이 처리는 파이프라인 처리의 성능에 추가로 영향을 미칠 수 있습니다.

SOAP 메시지 보안 옵션

CICS는 SOAP 메시지의 서명 및 암호화를 둘 다 지원하므로 SOAP 메시지로 보내거나 받는 데이터에 가장 적합한 보안 레벨을 선택할 수 있습니다.

SOAP 메시지의 서명 및 암호화는 제공자 모드 Axis2 웹 서비스 Java 애플리케이션 또는 Axis2 MessageContext를 사용하여 파이프라인에 연결하는 제공자 웹 서비스에서 지원되지 않습니다.

다음 옵션 중 선택할 수 있습니다.

신뢰 인증

서비스 제공자 파이프라인에서 CICS는 SOAP 메시지 헤더의 사용자 이름 토큰을 신뢰 토큰으로 승인할 수 있습니다. 이 유형의 보안 토큰은 일반적으로 사용자 이름 및 비밀번호를 포함하지만 이 경우 비밀번호는 필요하지 않습니다. CICS는 제공된 사용자 이름을 신뢰하고 이 사용자 이름을 컨테이너 DFHWS-USERID에 배치합니다. 그러면 메시지가 파이프라인에서 처리됩니다.

서비스 요청자 파이프라인에서 CICS는 서비스 제공자에게 SOAP 메시지 헤더의 비밀번호 없는 사용자 이름 토큰을 전송할 수 있습니다.

기본 인증

서비스 제공자 모드에서 CICS는 인바운드 SOAP 메시지의 인증을 위해 SOAP 메시지 헤더의 사용자 이름 토큰을 승인할 수 있습니다. 이 유형의 보안 토큰에는 사용자 이름 및 비밀번호가 포함됩니다. CICS는 RACF와 같은 외부 보안 관리자를 사용하여 사용자 이름 토큰을 확인합니다. 성공적이면 사용자 이름이 컨테이너 DFHWS-USERID에 배치되고 SOAP 메시지는 파이프라인에서 처리됩니다. CICS가 사용자 이름 토큰을 확인할 수 없는 경우 서비스 요청자에게 SOAP 결함 메시지가 리턴됩니다.

비밀번호를 포함하는 사용자 이름 토큰은 서비스 요청자 모드에서 또는 아웃바운드 SOAP 메시지에 대해 지원되지 않습니다.

HTTP 기본 인증

서비스 제공자 모드에서 CICS는 HTTP 프로토콜을 통해 기본 인증 정보를 승인할 수 있습니다. 서비스 요청자는 URIMAP 정의를 사용하여 신임 정보(사

용자 식별 정보)가 글로벌 사용자 종료 XWBAUTH에 의해 캡처될 수 있도록 지정합니다. XWBAUTH는 요청 시 이 정보를 CICS에 전달하고 CICS는 HTTP 인증 헤더로 이 정보를 서비스 제공자에게 전송합니다.

고급 인증

서비스 제공자 및 요청자 파이프라인에서 인증을 목적으로 보안 토큰을 확인하거나 STS(Security Token Service)와 교환할 수 있습니다. 이 인증을 사용하면 CICS가 메시지 헤더에 일반적으로 지원되지 않는 보안 토큰(예: Kerberos 토큰 또는 SAML 어설션)이 있는 메시지를 승인하고 보낼 수 있습니다.

인바운드 메시지의 경우 보안 토큰 확인 또는 교환을 선택할 수 있습니다. 요청을 보안 토큰과 교환할 경우 CICS는 STS로부터 사용자 이름 토큰을 다시 받아야 합니다. 아웃바운드 메시지의 경우 보안 토큰에 대해서만 사용자 이름 토큰을 교환할 수 있습니다.

X.509 인증서에 서명

서비스 제공자 및 서비스 요청자 모드에서 인증을 위해 SOAP 메시지 헤더에 X.509 인증서를 제공하여 SOAP 메시지 본문에 서명할 수 있습니다. 이 유형의 보안 토큰을 2진 보안 토큰이라고 합니다. 인바운드 SOAP 메시지의 2진 보안 토큰을 승인하려면 인증서와 연관된 공개 키를 RACF와 같은 외부 보안 관리자로 가져와서 **KEYRING** 시스템 초기화 매개변수에 지정되는 키 링과 연관시켜야 합니다. 아웃바운드 SOAP 메시지의 경우 공개 키를 생성하고 계획된 수신인에게 공개합니다. ICSF(Integrated Cryptographic Service Facility)는 공개 키를 생성하는 데 사용됩니다.

X.509 디지털 인증서와 연관된 레이블을 지정하는 경우 다음 문자는 사용하지 마십시오.

< > : ! =

또한 두 번째 X.509 인증서를 헤더에 포함시키고 첫 번째 인증서를 사용하여 서명할 수 있습니다. 이 두 번째 인증서를 사용하여 두 번째 X.509 인증서와 연관된 사용자 ID로 CICS에서 작업을 실행할 수 있습니다. SOAP 메시지에 서명하는 데 사용하는 인증서는 신뢰 사용자 ID와 연관되어야 하고 해당 ID와 연관된 비밀번호가 있는 신뢰 사용자 ID 없이도 다른 ID(어설션된 ID)로 실행되도록 어설션하기 위한 대리 권한이 있어야 합니다.

암호화 서비스 제공자 및 서비스 요청자 모드에서 3-DES 또는 AES와 같은 대칭 알고리즘을 사용하여 SOAP 메시지 본문을 암호화할 수 있습니다. 대칭 알고리즘은 데이터를 암호화하고 복호화하는 데 같은 키가 사용되는 위치입니다. 이 키를 대칭 키라고 합니다. 그러면 이 키는 메시지에 포함되어 계획된 수신인의 공개 키와 비대칭 키 암호화 알고리즘 RSA 1.5의 결합을 사용하여 암호화됩니다. 이 암호화는 향상된 보안을 제공합니다. 비대칭 알고리즘은 복잡하고 대

칭 키 복호화는 어렵기 때문입니다. 그러나 더 나은 성능을 얻을 수 있습니다. SOAP 메시지의 대부분이 대칭 알고리즘으로 암호화되고 이러한 암호화는 복호화보다 빠르기 때문입니다.

인바운드 SOAP 메시지의 경우 SOAP 본문의 요소를 암호화한 다음 SOAP 본문 전체를 암호화할 수 있습니다. 이러한 종류의 암호화는 특히 민감한 데이터를 포함하는 요소에 적합합니다. CICS가 두 레벨로 암호화된 SOAP 메시지를 받으면 CICS는 두 레벨 모두 자동으로 복호화합니다. 이러한 유형의 암호화는 아웃바운드 SOAP 메시지에 대해 지원되지 않습니다.

CICS는 메시지 헤더에만 암호화된 요소가 있고 SOAP 본문에는 암호화된 요소가 없는 인바운드 SOAP 메시지를 지원하지 않습니다.

서명 및 암호화

서비스 제공자 및 서비스 요청자 모드에서 SOAP 메시지 서명 및 암호화를 둘 다 선택할 수 있습니다. CICS는 항상 SOAP 메시지 본문에 먼저 서명한 다음 암호화합니다. 이 메소드의 장점은 메시지 기밀성 및 무결성을 둘 다 제공한다는 점입니다.

ICRX 기반 ID 전과

서비스 제공자 모드에서 미인증 WS-Security 사용자 ID 토큰을 사용하는 경우와 같은 환경에서 미인증 ICRX(Extended Identity Context Reference) ID 토큰을 사용할 수 있습니다. ICRX ID 토큰은 사용자 ID로 맵핑하는 z/OS ID입니다. CICS는 ICRX ID 토큰을 사용자 ID로 분석하고 사본을 DFHWS-ICRX 컨테이너에 배치합니다. CICS는 DFHWS-USERID 컨테이너도 채웁니다. ICRX ID 토큰에 대한 자세한 정보는 보안에서 ID 프로포게이션 및 분배 보안의 내용을 참조하십시오.

보안 토큰 서비스를 사용하여 인증

CICS는 Tivoli Federated Identity Manager와 같은 STS(Security Token Service)와 상호 운용하여 웹 서비스의 고급 인증을 제공합니다.

STS는 웹 서비스 요청자와 웹 서비스 제공자 사이의 신뢰 관계를 중개하는 신뢰 씨드 파티 역할을 하는 웹 서비스입니다. SSL 핸드셰이크에서 인증 기관과 유사한 방식으로 STS는 요청자와 제공자가 메시지에 제공된 신임 정보를 "신뢰"할 수 있도록 보장합니다. 이 신뢰는 보안 토큰의 교환을 통해 표시됩니다. STS는 이러한 보안 토큰을 발행하고 교환하고 유효성 검증할 수 있고 다른 신뢰 도메인의 웹 서비스가 성공적으로 통신할 수 있도록 하는 신뢰 관계를 확립할 수 있습니다. 세부사항은 웹 서비스 신뢰 언어 스펙을 참조하십시오.

CICS는 신뢰 클라이언트 역할을 하고 두 가지 유형의 웹 서비스 요청을 STS에 보낼 수 있습니다. 첫 번째 유형의 요청은 WS-Security 메시지 헤더에 있는 보안 토큰을 유효성 검증하는 것이고 두 번째 유형의 요청은 다른 유형의 보안 토큰을 교환하는 것입

니다. 이러한 요청을 사용하면 CICS가 SAML 어설션과 Kerberos 토큰과 같은 다양한 신뢰 도메인의 다른 보안 토큰을 포함하는 메시지를 보내고 수신할 수 있습니다.

CICS가 STS와 상호작용하는 방법을 정의하도록 CICS 보안 핸들러를 구성하거나 별도로 제공된 신뢰 클라이언트 인터페이스를 사용할 수 있도록 자체 메시지 핸들러를 작성할 수 있습니다. 어떤 방법을 선택하든 SSL을 사용하여 CICS와 STS 간 연결을 보호하십시오.

보안 핸들러가 STS를 호출하는 방법

CICS 보안 핸들러는 파이프라인 구성 파일의 정보를 사용하여 STS(Security Token Service)에 웹 서비스 요청을 전송합니다. 전송되는 요청의 유형은 STS가 수행하게 할 조치에 따라 다릅니다.

서비스 제공자 파이프라인에서

서비스 제공자 파이프라인에서 보안 핸들러는 보안 핸들러를 구성하는 방식에 따라 두 가지 유형의 조치를 지원합니다.

- STS에 요청을 전송하여 보안 토큰의 첫 번째 인스턴스 또는 인바운드 메시지의 WS-Security 헤더에 있는 특정 유형의 첫 번째 보안 토큰을 유효성 검증합니다.
- STS에 요청을 전송하여 CICS가 이해할 수 있는 보안 토큰에 대해 인바운드 메시지의 WS-Security 헤더에 있는 특정 유형의 첫 번째 보안 토큰 또는 보안 토큰의 첫 번째 인스턴스를 교환합니다.

보안 핸들러는 동적으로 파이프라인을 작성하여 STS에 웹 서비스 요청을 전송합니다. 이 파이프라인은 STS에서 응답을 수신할 때까지 존재하고 그 후에는 삭제됩니다. 요청에 성공하면 STS가 ID 토큰을 리턴하거나 토큰의 유효성 상태를 리턴합니다. 보안 핸들러가 토큰에서 파생된 RACF ID를 DFHWS-USERID 컨테이너에 배치합니다.

STS에 오류가 발생하면 STS가 보안 핸들러에 SOAP 결함을 리턴합니다. 그러면 보안 핸들러가 웹 서비스 요청자로 결함을 다시 전달합니다.

서비스 요청자 파이프라인에서

서비스 요청자 파이프라인에서 보안 핸들러는 STS와의 토큰 교환만 요청할 수 있습니다. 파이프라인 구성 파일은 STS가 보안 핸들러에게 발행하는 토큰 유형을 정의합니다.

요청에 성공하면 RACF ID가 DFHWS-USERID 컨테이너에 배치되고 토큰은 아웃바운드 메시지 헤더에 포함됩니다. STS에 오류가 발생하면 STS가 보안 핸들러에 SOAP 결함을 리턴합니다. 그러면 보안 핸들러가 파이프라인을 통해 결함을 웹 서비스 요청자 애플리케이션에 다시 전달합니다.

보안 핸들러는 STS에서 파이프라인에 대해 한 가지 유형의 조치만 요청할 수 있습니다. 또한 아웃바운드 요청 메시지에 대해 한 가지 유형의 토큰만 교환할 수 있고 첫 번째 인스턴스 또는 특정 유형의 첫 번째 인스턴스 중 WS-Security 메시지 헤더의 첫 번째 토큰만 핸들링하도록 제한됩니다. 이러한 옵션으로 대부분의 STS 사용 공통 시나리오를 처리할 수 있지만 인바운드 및 아웃바운드 메시지를 핸들링하는 데 필요한 처리를 제공하지 않을 수도 있습니다.

인바운드 메시지 헤더에 있는 여러 토큰을 처리하기 위해 보다 특정한 처리를 제공하거나 아웃바운드 메시지에 대해 다양한 유형의 토큰을 교환하려는 경우 신뢰 클라이언트 인터페이스를 사용하십시오. STS에 자체 웹 서비스 요청을 전송하기 위해 이 인터페이스를 사용하여 사용자 정의 메시지 핸들러를 작성할 수 있습니다.

신뢰 클라이언트 인터페이스

신뢰 클라이언트 인터페이스를 사용하면 보안 핸들러를 사용하지 않고 직접 STS(Security Token Service)와 상호작용할 수 있습니다. 이 방식에서는 보안 핸들러에서 제공하는 처리에 비해 고급 토큰 처리를 제공하는 유연성을 가지게 됩니다.

신뢰 클라이언트 인터페이스는 CICS 제공 프로그램 DFHPIRT의 개선사항입니다. 이 프로그램은 보통 웹 서비스 요청자 애플리케이션이 CICS 웹 서비스 지원을 사용하여 배치되지 않은 경우에 파이프라인을 시작하는 데 사용됩니다. 그러나 STS에 대한 신뢰 클라이언트 인터페이스 역할도 할 수 있습니다.

메시지 핸들러 또는 헤더 처리 프로그램에서 DFHPIRT에 링크하여 신뢰 클라이언트 인터페이스를 호출할 수 있고 DFHWSTC-V1이라고 하는 채널 및 일련의 보안 컨테이너를 전달할 수 있습니다. 이러한 컨테이너를 사용하면 STS로부터 유효성 검증 또는 실행 조치를 요청하고 교환할 토큰 유형을 선택할 수 있으며 메시지 헤더로부터 적절한 토큰을 전달할 수 있는 유연성을 가질 수 있습니다. DFHPIRT는 파이프라인을 동적으로 작성하고 보안 컨테이너로부터 웹 서비스 요청을 구성하고 이 요청을 STS로 전송합니다.

DFHPIRT는 STS의 응답을 대기하고 DFHWS-RESTOKEN 컨테이너에서 이 응답을 메시지 핸들러로 다시 전달합니다. STS에 오류가 발생하면 SOAP 결함을 리턴합니다. DFHPIRT는 DFHWS-STSFault 컨테이너에 결함을 넣고 파이프라인에서 링크 프로그램으로 돌아갑니다.

서비스 제공자 및 서비스 요청 파이프라인에서 보안 핸들러를 사용 설정하지 않고 신뢰 클라이언트 인터페이스를 사용하거나 보안 핸들러에 추가하여 신뢰 클라이언트 인터페이스를 사용할 수 있습니다.

SOAP 메시지 서명

인바운드 메시지의 경우 CICS는 SOAP 본문의 요소 및 SOAP 헤더 블록에 대한 디지털 서명을 지원합니다. 아웃바운드 메시지의 경우 CICS는 SOAP 본문의 모든 요소에 서명합니다.

SOAP 메시지는 선택적 <Header> 요소와 필수 <Body> 요소를 포함하는 <Envelope> 요소로 구성되는 XML 문서입니다.

WSS: SOAP 메시지 보안 스펙에서는 <Header>와 <Body>의 콘텐츠를 요소 레벨에서 서인될 수 있습니다. 즉, 지정된 메시지에서 개별 요소에 서명하거나 서명하지 않거나 다른 서명 또는 다른 알고리즘을 사용하여 서명할 수 있습니다. 예를 들어, 온라인 구매 애플리케이션에서 사용되는 SOAP 메시지에서는 주문 수신을 확인하는 요소에 서명하는 것이 적절합니다. 이러한 요소는 법적 상태를 가질 수 있기 때문입니다. 그러나 전체 메시지 암호화의 오버헤드를 피하기 위해서 기타 정보는 미서명 상태로 안전하게 유지할 수 있습니다.

인바운드 메시징의 경우 보안 메시지 핸들러는 SOAP <Header> 및 <Body>에 있는 개별 요소의 디지털 서명을 확인할 수 있습니다.

- <Header>에서 발생하는 서명된 요소
- SOAP <Body>에 있는 서명된 요소. 서명된 본문을 예상하도록 핸들러를 구성하면 CICS가 본문이 서명되지 않은 SOAP 메시지를 거부하고 SOAP 결함을 발행합니다.

아웃바운드 메시지의 경우 보안 메시지 핸들러는 SOAP <Body>에만 서명할 수 있고 <Header>는 서명하지 않습니다. 본문 서명에 사용되는 알고리즘 및 키는 핸들러 구성 정보에서 지정됩니다.

서명 알고리즘

CICS는 XML 서명 스펙에서 요구하는 서명 알고리즘을 지원합니다. 각 알고리즘은 URI(Universal Resource Identifier)로 식별됩니다.

알고리즘	URI
DSA with SHA1(Digital Signature Algorithm with Secure Hash Algorithm 1) 인바운드 SOAP 메시지에 대해서만 지원됩니다.	http://www.w3.org/2000/09/xmldsig#dsa-sha1
RSA with SHA1(Rivest-Shamir-Adleman algorithm with Secure Hash Algorithm 1)	http://www.w3.org/2000/09/xmldsig#rsa-sha1

서명된 SOAP 메시지의 예제

이 예제는 CICS에 의해 서명된 SOAP 메시지를 보여줍니다.

```

<?xml version="1.0" encoding="UTF8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header>
  <wsse:Security xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:xenc="http://www.w3.org/2001/04/xmenc#" SOAP-ENV:mustUnderstand="1">
    <wsse:BinarySecurityToken
      EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
      ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"
      wsu:Id="x509cert00">MIICDCCAE2gAwIBAgIBADANBgkqhkiG9w0BAQUFADAwMQswCQYDVQQGEwJHcEMMAoGA1UEChMD
      SUJNMRMwEQYDVQQDEwpXaWxsIF1hdGVzMB4XDTA2MDEzMDEwMDAwMFoXDTA3MDEzMTEzNTk1OVow
      MDELMAkGA1UEBhMCRC01xDDAKBgNVBAoTA01CTTETMBEGA1UEAxMKV21sbCBZYXR1czCBnzANBgkq
      hkiG9w0BAQEFAA0BjQAwgYkCgYEArsRj/n+3RN75+jaxuOMBWShvZCB0egv8qu2UwLWEeiogePsR
      6Ku4SuHbBwjTWNr0xBTAAS9IEa70yhVdppx0nJB0CiERg7S0HudP7a8JXPfZA+BqV63JqRgJyxN6
      msfTAvmEMR07LIXmZAt62nwcFrvCKNPFcIJ5mkaJ9v1p7jkCAwEAAaOBTCBqjA/BglghkgBhvhC
      AQ0EMhMR2VuZXJhdGVkIGJ5IHROZS0BTZWN1cm10eSBTZKJ2ZXIgzM9yIHovT1MgKFJBQ0YpMDGg
      ZQVRFU0BVSY5JQk0uQ09ggdJQk0uQ09NhgTVIcuSUJNLkNPTycECRR1BjAO
    </wsse:BinarySecurityToken>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      <ds:SignedInfo xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
        xmlns:xenc="http://www.w3.org/2001/04/xmenc#"
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
          <c14n:InclusiveNamespaces xmlns:c14n="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="ds wsu xenc SOAP-ENV "/>
        </ds:CanonicalizationMethod>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <ds:Reference URI="#TheBody">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
              <c14n:InclusiveNamespaces xmlns:c14n="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="wsu SOAP-ENV "/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/> 2
          <ds:DigestValue>Q0RZEA+gpaf1uShspHxhrrjaFlXE=</ds:DigestValue> 3
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>drDH0XESiyN6YJm27mfK1ZMG4Q4IsZqQ9N9V6kEnw21k7aM3if77XNFnyKS4deglbc3ga11kkaFJ 4
        p4jLOmYRqgycDPpqpM+UEu7mzfHRQGe7H0EnFqZpikNqZK5FF6fvYlV2JgTDPwrOSYXmhzwegUDT
        1TVjOvuUgYrFya03pw=</ds:SignatureValue>
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#x509cert00"
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"/> 5
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </ds:Signature>
  </wsse:Security>
</SOAP-ENV:Header>
<SOAP-ENV:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="TheBody">
  <getVersion xmlns="http://msgsec.wssecvt.ws.ibm.com"/>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

1. 2진 보안 토큰은 X.509 인증서의 base64binary 인코딩을 포함합니다. 이 인코딩에는 SOAP 메시지의 계획된 수신인이 서명을 확인하는 데 사용하는 공개 키가 포함됩니다.
2. 메시지 다이제스트를 생성하기 위해 해싱 프로세스 중에 사용되는 알고리즘입니다.
3. 메시지 다이제스트의 값입니다.
4. 이 다이제스트 값은 사용자의 개인 키로 암호화되고 서명 값으로 여기에 포함됩니다.
5. 서명 인증에 사용되는 공개 키를 포함하는 2진 보안 토큰을 참조합니다.

암호화된 SOAP 메시지에 대한 CICS 지원

인바운드 메시지의 경우 CICS는 SOAP 본문에 있는 암호화된 요소 및 본문도 암호화되는 암호화된 SOAP 헤더 블록을 복호화할 수 있습니다. 아웃바운드 메시지의 경우 CICS는 전체 SOAP 본문을 암호화합니다.

SOAP 메시지는 선택적 <Header> 요소와 필수 <Body> 요소를 포함하는 <Envelope> 요소로 구성되는 XML 문서입니다.

WSS: SOAP 메시지 보안 스펙에서는 <Header> 요소의 일부 콘텐츠와 <Body> 요소의 모든 콘텐츠를 요소 레벨에서 암호화할 수 있습니다. 즉, 지정된 메시지에서 개별 요소는 여러 암호화 레벨을 가지거나 서로 다른 알고리즘을 사용하여 암호화될 수 있습니다. 예를 들어, 온라인 구매 애플리케이션에서 사용되는 SOAP 메시지에서는 기밀 상태로 유지할 수 있도록 개인의 신용카드 세부사항을 암호화하는 것이 적절합니다. 그러나 전체 메시지 암호화의 오버헤드를 피하기 위해서 일부 정보는 보안 강도가 상대적으로 낮은(그러나 더 빠른) 알고리즘을 사용하여 암호화할 수 있으며, 기타 정보는 비암호 상태로 안전하게 유지할 수 있습니다.

인바운드 메시지의 경우 CICS 제공 보안 메시지 핸들러는 SOAP <Body>에 있는 개별 요소를 복호화하고 SOAP 본문도 암호화되는 경우에는 SOAP <Header>의 요소를 복호화할 수 있습니다. 보안 메시지 핸들러는 항상 다음과 같은 요소를 복호화합니다.

- 요소가 발견되는 순서대로 <Header> 요소에서 발생하는 요소
- SOAP <Body> 요소에 있는 요소. 암호화된 <Body>가 없는 SOAP 메시지를 거부하려면 <expect_encrypted_body> 요소를 사용하여 암호화된 본문을 예상하도록 핸들러를 구성하십시오.

아웃바운드 메시지의 경우 보안 메시지 핸들러는 SOAP <Body> 콘텐츠의 암호화만 지원하고 <Header> 요소에 있는 요소는 암호화하지 않습니다. 보안 메시지 핸들러가 <Body> 요소를 암호화할 때 본문의 모든 요소는 같은 알고리즘으로 같은 키를 사용하여 암호화됩니다. 알고리즘 및 키에 대한 정보는 핸들러에 대한 구성 정보에서 지정됩니다.

암호화 알고리즘

CICS는 XML 암호화 스펙에 필요한 암호화 알고리즘을 지원합니다. 각 알고리즘은 URI(Universal Resource Identifier)로 식별됩니다.

알고리즘	URI
3-DES(Data Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmlenc#tripleDES-cbc
키 길이가 128비트인 AES(Advanced Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmlenc#aes128-cbc
키 길이가 192비트인 AES(Advanced Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmlenc#aes192-cbc
키 길이가 256비트인 AES(Advanced Encryption Standard) 알고리즘	http://www.w3.org/2001/04/xmlenc#aes256-cbc

암호화된 SOAP 메시지의 예제

이 SOAP 메시지 예제는 CICS에 의해 암호화되었습니다.

```
<?xml version="1.0" encoding="UTF8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header>
  <wsse:Security xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" SOAP-ENV:mustUnderstand="1">

    <wsse:BinarySecurityToken
      EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ❶
      ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"
      wsu:Id="x509cert00">MIICDCCAe2gAwIBAgIBADANBgkqhkiG9w0BAQUFADAwMQswCQYDVQQGEWJHQjEMMAoGA1UEChMD
      SUJNMRMwEQYDVQQDEWpXaWxsIFllhdGVzMzB4XDAtMDEZMTAwMDAwMFOxDTA3MDEZMTIzNTk1OVow
      MDELMAKGA1UEBhMCRC0lxdDAKBgNVBAAoTA01CTTETMBEGA1UEAxMKV21sbCBZYXRlc2CBnzANBgkq
      hkiG9w0BAQEFAAOBjQAwYkKgYEArSj/n+3RN75+jaxuOMBvSHvZCB0egv8qu2UwLEieogePsR
      6Ku4SuHbBwJtWNr0xBTAAAS9IEa70yhVdppxOnJB0CiERg7S0HUdp7a8JXPfZA+Bqv63JqRgJyxN6
      msfTAvEMR07LIXmAte62nwcFrvcKNPCFIJ5mkaj9vlp7jkCAwEAaA0BrTCBqJA/BglghkgBhvhhc
      AQOEMHMwR2VuZVxjdGVkIGJ5IHRobSBTZW50cm10eSBTZXJ2ZXJ1gZm9yIHovT1MgKFBJB0YpMDgg
      A1UdeQXXmc+BEVDZVRFRU0BVSY5JQk0uQ09NggdJQk0uQ09NhgtXV1cuSUJNLKNTPTCECRRI BjAO
      BgNVNHQ8BAf8EBAMCAfyWHQYDVR0BBYEFMiPX6VZKP5+mSOY1TLNGQVvJzu+MA0GSQGScGsIb3QDEB
      BUAA4GBAHdrS409Jhoe67pHL2gs7x4SpV/NouJnn/w25sjjop3RLgJ2bktK6RIEevhCDim6tnYW
      NyjBL1Vdn7u5M6ktFd+HutR/HnIrQ3pkXZKAipgcORWDJ+8APLYSCxtFL+J0LN9EO6yjIHL6mq
      uzbTH2LvZFMy4PqEbmvKbmA87a1F

    </wsse:BinarySecurityToken>

    <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/> ❷
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#x509cert00"
            ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"/> ❸
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </xenc:EncryptedKey>
      <xenc:CipherData>
        <xenc:CipherValue>M6bDQtJrvX0PEjAEIcf6bq6MP3ySmB4TQ0a/B5U1Qj1vWjD56V+GRJbF7CZ5SejwCJHRVKW1ZB5 ❹
          Mb+aUzSWls0zhQixclJchgwCiyIn+E2TbG3R9m0zHD3XQsKTyVaOTIR7VPomBdiZLNDImxjZn2
          p7JfxxyKkObcSLhdZnc=</xenc:CipherValue>
        </xenc:CipherData>
        <xenc:ReferenceList>
          <xenc:DataReference URI="#Enc1"/>
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
    </wsse:Security>
  </SOAP-ENV:Header>
<SOAP-ENV:Body>
  <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="Enc1" Type="http://www.w3.org/2001/04/xmlenc#Content">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/> ❺
    <xenc:CipherData>
      <xenc:CipherValue>kgqvKnMcgiUn7rl1vkFXF0g4SodEd3dxAo/mVN6ef211B1MZeIg70yJEhf4ZXwlCdtoFbidIdnk ❻
        rksqllMpww6So7ID8zav+KPQUKGm4+E=</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

1. 2진 보안 토큰은 X.509 인증서의 base64binary 인코딩을 포함합니다. 이 인코딩에는 대칭 키 암호화에 사용된 공개 키가 포함됩니다.
2. 대칭 키 암호화에 사용된 알고리즘을 명시합니다.
3. 대칭 키 암호화에 사용된 공개 키를 포함하는 2진 보안 토큰을 참조합니다.
4. 메시지 암호화에 사용된 암호화된 대칭 키입니다.
5. 메시지 암호화에 사용된 암호화 알고리즘입니다.
6. 암호화된 메시지입니다.

웹 서비스 보안을 위한 RACF 구성

RACF와 같은 외부 보안 관리자를 구성하여 아웃바운드 SOAP 메시지를 서명하고 암호화하기 위한 공개-개인 키 쌍 및 X.509 인증서를 작성하고 서명 및 암호화된 인바운드 SOAP 메시지를 인증 및 복호화해야 합니다.

시작하기 전에

이 작업을 수행하기 전에 RACF는 CICS와 함께 작동하도록 설정되어 있어야 합니다. 웹 서비스 파이프라인을 포함하는 CICS 리전에서 **DFLTUSER**, **KEYRING** 및 **SEC=YES** 시스템 초기화 매개변수를 지정하십시오.

참고: 같은 **KEYRING**에서 같은 식별 이름을 가지는 다중 인증서는 지원되지 않습니다.

프로시저

1. 서명되는 인바운드 SOAP 메시지를 인증하려면 다음을 수행하십시오.

- a. RACF에 X.509 인증서를 ICSF 키로 가져오십시오.
- b. **RACDCERT** 명령을 사용하여 **KEYRING** 시스템 초기화 매개변수에 지정된 키 링에 인증서를 첨부하십시오.

```
RACDCERT ID(userid1)
CONNECT(ID(userid2) LABEL('label-name') RING(ring-name))
```

여기서,

- *userid1*은 키 링의 기본 사용자 ID이거나 인증서를 다른 사용자 ID의 키 링에 첨부할 수 있는 권한을 가집니다.
 - *userid2*는 인증서와 연관시킬 사용자 ID입니다.
 - *label-name*은 인증서의 이름입니다.
 - *ring-name*은 **KEYRING** 시스템 초기화 매개변수에 지정되는 키 링의 이름입니다.
- c. 옵션: 어설션된 ID를 사용하려는 경우 인증서와 연관된 사용자 ID에 다른 사용자 ID로 작업을 실행할 수 있도록 하는 대리 권한이 있는지 확인하십시오. 또한 SOAP 메시지 헤더에 포함된 추가 인증서도 RACF로 가져오는지 확인하십시오.

SOAP 메시지는 인증서를 포함하거나 인증서에 대한 참조를 포함하는 헤더에 2진 보안 토큰을 포함할 수 있습니다. 이 참조는 **KEYNAME**(RACF에서 인증서 레이블), **ISSUER** 및 **SERIAL** 번호의 조합 또는 **SubjectKeyIdentifier**일 수 있습니다. CICS는 RACF의 인증서 정의에서 속성으로 지정된 경우에만 **SubjectKeyIdentifier**를 인식할 수 있습니다.

2. 아웃바운드 SOAP 메시지에 서명하려면 다음을 수행하십시오.

- a. 다음의 **RACDCERT** 명령을 사용하여 X.509 인증서 및 공개-개인 키 쌍을 작성하십시오.

```
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')
```

여기서, *userid2*는 인증서와 연관시킬 사용자 ID입니다. 인증서 *label-name* 값을 지정할 때 다음 문자는 사용하지 마십시오.

< > ; ! =

- b. 인증서를 **KEYRING** 시스템 초기화 매개변수에 지정된 키 링에 첨부하십시오. **RACDCERT** 명령을 사용하십시오.

- c. 인증서를 내보내고 SOAP 메시지의 계획된 수신인에게 공개하십시오.

CICS가 서명을 유효성 검증하기 위해 계획된 수신인에 대한 SOAP 메시지 헤더의 2진 보안 토큰에 X.509 인증서를 자동으로 포함시키도록 파이프라인 구성 파일을 편집할 수 있습니다.

3. 암호화된 인바운드 SOAP 메시지를 복호화하려면 SOAP 메시지가 키 쌍의 일부인 공개 키를 포함해야 합니다. 여기서, 공개 키는 CICS에서 정의됩니다.

- a. 암호화를 위해 공개-개인 키 쌍 및 인증서를 RACF에서 생성하십시오. 키 쌍 및 인증서는 ICSF를 사용하여 생성해야 합니다.

- b. 인증서를 **KEYRING** 시스템 초기화 매개변수에 지정된 키 링에 첨부하십시오. **RACDCERT** 명령을 사용하십시오.

- c. 인증서를 내보내고 복호화할 SOAP 메시지의 생성기에 공개하십시오.

그러면 SOAP 메시지의 생성기는 공개 키를 포함하는 인증서를 가져와서 SOAP 메시지를 암호화하는 데 사용할 수 있습니다. SOAP 메시지는 공개 키를 포함하거나 이 키에 대한 참조를 포함하는 헤더에 2진 보안 토큰을 포함할 수 있습니다. 이 참조는 KEYNAME, ISSUER 및 SERIAL 번호의 조합 또는 SubjectKeyIdentifier 일 수 있습니다. CICS는 RACF의 공개 키 정의에서 속성으로 지정된 경우에만 SubjectKeyIdentifier를 인식할 수 있습니다.

4. 아웃바운드 SOAP 메시지를 암호화하려면 다음을 수행하십시오.

- a. 암호화에 사용할 공개 키가 포함된 인증서를 ICSF 키로 RACF에 가져오십시오. SOAP 메시지를 복호화하려면 계획된 수신인에게 공개 키와 연관된 개인 키가 있어야 합니다.

- b. 공개 키를 포함하는 인증서를 **KEYRING** 시스템 초기화 매개변수에 지정된 키 링에 첨부하십시오. **RACDCERT** 명령을 사용하십시오.

CICS는 인증서의 공개 키를 사용하여 SOAP 본문을 암호화하고 SOAP 메시지 헤더에 있는 2진 보안 토큰으로서 공개 키를 포함하는 인증서를 전송합니다. 공개 키는 파이프라인 구성 파일에서 정의됩니다.

다음에 수행할 작업

아웃바운드 메시지 서명 및 암호화를 위한 이 구성에서는 사용되는 인증서를 CICS 리전 사용자 ID가 소유해야 합니다. 이 인증서는 CICS 리전 사용자 ID가 소유합니다. RACF는 인증서 소유자만 서명 및 암호화 프로세스에 사용되는 개인 키를 추출하도록 허용하기 때문입니다.

CICS가 소유하지 않은 인증서를 사용하여 메시지를 서명하고 암호화해야 하는 경우 CICS 리전 사용자 ID가 소유하지 않은 기존 인증서 사용의 지시사항에 따라 CICS 시스템 사이에서 단일 인증서를 공유할 수 있습니다.

ID 전파를 위한 제공자 모드 웹 서비스 구성

웹 서비스 요청을 포함한 ID 전파는 신뢰 기반 구성에 의존합니다(예: WebSphere DataPower에서 클라이언트 인증 SSL 연결 사용). 이 태스크에서는 신뢰 클라이언트에서 전송된 WS-Security 헤더에서 ICRX ID 토큰을 예상하도록 PIPELINE 자원을 구성합니다.

시작하기 전에

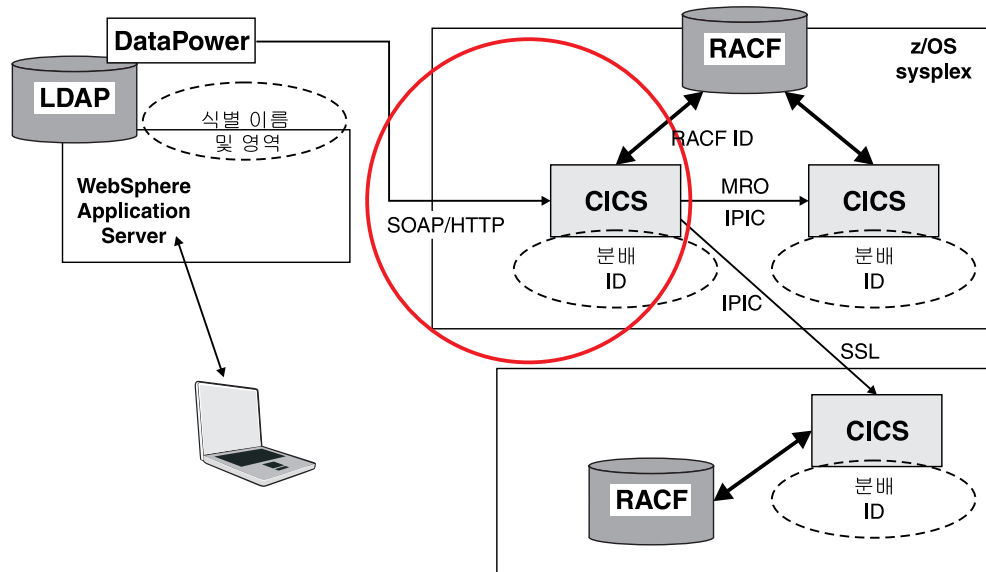
웹 서비스 연결을 구성하기 전에 RACF RACMAP 설정을 구성해야 합니다. 그렇지 않으면 RACF에 전송한 맵핑되지 않은 모든 요청에 대해 RACF ICH408I 메시지를 수신합니다. RACF **RACMAP** 명령 구성에 대한 자세한 정보는 ID 전파를 위한 RACF 구성의 내용을 참조하십시오.

WebSphere DataPower 어플라이언스와 CICS 사이에 신뢰 관계를 구성해야 합니다(예: WebSphere DataPower와 CICS 사이에 SSL 클라이언트 인증 사용). WebSphere DataPower가 스스로를 식별하는 데 사용하는 디지털 인증서는 사용자 ID와 연관되어 있어야 하고 이 ID에는 ID를 어설션하기 위한 대리 권한이 부여되어야 합니다. 대리 권한에 대한 자세한 정보는 대리 사용자 보안의 내용을 참조하십시오.

이 태스크 정보

이 태스크에서는 WebSphere DataPower 어플라이언스에서 CICS를 사용하여 안전하고 강력한 방식으로 분배 ID를 전파할 수 있는 웹 서비스 구성을 제공하는 방법을 설명합니다. 다이어그램에 표시된 원은 이 태스크가 CICS 특정 구성을 설명함을 나타냅니다.

그림 39. WebSphere DataPower로부터 ICRX ID 토큰을 예상하는 CICS 구성



WebSphere DataPower는 CICS와 다른 애플리케이션 사이에서 매개체 역할을 합니다. 원격 웹 서비스 요청자 애플리케이션은 SOAP 프로토콜을 사용하여 WebSphere DataPower 어플라이언스에 연결합니다. WebSphere DataPower는 원격 클라이언트에서 제공되는 신임 정보를 인증하고 이 신임 정보를 사용자의 분배 id를 식별하는 z/OS ICRX ID 토큰으로 맵핑합니다. 그러면 SOAP 메시지가 WS-Security 헤더의 ICRX ID 토큰과 함께 신뢰 SSL 연결을 통해 CICS에 전달됩니다. ICRX ID 토큰에 대한 자세한 정보는 z/OS Security Server RACF 데이터 영역의 내용을 참조하십시오.

CICS는 WebSphere DataPower로부터 SOAP 메시지를 수신합니다. PIPELINE 구성 파일은 *blind* 신뢰를 지정합니다. 유일하게 가능한 클라이언트가 WebSphere DataPower 어플라이언스이고 WebSphere DataPower는 보안 SSL 연결을 통해 CICS와 통신하기 때문입니다. 따라서 PIPELINE 구성 파일에서 추가 인증을 지정하지 않아도 됩니다. WS-Security 핸들러 프로그램은 WS-Security 헤더에서 찾은 첫 번째 ICRX를 찾고 이 ICRX를 사용자 식별에 사용합니다.

프로시저

1. PIPELINE 자원이 ICRX를 수신할 수 있게 하는 기본 ICRX 모드를 지정하도록 PIPELINE 자원을 작성하거나 기존 PIPELINE 자원을 편집하십시오. 가장 전형적인 조합은 블라인드 트러스트와 기본-ICRX 모드입니다. PIPELINE 자원 요소에 대한 자세한 정보는 316 페이지의 『<authentication> 요소』의 내용을 참조하십시오.

다음은 블라인드 트러스트와 기본-ICRX 모드를 보여주는 예제 PIPELINE 구성 파일입니다.

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

다음은 블라인드 트러스트를 사용하는 ICRX ID를 포함하는 예제 SOAP 메시지입니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
      SOAP-ENV:mustUnderstand="1">

      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
        wsu:Id="ICRX"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-utility-1.0.xsd"
        ValueType="http://www.ibm.com/xmlns/prod/zos/saf#ICRXV1">

          ICRX IS HERE

        </wsse:BinarySecurityToken>

      </wsse:Security>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>

      APPLICATION SPECIFIC XML IS HERE

    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

2. WebSphere DataPower가 ICRX 정보를 전송할 수 있도록 구성되는지 확인하십시오. ID 전파를 사용하기 위한 샘플 네트워크 토폴로지의 내용을 참조하십시오.

결과

WS-Security 헤더의 ICRX ID 토큰을 포함하며 클라이언트 인증 SSL 연결을 통해 연결되는 WebSphere DataPower의 웹 서비스 요청을 이제 플로우할 수 있습니다.

관련 태스크:

ID 전파를 위한 RACF 구성

ID 전파를 위한 IPIC 연결 구성

웹 서비스 보안을 위한 파이프라인 구성

웹 서비스 보안(WSS)을 지원하도록 파이프라인을 구성하려면 파이프라인 구성 파일에 보안 핸들러를 추가해야 합니다. CICS에서 제공된 보안 핸들러를 사용하거나 설명한 대로 자체 보안 핸들러를 작성할 수 있습니다.

시작하기 전에

CICS 제공 보안 핸들러를 정의하기 전에 WSS의 구성 정보를 추가할 파이프라인 구성 파일을 식별하거나 작성해야 합니다.

프로시저

1. 파이프라인에 `<wsse_handler>` 요소를 추가하십시오. 핸들러는 서비스 제공자 또는 요청자 파이프라인의 `<service_handler_list>` 요소에 포함되어야 합니다. 다음 요소를 코드화하십시오.

```
<wsse_handler>
  <dfhwsse_configuration version="1">

    </dfhwsse_configuration>
  </wsse_handler>
```

`<dfhwsse_configuration>` 요소는 구성에 있는 다른 요소를 위한 컨테이너입니다.

2. 옵션: `<인증>` 요소를 코드화하십시오.
 - 서비스 요청자 파이프라인에서 `<authentication>` 요소는 아웃바운드 SOAP 메시지의 보안 헤더에서 사용해야 하는 인증 유형을 지정합니다.
 - 서비스 제공자 파이프라인에서 이 요소는 CICS가 작업을 처리할 사용자 ID를 판별하기 위해 인바운드 SOAP 메시지에서 보안 토큰을 사용하는지 여부를 지정합니다.
 - a. 어설션된 ID가 사용되는지 여부 및 서비스 제공자와 요청자 간 신뢰 관계의 네이처를 지정하도록 **trust** 속성을 코드화하십시오. **trust** 속성에 대한 세부사항은 316 페이지의 『`<authentication>` 요소』의 내용을 참조하십시오.
 - b. 옵션: **trust=none**을 지정한 경우 메시지에서 찾은 신임 정보를 처리하는 방법을 지정하도록 **mode** 속성을 코드화하십시오. **mode** 속성에 대한 세부사항은 316 페이지의 『`<authentication>` 요소』의 내용을 참조하십시오.
 - c. `<authentication>` 요소에서 다음 요소를 코드화하십시오.
 - 1) 선택적으로 비어 있는 `<suppress/>` 요소.

서비스 제공자 파이프라인에서 이 요소가 지정되는 경우 핸들러는 작업을 실행하는 사용자 ID를 판별하기 위해 메시지의 보안 토큰을 사용하려고 시도하지 않습니다.

서비스 요청자 파이프라인에서 이 요소가 지정되는 경우 핸들러는 아웃바운드 SOAP 메시지에 인증에 필요한 보안 토큰을 추가하려고 시도하지 않습니다.

- 2) 요청자 파이프라인에서 SOAP 메시지의 본문에 서명하는 데 사용되는 알고리즘의 URI를 지정하는 선택적 <algorithm> 요소. 신뢰 및 모드 속성 값의 조합이 메시지가 서명됨을 나타내는 경우 이 요소를 지정해야 합니다. 이 요소에서 SHA1 알고리즘을 포함하는 RSA만 지정할 수 있습니다. 이 URI는 <http://www.w3.org/2000/09/xmldsig#rsa-sha1>입니다.
- 3) RACF에 설치된 X.509 디지털 인증서와 연관된 레이블을 지정하는 선택적 <certificate_label> 요소. 서비스 요청자 파이프라인에서 이 요소를 지정하고 <suppress> 요소가 지정되지 않은 경우에 인증서는 SOAP 메시지의 보안 헤더에 추가됩니다. <certificate_label> 요소를 지정하지 않는 경우 CICS는 RACF 키 링의 기본 인증서를 사용합니다.

이 요소는 서비스 제공자 파이프라인에서 무시됩니다.

3. 옵션: <sts_authentication> 요소를 <authentication> 요소에 대한 대체로 코드화하십시오. 파이프라인 구성 파일에서 둘 다 코드화해서는 안됩니다. 이 요소는 STS(Security Token Service)가 인증에 사용되며 전송되는 요청의 유형을 판별하도록 지정합니다.

- a. 옵션: 서비스 제공자 모드에서만 STS가 보안 토큰을 확인하거나 교환하는지 여부를 지정하도록 **action** 속성을 코드화하십시오. **action** 속성에 대한 세부사항은 320 페이지의 『<sts_authentication> 요소』의 내용을 참조하십시오.

- b. <sts_authentication> 요소 내에서 다음 요소를 코드화하십시오.

- 1) <auth_token_type> 요소. 이 요소는 서비스 요청자 파이프라인에서 <sts_authentication> 요소를 지정하는 경우에는 필수이고 서비스 제공자 파이프라인에서는 선택적입니다. 자세한 정보는 <auth_token_type>의 내용을 참조하십시오.

- 서비스 요청자 파이프라인에서 <auth_token_type> 요소는 CICS가 DFHWS-USERID 컨테이너에 포함된 사용자 ID를 전송할 때 STS가 발행하는 토큰의 유형입니다. STS에서 CICS가 수신하는 토큰은 아웃바운드 메시지의 헤더에 배치됩니다.
- 서비스 제공자 파이프라인에서 <auth_token_type> 요소는 CICS가 메시지 헤더에서 가져와서 교환 또는 유효성 검증을 위해 STS로 전송하는 ID 토큰을 판별하는 데 사용됩니다. CICS는 메시지 헤더에서 지정된 유형의 ID 토큰을 먼저 사용합니다. 이 요소를 지정하지 않으면 CICS는 메시지 헤더에서 발견하는 첫 번째 ID 토큰을 사용합니다. CICS는 다음을 ID 토큰으로 간주하지 않습니다.

– wsu:Timestamp

- xenc:ReferenceList
- xenc:EncryptedKey
- ds:Signature

2) 비어 있는 선택적 <suppress/> 요소(서비스 제공자 파이프라인에서만 해당됨). 이 요소가 지정되면 핸들러가 작업을 실행하는 사용자 ID를 판별하기 위해 메시지의 보안 토큰을 사용하려고 시도하지 않습니다. <suppress/> 요소는 STS에 의해 리턴되는 ID 토큰을 포함합니다.

4. 옵션: <sts_endpoint> 요소를 코드화하십시오. 이미 <sts_authentication> 요소를 지정한 경우에만 이 요소를 사용하십시오. <sts_endpoint> 요소에서 다음 요소를 코드화하십시오.

- <endpoint> 요소. 이 요소에는 네트워크에서 STS(Security Token Service)의 위치를 가리키는 URI가 포함됩니다. HTTP를 사용하지 않고 SSL 또는 TLS를 사용하여 STS 보안에 대한 연결을 유지하도록 권장됩니다.

SAML 지원을 사용하려면 엔드포인트를 cics://PROGRAM/DFHSAML로 설정하십시오.

JMS 형식의 URI를 사용하여 WebSphere MQ 엔드포인트를 지정할 수도 있습니다.

- 선택적 <jvmserver> 요소. 이 요소는 SAML 토큰 서비스를 실행하도록 구성되는 JVM 서버를 식별합니다. 이 요소가 포함되지 않은 경우 기본 샘플 자원 JVM 서버 DFHXSTS가 가정됩니다. 이 요소는 SAML을 사용하는 경우에만 유효하며 다른 상황에서 사용하는 경우에는 오류가 발생합니다.

5. 옵션: 인바운드 SOAP 메시지가 디지털로 서명되어야 하는 경우 비어 있는 <expect_signed_body/> 요소를 코드화하십시오.

<expect_signed_body/> 요소는 인바운드 메시지의 <body>이 서명되어야 함을 표시합니다. 인바운드 메시지의 본문이 올바르게 서명되지 않은 경우 CICS는 보안 결함을 이유로 이 메시지를 거부합니다.

6. 옵션: 디지털로 서명되는 인바운드 SOAP 메시지를 거부하려면 비어 있는 <reject_signature/> 요소를 코드화하십시오.

7. 옵션: 인바운드 SOAP 메시지를 암호화해야 하는 경우 비어 있는 <expect_encrypted_body/> 요소를 코드화하십시오.

<expect_encrypted_body/> 요소는 인바운드 메시지의 <body>가 암호화되어야 함을 표시합니다. 인바운드 메시지의 본문이 올바르게 암호화되지 않은 경우 CICS는 보안 결함을 이유로 이 메시지를 거부합니다.

8. 부분적으로 또는 완전히 암호화되는 인바운드 SOAP 메시지를 거부하려면 비어 있는 <reject_encryption/> 요소를 코드화하십시오.

9. 옵션: 아웃바운드 SOAP 메시지가 서명되어야 하는 경우 <sign_body> 요소를 코드화하십시오.

a. <sign_body> 요소에서 <algorithm> 요소를 코드화하십시오.

b. <algorithm> 요소에 따라 <certificate_label> 요소를 코드화하십시오.

다음은 완성된 <sign_body> 요소의 예제입니다.

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

10. 옵션: 아웃바운드 SOAP 메시지를 암호화해야 하는 경우 <encrypt_body> 요소를 코드화하십시오.

a. <encrypt_body> 요소에서 <algorithm> 요소를 코드화하십시오.

b. <algorithm> 요소에 따라 <certificate_label> 요소를 코드화하십시오.

다음은 완성된 <encrypt_body> 요소의 예제입니다.

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

예제

다음 예제는 대부분의 선택적 요소가 표시되는 완성된 보안 핸들러를 보여줍니다.

```
<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>
```

사용자 정의 보안 핸들러 작성

자체 보안 프로시저 및 처리를 사용하려는 경우 사용자 정의 메시지 핸들러를 작성하여 파이프라인에서 보안 SOAP 메시지를 처리할 수 있습니다.

시작하기 전에

보안 핸들러가 지원해야 하는 보안 레벨을 결정하고 메시지에 지원되지 않는 보안이 포함될 경우 적절한 SOAP 결함이 리턴되는지 확인해야 합니다.

이 태스크 정보

또한 메시지 핸들러는 인바운드 및 아웃바운드 메시지에서 보안에 대처할 수 있어야 합니다.

프로시저

1. **EXEC CICS GET CONTAINER** 명령을 사용하여 DFHREQUEST 또는 DFHRESPONSE 컨테이너를 검색하십시오.
2. XML을 구문 분석하여 WS-Security 메시지 헤더에 있는 보안 토큰을 찾으십시오. 이 헤더는 <wsse:Security> 요소로 시작됩니다. 보안 토큰은 사용자 이름 및 비밀번호, 디지털 인증서 또는 암호화 키일 수 있습니다. 메시지는 보안 헤더에 다수의 토큰이 있을 수 있기 때문에 핸들러는 처리를 위해 정확한 토큰을 식별해야 합니다.
3. 메시지에서 구현되는 보안에 따라 적절한 처리를 수행하십시오.
 - a. Kerberos 토큰의 기본 인증을 수행하려면 **EXEC CICS VERIFY TOKEN** 명령을 실행하십시오. 이 명령은 제공된 Kerberos 토큰이 유효한지 확인합니다. 명령이 성공하면 **EXEC CICS PUT CONTAINER**로 DFHWS-USERID 컨테이너를 갱신하십시오. 그렇지 않으면 **EXEC CICS SOAPFAULT CREATE** 명령을 실행하십시오.
 - b. 비밀번호 또는 비밀번호 문구 기본 인증을 수행하려면 **EXEC CICS VERIFY PHRASE** 명령을 실행하십시오. 이 명령은 메시지의 보안 헤더에 있는 사용자 이름 및 비밀번호를 확인합니다. 명령이 성공하면 **EXEC CICS PUT CONTAINER**로 DFHWS-USERID 컨테이너를 갱신하십시오. 그렇지 않으면 **EXEC CICS SOAPFAULT CREATE** 명령을 실행하십시오.
 - c. STS(Security Token Service)로 다양한 토큰을 교환하거나 유효성 검증하여 고급 인증을 수행하려는 경우 신뢰 클라이언트 인터페이스를 사용하십시오. 세부 사항은 697 페이지의 『메시지 핸들러에서 트러스트 클라이언트 호출』의 내용을 참조하십시오.
 - d. 메시지에 서명되면 디지털 인증서의 신임 정보를 유효성 검증하십시오.
 - e. 메시지의 일부가 암호화되는 경우 보안 헤더의 정보를 사용하여 메시지를 복호화하십시오. WSS(Web Services Security): SOAP 메시지 보안 스펙은 이를 수행하는 방법에 대한 정보를 제공합니다.

결과

CICS에서 보안 핸들러 프로그램을 정의하고 파이프라인 구성 파일을 갱신하여 이 프로그램이 XML에서 올바르게 배치되는지 확인합니다. 서비스 요청자 파이프라인 구성

파일에서 보안 핸들러는 파이프라인의 끝에서 실행되도록 구성되어야 합니다. 서비스 제공자 파이프라인 구성 파일에서 보안 핸들러는 파이프라인의 시작에서 실행되도록 구성되어야 합니다.

다음에 수행할 작업

사용자 정의 메시지 핸들러를 작성하는 방법에 대한 일반 정보는 IBM Redbooks 서적 *CICS 웹 서비스용 애플리케이션 개발*(<http://www.redbooks.ibm.com/abstracts/sg247126.html>에 있음)을 참조하십시오.

메시지 핸들러에서 트러스트 클라이언트 호출

CICS는 STS(Security Token Service)를 호출하기 위해 자체 메시지 핸들러를 작성할 수 있도록 인터페이스를 제공합니다. 이 인터페이스를 사용하여 CICS 제공 보안 핸들러보다 보다 고급 처리를 수행할 수 있습니다.

시작하기 전에

이 태스크 정보

보안 핸들러 대신 또는 보안 핸들러에 추가하여 신뢰 클라이언트를 사용할 수 있습니다. 신뢰 클라이언트 인터페이스를 사용하려면 다음을 수행하십시오.

프로시저

1. 인바운드 또는 아웃바운드 메시지의 보안 메시지 헤더에서 올바른 토큰을 추출하십시오.
2. 채널 DFHWSTC-V1 및 다음 필수 컨테이너를 통과하는 프로그램 DFHPIRT에 링크하십시오.
 - DFHWS-STSUR, 네트워크의 STS 위치를 포함함
 - DFHWS-STSACTION, STS가 수행해야 되는 요청 유형의 URI를 포함함. 지원되는 두 개의 조치는 실행 및 유효성 검증입니다.
 - DFHWS-IDTOKEN, STS에서 확인 또는 교환해야 하는 토큰을 포함함
 - DFHWS-TOKENTYPE, STS가 응답으로 돌려 보내야 하는 토큰 유형을 포함함
 - DFHWS-SERVICEURI, 호출 중인 웹 서비스 조작의 URI를 포함함선택적으로 DFHWS-XMLNS 컨테이너를 포함시켜 보안 토큰을 포함하는 SOAP 메시지의 네임스페이스를 제공할 수 있습니다. 이 컨테이너에 대해서는 342 페이지의 『헤더 처리 프로그램 인터페이스』에 자세히 설명되어 있습니다.
3. DFHPIRT는 STS의 응답을 사용하여 리턴합니다. 성공적인 응답은 DFHWS-RESTOKEN 컨테이너에 저장됩니다.

STS에 요청과 관련된 문제점이 발생하면 SOAP 결함을 리턴합니다. DFHPIRT는 SOAP 결함을 DFHWS-STSFault 컨테이너에 넣습니다. STS가 SOAP 결함의 발행 이유를 제공하면 이 이유는 DFHWS-STsReason 컨테이너에 배치됩니다.

이상 종료가 발생하면 처리 오류의 세부사항을 포함하는 DFHError 컨테이너가 리턴됩니다.

메시지 핸들러는 이러한 응답을 처리하고 오류 발생 시 적절한 처리를 수행해야 합니다. 예를 들어, 메시지 핸들러는 웹 서비스 요청자에게 적절한 SOAP 결함을 리턴할 수 있습니다.

4. 응답을 적절하게 처리하십시오. 제공자 모드에서 파이프라인 처리는 메시지가 애플리케이션 핸들러에 도달할 때 CICS가 이해할 수 있는 사용자 이름이 DFHWS-USERID 컨테이너에 배치되는지 확인해야 합니다. 요청자 모드에서 메시지 핸들러는 아웃바운드 메시지 보안 헤더에 올바른 토큰을 추가해야 합니다.

다음에 수행할 작업

메시지 핸들러 작성을 완료했으면 CICS에서 프로그램을 배치하고 해당 파이프라인 구성 파일을 갱신하십시오. 서비스 요청자 파이프라인에서 파이프라인 처리의 끝에서 발생하지만 CICS 제공 보안 핸들러 앞에 있도록 메시지 핸들러를 정의하십시오. 서비스 제공자 파이프라인에서 파이프라인의 시작부에서 발생하지만 CICS 제공 보안 핸들러 뒤에 있도록 메시지 핸들러를 정의하십시오.

관련 참조:

374 페이지의 『DFHWS-STsURI 컨테이너』

DFHWS-STsURI는 DATATYPE(CHAR) 컨테이너입니다. SOAP 메시지의 ID 토큰을 발행하거나 유효성을 검증하는 데 사용되는 보안 토큰 서비스(STS)의 절대 URI를 포함합니다.

373 페이지의 『DFHWS-STsACTION 컨테이너』

DFHWS-STsACTION은 DATATYPE(CHAR) 컨테이너입니다. 보안 토큰 서비스(STS)가 보안 토큰을 발행하거나 유효성을 검증하기 위해 수행하는 조치의 URI를 포함합니다.

372 페이지의 『DFHWS-IDTOKEN 컨테이너』

DFHWS-IDTOKEN은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 보안 토큰 서비스(STS)가 메시지의 ID 토큰을 발행하기 위해 사용하거나 유효성을 검증하는 토큰을 포함합니다.

374 페이지의 『DFHWS-TOKENTYPE 컨테이너』

DFHWS-TOKENTYPE은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 보안 토큰 서비스(STS)가 SOAP 메시지의 ID 토큰으로 발행하는 요청된 토큰 유형의 URI를 포함합니다.

373 페이지의 『DFHWS-SERVICEURI 컨테이너』

DFHWS-SERVICEURI는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 보안 토큰 서비스(STS)가 AppliesTo 범위로 사용하는 URI를 포함합니다.

372 페이지의 『DFHWS-RESTOKEN 컨테이너』

DFHWS-RESTOKEN은 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 보안 토큰 서비스(STS)로부터의 응답을 포함합니다.

373 페이지의 『DFHWS-STSFault 컨테이너』

DFHWS-STSFault는 DATATYPE(CHAR) 컨테이너입니다. 이 컨테이너는 보안 토큰 서비스(STS)에서 리턴된 오류를 포함합니다.

373 페이지의 『DFHWS-STReason 컨테이너』

DFHWS-STReason은 DATATYPE(CHAR) 컨테이너입니다. 이 요소가 보안 토큰 서비스(STS)의 응답 메시지에 있는 경우 이 컨테이너에는 <wst:Reason> 요소의 콘텐츠가 포함됩니다.

347 페이지의 『DFHERROR 컨테이너』

DFHERROR은 파이프라인 오류에 대한 정보를 다른 메시지 핸들러로 전달하는 데 사용되는 DATATYPE(BIT) 컨테이너입니다.

제 14 장 웹 서비스 지원과 WSRR 간의 상호 운용성

CICS 웹 서비스 지원은 WSRR(IBM WebSphere Service Registry and Repository)과 상호 운용될 수 있습니다. 요청하는 웹 서비스를 보다 빠르게 찾고 제공하는 웹 서비스의 버전 제어를 강제 실행하려면 WSRR을 사용하십시오.

DFHLS2WS 및 DFHWS2LS 모두에는 WSRR과 상호 운용하기 위한 매개변수가 포함됩니다. DFHLS2WS에는 또한 WSRR의 WSDL 문서에 사용자 정의된 고유한 메타데이터를 추가할 수 있는 선택적 매개변수가 포함됩니다.

웹 서비스 지원이 WSRR과 안전하게 통신하도록 하기 위해 보안 소켓 레이어(SSL) 암호화를 사용할 수 있습니다. DFHLS2WS 및 DFHWS2LS에 모두 SSL 암호화 사용을 위한 매개변수가 포함됩니다.

웹 서비스 지원과 WSRR에 SSL을 사용하려면 244 페이지의 『웹 서비스 지원 및 WSRR에서 SSL을 사용하는 방법의 예제』를 참조하십시오.

웹 서비스 지원 및 WSRR에서 SSL을 사용하는 방법의 예제

SSL(Secure Socket Layer) 암호화를 사용하여 웹 서비스 지원과 WSRR(IBM WebSphere Service Registry and Repository) 서버 사이에서 안전하게 상호 운용할 수 있습니다. SSL 암호화를 사용하려면 키 저장소 및 신뢰 저장소가 필요하고 웹 서비스 지원에 대한 특정 매개변수도 지정해야 합니다.

이 태스크 정보

웹 서비스 지원과 WSRR 간 상호작용을 위해 SSL을 사용하려면 다음 단계를 완료하십시오.

프로시저

1. 개인 키 및 공개 키 인증서(PKC)를 위한 키 저장소를 작성하십시오.
 - a. IBM Key Management Utility(iKeyman)와 같은 키 구성 프로그램을 사용하여 키 저장소를 작성할 수 있습니다.
 - b. DFHWS2LS 또는 DFHLS2WS의 **SSL-KEYSTORE** 매개변수에 작성한 키 저장소의 완전한 이름을 지정하십시오.
 - c. 옵션: DFHWS2LS 또는 DFHLS2WS의 **SSL-KEYPWD** 매개변수에 작성한 키 저장소의 비밀번호를 지정하십시오.
2. 모든 신뢰 루트 인증 기관(CA) 인증서를 위한 신뢰 저장소를 작성하십시오. 이러한 인증서는 인바운드 공개 키 인증서의 신뢰를 확립하는 데 사용됩니다.

- a. IBM Key Management Utility(iKeyman)와 같은 키 구성 프로그램을 사용하여 신뢰 저장소를 작성할 수 있습니다.
 - b. DFHWS2LS 또는 DFHLS2WS의 **SSL-TRUSTSTORE** 매개변수에 작성한 신뢰 저장소의 완전한 이름을 지정하십시오.
 - c. 옵션: DFHWS2LS 또는 DFHLS2WS의 **SSL-TRUSTPWD** 매개변수에 작성한 신뢰 저장소의 비밀번호를 지정하십시오.
3. 웹 서비스 지원이 SSL 암호화를 사용하여 WSRР과 통신할 수 있는지 테스트하십시오.
- a. IBM WebSphere Application Server가 제공하는 샘플 파일을 사용하여 WSRР과 웹 서비스 지원을 테스트할 수 있습니다.
 - WebSphere Application Server가 제공하는 샘플 키 저장소는 DummyClientKeyFile.jks 및 DummyServerKeyFile.jks입니다.
 - WebSphere Application Server가 제공하는 샘플 신뢰 저장소는 DummyClientTrustFile.jks 및 DummyServerTrustFile.jks입니다.
 - b. 샘플 키의 키 및 신뢰 저장소 파일을 바꾸십시오. 이러한 키는 WebSphere Application Server와 함께 제공되고 보안을 위해 바뀌어야 합니다.

결과

이제 웹 서비스 지원이 SSL 암호화를 사용하여 네트워크 전체에서 안전하게 WSRР과 통신할 수 있습니다.

제 15 장 SOAP 웹 서비스 문제점 해결

CICS에서 SOAP 웹 서비스 구현 중 수신할 수 있는 문제점은 배치 프로세스 중에 또는 CICS가 SOAP 메시지를 변환하는 중 런타임 시 발생할 수 있습니다.

배치 오류 진단

배치 오류는 CICS 웹 서비스 지원 일괄처리 작업 또는 CICS XML 지원 일괄처리 작업을 실행하거나 CICS에서 PIPELINE 자원을 설치하거나 CICS에서 WEBSERVICE 자원을 설치하려고 할 때 발생할 수 있습니다. 다음은 문제점의 증상, 원인 및 솔루션을 포함하는 가장 일반적인 배치 오류에 대한 설명입니다.

이 태스크 정보

배치 오류가 발생하는 경우 PIPELINE 자원은 일반적으로 DISABLED 상태로 설치되고 WEBSERVICE 자원은 UNUSABLE 상태로 설치됩니다.

CICS 웹 서비스 지원 일괄처리 작업 및 CICS XML 지원 일괄처리 작업과 연관된 정보 및 오류 메시지는 작업 로그에 있습니다. 설치하는 자원과 연관된 오류 메시지는 시스템 로그에 있습니다.

코드 0, 4, 8 또는 12는 지원에 의해 발행되고 다른 코드는 일반적으로 BPXBATCH, JVM 또는 IEBGENER에 의해 발행됩니다.

BPXBATCH에서 발행되는 코드는 두 개의 카테고리로 분류됩니다. 128 미만의 코드는 명령 실패를 나타내고 128 이상의 코드는 프로세스가 신호에 의해 종료되었음을 나타냅니다. BPXBATCH 및 그 리턴 코드에 대한 자세한 정보는 *z/OS UNIX System Services Command Reference*를 참조하십시오.

프로시저

- CICS 웹 서비스 지원 일괄처리 작업 또는 CICS XML 지원 일괄처리 작업을 실행하는 중에 리턴 코드 0, 4, 8 또는 12를 수신합니다. 이러한 리턴 코드는 다음을 의미합니다.
 - 0 - 작업이 성공적으로 완료되었습니다.
 - 4 - 경고. 작업이 성공적으로 완료되었지만 한 개 이상의 경고 메시지가 발행되었습니다.
 - 8 - 입력 오류. 작업이 성공적으로 완료되지 않았습니다. 입력 매개변수를 유효성 검증하는 동안 한 개 이상의 오류 메시지가 발행되었습니다.

- 12 - 오류. 작업이 성공적으로 완료되지 않았습니다. 실행 중 한 개 이상의 오류 메시지가 발행되었습니다.
- 1. 경고 또는 오류 메시지가 있는지 작업 로그를 확인하십시오. 메시지에 대한 세부 설명을 찾아보십시오. 이러한 설명은 일반적으로 문제점을 수정하기 위해 취할 수 있는 조치를 설명합니다.
- 2. 작업의 각 매개변수에 올바른 값을 입력했는지 확인하십시오. 웹 서비스 설명에서 파일 이름 및 요소와 같은 매개변수 값은 대소문자를 구분하여 처리해야 합니다.
- 3. 올바른 매개변수 조합을 지정했는지 확인하십시오. 예를 들어, 서비스 요청자를 위한 웹 서비스 바인딩 파일을 생성할 때 DFHWS2LS에 **PGMNAME** 매개변수를 포함시킬 경우 오류가 발생하고 작업이 성공적으로 완료되지 않습니다.
- CICS 웹 서비스 지원 일괄처리 작업 또는 CICS XML 지원 일괄처리 작업을 실행하는 중에 리턴 코드 1, 136 또는 139를 수신합니다. 이러한 리턴 코드는 일반적으로 사용 가능한 저장영역이 부족하여 JVM이 실패했음을 표시합니다. CICS 지원에 게는 최소 200MB 이상의 JCL 리전 크기가 필요합니다.
 - 1. 리전 크기를 늘리거나 리전 크기를 0M으로 설정하는 것을 고려해 보십시오.
 - 2. 리전 크기를 제한할 수 있는 활성 IEFUSI가 존재하는지 확인하십시오.
- CICS 웹 서비스 지원 일괄처리 작업 DFHLS2WS 또는 CICS XML 지원 일괄처리 작업 DFHLS2SC를 실행하는 중에 리턴 코드 137을 수신합니다. 이 리턴 코드는 작업이 제한시간을 초과했음을 의미합니다.
 - 1. 작업의 EXEC 명령문에서 **TIME** 매개변수를 **TIME=1440**으로 코드화하여 시간을 늘리거나 SYS1.PARMLIB(BPXPRMxx) 멤버의 MAXCPUIME 값을 늘리십시오.
- WEBSERVICE 자원을 설치하려고 시도하는 중에 DFHPI0914 오류 메시지를 수신합니다. 이 메시지에는 설치 실패의 원인에 대한 일부 정보가 포함됩니다.
 - 1. z/OS UNIX에서 웹 서비스 바인딩 파일을 읽을 권한을 CICS에 부여했는지 확인하십시오.
 - 2. 웹 서비스 바인딩 파일이 손상되지 않았는지 확인하십시오. 예를 들어, FTP를 사용하여 2진 모드가 아닌 텍스트 모드로 z/OS UNIX에 파일을 전송하는 경우에 이 오류가 발생할 수 있습니다.
 - 3. 같은 이름을 가진 두 개의 웹 서비스 바인딩 파일이 서로 다른 픽업 디렉토리에 있지 않은지 확인하십시오.
 - 4. 웹 서비스 요청자 애플리케이션에 대한 자원을 설치하려고 하는 경우 SOAP 바인딩의 버전이 파이프라인에서 지원되는 레벨과 일치하는지 확인하십시오. SOAP 1.2를 지원하는 서비스 요청자 파이프라인에 SOAP 1.1 WEBSERVICE를 설치할 수 없습니다.

5. 제공자 모드 WEBSERVICE 자원을 요청자 모드 파이프라인에 설치하고 있지 않은지 확인하십시오. 제공자 모드 웹 서비스 바인딩 파일은 **PROGRAM** 값을 지정합니다. 반면에 요청자 모드 바인딩 파일은 이 값을 지정하지 않습니다.
 6. DFHWS2LS 또는 DFHLS2WS를 사용하는 경우 웹 서비스 바인딩 파일을 생성할 때 올바른 매개변수를 지정했는지 확인하십시오. **PGMNAME**과 같은 일부 매개변수는 웹 서비스 제공자에 대해서만 허용되며 웹 서비스 요청자를 작성하는 경우에는 제외시켜야 합니다.
 7. DFHWS2LS 또는 DFHLS2WS를 사용하는 경우 WEBSERVICE 자원을 작성하기 전에 해결해야 하는 문제점이 있는지 확인하려면 작업에서 발행되는 메시지를 확인하십시오.
- PIPELINE 자원 설치가 실패하고 DFHPI0700, DFHPI0712, DFHPI0714 또는 유사한 오류 메시지를 수신합니다.
 1. DFHPI0700 오류 메시지를 수신한 경우 CICS 리전에서 PL/I 언어 지원을 사용으로 설정해야 합니다. PIPELINE 자원을 설치하려면 먼저 이를 수행해야 합니다. 자세한 정보는 *z/OS용 CICS Transaction Server 설치 안내서*의 내용을 참조하십시오.
 2. z/OS UNIX 디렉토리에 액세스해서 파이프라인 구성 파일을 읽을 수 있는 권한을 CICS에 부여했는지 확인하십시오.
 3. **WSDIR** 매개변수에 지정하는 디렉토리가 유효한지 확인하십시오. 특히, z/OS UNIX의 디렉토리 및 파일 이름은 대소문자를 구분하기 때문에 대소문자를 확인하십시오.
 4. CICS 리전에 ENABLED 상태인 같은 이름의 PIPELINE 자원이 없는지 확인하십시오.
 - PIPELINE 자원은 DISABLED 상태로 설치됩니다. DFHPI0702에서 DFHPI0711 범위의 오류 메시지를 수신합니다.
 1. 파이프라인 구성 파일에 오류가 없는지 확인하십시오. 파이프라인 구성 파일의 요소는 특정 위치에서만 표시될 수 있습니다. 이러한 요소를 잘못 지정하면 DFHPI0702 오류 메시지를 수신합니다. 이 메시지에는 문제점의 원인이 되는 요소의 이름이 포함됩니다. 요소 설명을 확인하여 올바른 위치에서 요소를 코드화했는지 확인하십시오.
 2. 파이프라인 구성 파일에서 탭과 같이 인쇄 불가능한 문자가 없는지 확인하십시오.
 3. XML이 유효한지 확인하십시오. XML이 유효하지 않은 경우 PIPELINE 자원을 설치하려고 할 때 구문 분석 오류가 발생할 수 있습니다.
 4. 파이프라인 구성 파일이 US EBCDIC로 인코딩되는지 확인하십시오. 다른 EBCDIC 인코딩을 사용하려고 시도하는 경우 CICS는 파일을 처리할 수 없습니다.

- WEBSERVICE 자원이 DISABLED 상태입니다. 상태 DISABLED와 DISABLING은 CICS 번들에서 정의되고 설치되는 WEBSERVICE 자원에 대해서만 사용 가능합니다.
1. WEBSERVICE 자원과 연관된 PIPELINE 자원이 버려진 경우 WEBSERVICE 자원은 DISABLED 상태가 됩니다. PIPELINE 자원이 누락된 이유를 조사하고 필요한 경우 이 자원을 대체하십시오.
 2. WEBSERVICE 자원이 정의되는 CICS 번들에 대해 사용 안함 조치가 수행된 경우 웹 서비스를 더 이상 사용하지 않으면 WEBSERVICE 자원이 DISABLED 상태가 됩니다. CICS 번들의 상태를 조사하고 필요한 경우 이 상태를 사용으로 설정하십시오.

서비스 제공자 런타임 오류 진단

제공자 모드 파이프라인에서 인바운드 메시지 수신 및 처리에 문제점이 있는 경우 전송 또는 특정 SOAP 메시지에 문제점이 있을 수 있습니다.

프로시저

- HTTP 또는 WebSphere MQ 전송 오류가 발생했음을 나타내는 DFHPI0401, DFHPI0502 또는 유사한 메시지를 수신합니다. 전송이 HTTP인 경우 클라이언트는 500 서버 내부 오류 메시지를 수신합니다. 전송이 WebSphere MQ인 경우 메시지가 데드 레터 큐(DLQ)에 작성됩니다. SOAP 결함은 웹 서비스 요청자로 리턴되지 않습니다. CICS가 수신된 메시지 유형을 판별할 수 없기 때문입니다.
- DFHxx 메시지 및 404 찾을 수 없음 오류 메시지를 수신합니다.
 1. 웹 서비스 지원을 사용하지 않는 경우 URIMAP 자원을 작성해야 합니다. 웹 서비스 지원을 사용 중인 경우 **PIPELINE SCAN** 명령 실행 시 URIMAP이 자동으로 작성됩니다. 시스템 로그는 이 명령의 실행 결과로 발생한 오류에 대한 정보를 제공합니다.
 2. WEBSERVICE 자원이 사용으로 설정되었는지 및 이 자원과 연관되는 URIMAP이 예상한 것인지 확인하십시오. WEBSERVICE 자원이 UNUSABLE 상태인 경우 703 페이지의 『배치 오류 진단』의 내용을 참조하십시오.
 3. URI 및 포트 번호를 올바르게 지정했는지 확인하십시오. 특히 대소문자를 확인하십시오. URIMAP 자원의 속성 PATH는 대소문자를 구분하기 때문입니다.
- 예기치 않은 오류가 보고되는 경우 웹 서비스 애플리케이션을 디버깅하는 데 CEDX를 사용하는 것을 고려해 보십시오.
 1. CICS에 의해 보고되고 있는 오류 메시지를 확인하려면 시스템 로그를 확인하십시오. 이는 발생 중인 오류의 유형을 표시할 수 있습니다. CICS가 오류를 보고하지 않는 경우 네트워크를 통해 요청이 CICS에 도달하는지 확인하십시오.

2. HTTP 전송의 경우 CPIH에 대해 CEDX를 실행하고 WebSphere MQ 전송의 경우 CPIQ를 실행하십시오. 또는 이와 다를 경우 URIMAP에서 지정한 트랜잭션을 실행하십시오.

애플리케이션 핸들러 전 파이프라인 처리 중 태스크 전환이 발생하면 DFHWS-TRANID 컨테이너가 채워지지 않는 한 새 태스크가 첫 번째 태스크와 같은 트랜잭션 ID로 실행됩니다. 이 경우 CEDX 실행을 방해할 수 있습니다. 첫 번째 태스크가 CEDX 세션에 대한 잠금을 가지고 있기 때문입니다. 태스크가 전환되면 트랜잭션 ID를 변경하는 DFHWS-TRANID를 사용하여 이 문제점을 피할 수 있습니다. 이 경우 파이프라인 및 애플리케이션 태스크 둘 다에 CEDX를 개별적으로 사용할 수 있습니다. CEDX에 대한 자세한 정보는 *CICS* 제공된 트랜잭션의 CEDX 트랜잭션 사용을 참조하십시오.

3. CEDX가 활성화되지 않거나 사용자가 문제점을 해결하도록 허용하는 경우 PI, SO, AP, EI 및 XS 도메인이 활성화된 보조 추적 실행을 고려해 보십시오. 이는 CICS 리전에 보안 문제점, TCP/IP 문제점, 애플리케이션 프로그램 문제점 또는 파이프라인 문제점이 있는지 여부를 표시합니다. 예외 추적점 또는 이상 종료점이 있는지 찾으십시오.

- 변환 오류를 수신하는 경우 713 페이지의 『데이터 변환 오류 진단』의 내용을 참조하십시오.
- 문제점이 MTOM 메시지와 관련이 있다고 판단되면 710 페이지의 『MTOM/XOP 오류 진단』의 내용을 참조하십시오.

- 지속적 HTTP 연결을 주기적으로 닫는 경우:

1. HTTP 연결을 위한 성능 튜닝이 사용으로 설정되어 있는지 확인하십시오. 자세한 정보는 SOTUNING의 내용을 참조하십시오.
2. 성능 튜닝을 사용하는 경우 CICS는 연결이 공유 IP 엔드 포인트를 청취하는 리전 사이에서 분배할 수 있도록 주기적으로 지속적 HTTP 연결을 닫습니다.

- CICS 리전이 최대 용량 상태인 경우 연결 시도가 거부됩니다.

1. HTTP 연결을 위한 성능 튜닝이 사용으로 설정되어 있는지 확인하십시오. 자세한 정보는 SOTUNING의 내용을 참조하십시오.
2. 성능 튜닝이 사용으로 설정되어 있는 경우 CICS가 최대 용량 상태일 때 모든 인바운드 HTTP 연결 열기 요청은 TCP/IP에서 TCPIPService 청취 연결 백로그 큐의 CICS 외부에 대기합니다. 리전의 TCP/IP 글로벌 통계 SOG_PAUSING_HTTP_LISTENING 필드는 현재 케이스인지 여부를 반영하며 SOG_TIMES_AT_ACCEPT_LIMIT/SOG_TIME_LAST_PAUSED_HTTP_LISTENING 필드는 과거 발생에 대한 자세한 정보를 포함합니다.

3. TCPIPService 청취 연결에 대한 ConnectionsDropped 값을 얻으려면 NETSTAT ALL을 사용하십시오. 이는 수신된 후 최대 연결 요청 수가 이미 백로그 큐에 있

어서 삭제된 연결 요청의 수입입니다. 자세한 정보는 z/OS Communication Server IP System Administrator's Commands의 Netstat을 참조하십시오.

4. ConnectionsDropped 값이 너무 높으면 TCPIPService의 백로그 속성 값을 늘리는 것을 고려해 보십시오.

서비스 요청자 런타임 오류 진단

서비스 요청자 애플리케이션에서 웹 서비스 요청을 보내는 데 문제점이 있거나 웹 서비스 제공자로부터 SOAP 결함 메시지를 수신하는 경우 이 섹션을 읽어 보십시오.

이 태스크 정보

발생하는 문제점은 개별 웹 서비스의 오류 또는 전송 레벨에서의 문제로 인한 것일 수 있습니다.

프로시저

- 애플리케이션 프로그램에서 **INVOKE SERVICE** 명령을 사용하는 경우 문제점이 있으면 RESP 및 RESP2 코드가 리턴됩니다.
 1. INVOKE SERVICE 명령에 대한 RESP 및 RESP2 코드의 의미를 찾아보십시오. 문제점이 될 수 있는 사항에 대한 표시를 제공합니다.
 2. 문제점의 원인을 판별하는 데 도움을 줄 수 있는 메시지가 있는지 확인하려면 CICS 시스템 로그를 확인하십시오.
- SOAP 요청 메시지를 전송할 수 없고 파이프라인이 DFHERROR 컨테이너를 리턴하는 경우 파이프라인이 SOAP 메시지를 처리하려고 시도할 때 문제점이 발생한 것입니다.
 1. DFHERROR 컨테이너의 콘텐츠를 확인하십시오. 이 콘텐츠에는 발생한 문제점을 설명하는 오류 메시지 및 일부 데이터가 포함되어 있어야 합니다.
 2. 파이프라인에 새 메시지 핸들러 또는 헤더 처리 프로그램을 추가했습니까? 추가한 경우 새 프로그램 제거 및 웹 서비스 재실행을 시도하여 이러한 작업이 문제점을 해결하는지 확인하십시오. 메시지 핸들러가 파이프라인에 없는 컨테이너를 사용하여 일부 처리를 수행하려고 시도하거나 읽기 전용인 컨테이너를 갱신하려고 시도하는 경우 파이프라인이 처리를 중지하고 DFHERROR 컨테이너에서 오류를 리턴합니다. 헤더 처리 프로그램은 파이프라인에서 제한된 컨테이너 세트만 갱신할 수 있습니다. 세부사항은 342 페이지의 『헤더 처리 프로그램 인터페이스』의 내용을 참조하십시오.
 3. 웹 서비스 요청자 애플리케이션이 웹 서비스 요청을 전송하는 데 **INVOKE SERVICE** 명령을 사용하지 않는 경우 이 애플리케이션이 필요한 모든 제어 컨테이너를 작성했는지 및 이러한 컨테이너가 올바른 데이터 유형인지 확인하십시오. 특히 DFHREQUEST 컨테이너의 데이터 유형이 BIT가 아닌 CHAR인지 확인하십시오.

4. 웹 서비스 요청자 애플리케이션이 **INVOKE SERVICE** 명령을 사용하고 있고 RESP 값 INVREQ 및 RESP2 코드 14가 리턴되는 경우 이는 데이터 변환 오류가 발생했음을 나타냅니다. 713 페이지의 『데이터 변환 오류 진단』의 내용을 참조하십시오.
 5. SOAP 메시지의 XML이 파이프라인 처리 중에 사용자 정의 메시지 핸들러에 의해 무효화되지 않았는지 확인하십시오. CICS는 파이프라인에서 아웃바운드 메시지에 대한 유효성 검증을 수행하지 않습니다. 애플리케이션이 **INVOKE SERVICE** 명령을 사용하는 경우 SOAP 메시지의 본문이 DFHREQUEST 컨테이너에 배치되면 XML이 CICS에 의해 생성되고 잘 구성됩니다. 그러나 SOAP 메시지의 콘텐츠를 변경하는 추가 메시지 핸들러가 있는 경우 이는 파이프라인에서 유효성 검증되지 않습니다.
- SOAP 메시지를 보낼 수 있지만 제한시간 초과 또는 전송 오류를 수신하는 경우 이는 보통 SOAP 결함으로 리턴됩니다. 프로그램이 **INVOKE SERVICE** 명령을 사용하는 경우 CICS는 제한시간 초과 오류에 대해 RESP 값 TIMEDOUT 및 RESP2 코드 2를 리턴하고 전송 오류에 대해 RESP 값 INVREQ 및 RESP2 코드 17을 리턴합니다.
 1. 네트워크 종료점이 있는지 확인하십시오.
 2. PIPELINE 자원의 RESPWAIT 속성이 애플리케이션의 요구사항에 맞게 올바르게 구성되었는지 확인하십시오. 이 RESPWAIT 속성은 애플리케이션으로 돌아가기 전에 CICS가 웹 서비스 제공자의 응답을 대기하는 시간을 정의합니다. 값이 지정되지 않으면 CICS는 HTTP에 대해서는 기본값 10초를 사용하고 WebSphere MQ에 대해서는 60초를 사용합니다. 그러나 CICS에도 각 트랜잭션에 대한 디스패처에 제한시간이 있으며 이 값이 사용 중인 프로토콜의 기본값보다 적으면 CICS는 대신 디스패처 제한시간을 사용합니다.
 - SOAP 메시지를 보낼 수 있지만 웹 서비스 제공자로부터 예상치 못한 SOAP 결함 응답을 수신하는 경우 SOAP 결함의 세부사항은 DFHWS-BODY 컨테이너의 콘텐츠를 확인하십시오.
 1. DFHPIRT 인터페이스를 사용하여 DFHREQUEST에서 완전한 SOAP 엔벨로프를 전송한 경우 아웃바운드 메시지에 중복 SOAP 헤더가 없는지 확인하십시오. 이는 요청자 파이프라인이 SOAP 1.1 또는 SOAP 1.2 메시지 핸들러를 사용하는 경우에 발생할 수 있습니다. 이 SOAP 메시지 핸들러는 서비스 요청자 애플리케이션이 SOAP 엔벨로프에서 이미 SOAP 헤더를 지정했더라도 이를 추가합니다. 이 시나리오에서 다음 중 하나를 수행할 수 있습니다.
 - 파이프라인에서 SOAP 1.1 또는 SOAP 1.2 메시지 핸들러를 제거합니다. 이 경우 이 파이프라인을 사용하는 다른 모든 서비스 요청자 애플리케이션에 영향을 줄 수 있습니다.

- 애플리케이션이 DFHREQUEST에 넣은 SOAP 엔벨로프에서 SOAP 헤더를 제거합니다. CICS는 필요한 SOAP 헤더를 추가합니다. 헤더에 대한 추가 처리를 수행하려는 경우 헤더 처리 프로그램 인터페이스를 사용할 수 있습니다.
- 애플리케이션에서 **WEB SEND** 명령을 대신 사용하고 웹 서비스 지원에서 나오십시오.
- 문제점이 MTOM 메시지 전송 또는 수신과 관련이 있다고 판단되면 『MTOM/XOP 오류 진단』의 내용을 참조하십시오.

MTOM/XOP 오류 진단

MTOM/XOP 오류는 런타임 시 요청자 및 제공자 모드 파이프라인 둘 다에서 발생할 수 있습니다.

시작하기 전에

MTOM/XOP를 지원하기 위한 파이프라인 구성에 문제점이 있으면 703 페이지의 『배치 오류 진단』의 내용을 읽어 보십시오.

이 태스크 정보

프로시저

- MTOM 형식으로 웹 서비스 요청 메시지를 전송할 수 있지만 웹 서비스 제공자로부터 SOAP 결함 메시지를 수신하는 경우 SOAP 결함의 세부사항은 DFHWS-BODY 컨테이너의 콘텐츠를 확인하십시오.
 1. 웹 서비스 제공자가 MIME 다중 파트/관련 메시지를 수신할 수 있습니까? 웹 서비스 제공자가 MTOM 형식을 지원하지 않는 경우 사용자가 다시 수신하는 결함은 구현에 따라 달라질 수 있습니다. 웹 서비스 제공자가 다른 CICS 애플리케이션인 경우 SOAP 결함은 MIME 메시지가 올바른 콘텐츠 유형이 아님을 나타냅니다.
 2. 웹 서비스 제공자가 MIME 메시지를 수신할 수 있는 경우 파이프라인이 메시지를 직접 모드로 전송하는지 또는 호환성 모드로 전송하는지 확인하십시오. 직접 모드로 MTOM 메시지를 전송하는 경우 XML 관련 문제점이 발생할 수 있습니다.
 3. 문제점이 XML과 관련된 문제점인지 확인하려면 웹 서비스에 대해 유효성 검증을 켜십시오. 이는 MTOM 메시지가 파이프라인을 통해 호환성 모드로 처리되는 원인이 됩니다. 이 처리의 일부로 MTOM 핸들러는 메시지 콘텐츠를 구문 분석하여 base64binary 데이터를 최적화합니다. XML에 오류가 있는 경우 CICS는 이 오류를 DFHERROR 컨테이너에 넣고 파이프라인에서 MTOM 전송 실패를 발행합니다.

4. DFHERROR 컨테이너의 콘텐츠를 조사하여 발생한 문제점을 표시하는지 확인하십시오. 이 정보가 문제점의 원인을 진단하는 데 도움이 되기에는 부족한 경우 PI 도메인의 레벨 2 추적을 실행하십시오.
 5. 추적점 PI 0C16을 찾으십시오. 이는 발생한 문제점에 대해 좀 더 세부적으로 설명하며 요청자 애플리케이션에서 제공되는 XML로 문제점을 수정하는 데 도움이 되어야 합니다.
- 예상 2진 첨부 파일이 아웃바운드 MTOM 메시지에서 누락될 경우 2진 데이터가 2진 첨부 파일로 최적화되기에는 너무 작은 것으로 간주됨을 나타낼 수 있습니다. CICS는 파이프라인에서의 데이터 최적화 오버헤드 처리를 정당화하기에 충분히 큰 데이터에 대해서만 2진 첨부 파일을 작성합니다. 크기가 1,500바이트 미만인 2진 데이터는 최적화되지 않습니다.
 - 호환성 모드로 아웃바운드 MTOM 메시지를 전송할 수 없고 파이프라인이 DFHERROR 컨테이너를 리턴하는 경우 파이프라인이 MTOM 메시지를 처리하려고 시도할 때 문제점이 발생한 것입니다.
 1. DFHERROR 컨테이너의 콘텐츠를 확인하십시오. 이 콘텐츠에는 발생한 문제점을 설명하는 오류 메시지 및 일부 데이터가 포함되어 있어야 합니다.
 2. 아웃바운드 MTOM 메시지의 XML이 유효한지 확인하십시오. CICS는 파이프라인에서 아웃바운드 메시지에 대해 유효성 검증을 수행하지 않습니다.
 - DFHPI1100E 메시지를 수신한 경우 CICS에 의해 수신된 MTOM 메시지의 MIME 헤더에 문제점이 있는 것입니다. CICS 메시지에는 발생한 일반 클래스 MIME 오류가 포함됩니다. 발생한 정확한 문제점을 찾으려면 다음을 수행하십시오.
 1. CICS 리전에서 활성인 보조 추적이 있는 경우 예외 추적 항목이 있는지 확인하십시오.
 2. 추적점 PI 1305를 찾으십시오. 이는 오류가 발생한 위치의 컨텍스트를 이해할 수 있도록 MIME 헤더 오류의 네이처, 헤더에서 오류의 위치 및 오류 전 후 최대 80바이트 텍스트를 설명합니다.
- 예를 들어, 다음 추적 발췌는 MIME 콘텐츠 유형 시작 매개변수가 따옴표 안에 묶여 있지 않지만 따옴표 안 문자열 외부에서 유효하지 않은 문자를 포함하고 있기 때문에 올바르지 않음을 표시합니다.

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 79382074 7970653D 22617070 6C696361 74696F6E 2F786F70 * boundary; type="application/xop*
0020 2B786D6C 22382073 74617274 2D696E66 6F3D2261 70706C69 63617469 6F6E2F73 **xml"; start-info="application/s*
0040 6F61702B 786D6C22 3B207374 6172743D 68757273 6C65792E 69626D2E 636F6D3E *oap+xml"; start=
4-0000 3C736F61 70736C75 6E674074 6573742E 68757273 6C65792E 69626D2E 636F6D3E *<soapslung@test.hursley.ibm.com>*
0020 3B206368 61727365 743D7574 662D38 *; charset=utf-8 *
```

- 파이프라인 처리가 인바운드 MTOM 메시지 구문 분석에 실패하고 웹 서비스 요청자는 SOAP 결함 메시지를 수신합니다. 이 메시지는 MTOM 메시지에서 XOP 문서 관련 문제점이 있었음을 나타냅니다. 직접 모드에서 SOAP 결함은 애플리케이션

핸들러에 의해 생성됩니다. 파이프라인이 호환성 모드로 실행되고 있는 경우 SOAP 메시지를 구성할 때 메시지는 MTOM 핸들러에 의해 구문 분석됩니다. 이 경우, CICS는 DFHPI 접두부가 있는 오류 메시지와 SOAP 결함을 발행합니다.

1. DFHPI 접두부가 있는 오류 메시지는 XOP 문서의 문제점을 표시합니다. 예를 들어, 이러한 문제점은 올바르지 않은 MIME 헤더 또는 누락된 2진 첨부 파일이 될 수 있습니다.
2. 문제점의 정확한 원인을 찾으려면 예외 추적점이 있는지 확인하십시오. 특히 PI 13xx로 시작하는 추적점을 찾으십시오. 이러한 추적점은 발생한 예외에 대해 보다 자세히 설명합니다.

또한 PI 레벨 2 추적을 실행하여 오류로 이어지는 이벤트의 시퀀스를 설정할 수 있지만 이 경우 성능에 심각한 영향을 미칠 수 있기 때문에 프로덕션 리전에서는 권장되지 않습니다.

웹 서비스가 MTOM/XOP을 지원하는지 판별

다음 인터페이스를 사용하여 웹 서비스가 MTOM/XOP를 지원하는지 여부를 확인할 수 있습니다.



CICS Explorer



CICS Explorer 관리 보기

웹 서비스 보기의 **XOP Direct Status** 및 **XOP Support Status** 속성을 사용하십시오.

CICSplex SM

WEBSERVICE 정의 보기

CEMT



INQUIRE WEBSERVICE 명령

CICS SPI



INQUIRE WEBSERVICE 명령

PIPELINE 자원의 조작 모드 확인

다음 인터페이스를 사용하여 PIPELINE의 조작 모드를 확인할 수 있습니다.



CICS Explorer

 CICS Explorer 관리 보기

파이프라인 보기의 **Operation mode** 속성을 사용하십시오.

CICSplex SM

PIPELINE 정의 보기

CEMT

 INQUIRE PIPELINE 명령

CICS SPI

 INQUIRE PIPELINE 명령

데이터 변환 오류 진단

데이터 변환 오류는 SOAP 메시지를 CICS COMMAREA 또는 컨테이너로 변환할 때 및 COMMAREA 또는 컨테이너를 SOAP 메시지로 변환할 때 런타임 시 발생할 수 있습니다.

시작하기 전에

증상에는 SOAP 결합 메시지의 생성 및 결합이 발생했음을 나타내는 CICS 메시지가 있습니다.

이 태스크 정보

데이터 변환 문제점이 있을 경우 다음 단계를 수행하십시오.

프로시저

1. WEBSERVICE 자원이 최신인지 확인하십시오. 웹 서비스의 웹 서비스 바인딩 파일을 재생성하고 이 파일을 CICS에 재배포하십시오.
2. CICS에서 사용되거나 생성된 것과 같은 버전의 웹 서비스 문서(WSDL)를 사용하여 원격 웹 서비스가 생성되었는지 확인하십시오.
3. WEBSERVICE 자원이 최신 웹 서비스 바인딩 파일을 사용 중임을 확인하는 경우 다음을 수행하십시오.
 - a. 명령 SET WEBSERVICE(*name*) VALIDATION(여기서, *name*은 WEBSERVICE 자원 이름임)을 사용하여 WEBSERVICE 자원에 대한 런타임 유효성 검증을 사용으로 설정하십시오.

- b. 메시지 로그에서 CICS 메시지 DFHPI1001 또는 DFHPI1002를 확인하십시오. DFHPI1001은 데이터 변환 문제점의 정확한 네이처를 설명하여 사용자가 변환 오류의 소스를 식별하는 데 도움을 줄 수 있습니다. DFHPI1002는 문제점을 찾을 수 없음을 표시합니다.
- c. 웹 서비스에 대한 유효성 검증이 더 이상 필요하지 않은 경우 명령 SET WEBSERVICE(*name*) NOVALIDATION을 사용하여 유효성 검증을 끄십시오.
4. 여전히 변환 오류의 이유를 판별하지 못한 경우에는 실패한 웹 서비스의 CICS 추적을 가져오십시오. 다음 PI 도메인 예외 추적 항목을 찾으십시오.

```
PI 0F39 - PICC *EXC* - CONVERSION_ERROR
PI 0F08 - PIII *EXC* - CONVERSION_ERROR
```

PICC 변환 오류는 인바운드 SOAP 메시지를 COMMAREA 또는 컨테이너로 변환하는 중에 문제점이 발생했음을 나타냅니다. PIII 변환 오류는 애플리케이션 프로그램이 제공한 COMMAREA 또는 컨테이너에서 SOAP 메시지를 생성하는 중에 문제점이 발생했음을 나타냅니다. 두 경우 모두에서 추적 지점은 변환 오류와 연관된 필드의 이름을 식별하고 문제점의 원인이 된 값을 식별할 수도 있습니다. 이러한 추적 지점 중 어느 하나가 발생하면 그 뒤에 변환 오류가 발생하게 됩니다. 이러한 변환 오류에 대한 가능한 해석은 메시지(DFHPI1007에서 DFHPI1010까지)에 대한 설명을 참조하십시오.

데이터 변환 오류가 발생하는 이유

CICS는 잘 구성된 XML을 포함하는지 확인하고 이를 변형하기 위해 필요한 범위에 대한 SOAP 메시지만 유효성 검증합니다. 이는 SOAP 메시지가 WSDL을 사용하여 성공적으로 유효성 검증될 수 있다는 것을 의미하지만, 런타임 환경과 그 반대의 경우에 실패할 수 있다는 것을 의미합니다.

WEBSERVICE 자원은 CICS가 런타임 시 데이터 변환을 수행하도록 매핑 지시사항을 캡슐화합니다. 변환 오류는 WEBSERVICE 자원에 설명된 대로 입력이 예상된 데이터와 일치하지 않을 때 발생합니다.

이 불일치는 다음 이유 중 어떤 것 때문에 발생할 수 있습니다.

- CICS가 수신하는 SOAP 메시지는 WEBSERVICE 자원과 연관되는 웹 서비스 설명(WSDL)에 대해 확인될 때 잘 구성되지 않고 유효하지 않습니다.
- CICS가 수신하는 SOAP 메시지는 잘 구성되고 유효하지만 WEBSERVICE 자원에 대한 범위 밖에 있는 값을 포함합니다.
- COMMAREA 또는 컨테이너의 콘텐츠는 웹 서비스가 생성된 언어 구조 및 WEBSERVICE 자원과 일치하지 않습니다.

예를 들어, WSDL 문서는 필드에서 범위 제한사항(예: 10과 20 사이 값만 가질 수 있는 unsignedInt)을 지정할 수 있습니다. SOAP 메시지에 25의 값이 포함되면 SOAP

메시지를 유효성 검증하면 올바르지 않은 것으로 거부될 수 있습니다. 값 25는 정수의 올바른 값으로 허용되고 애플리케이션으로 전달됩니다.

두 번째 예제는 최대 길이를 지정하지 않고 WSDL 문서가 문자열을 지정하는 장소입니다. DFHWS2LS는 웹 서비스 바인딩 파일을 생성할 때 기본적으로 최대 255자 길이를 추측합니다. SOAP 메시지에 300자가 포함되면 비록 WSDL에 대한 검사가 설정된 최대 길이가 없는 것처럼 메시지의 유효성을 검증한다 할지라도 값이 CICS에 의해 할당되는 255자 버퍼와 맞지 않을 때 메시지를 변환하기 위해 시도할 때 오류가 보고됩니다.

코드 페이지 실행

CICS는 **LOCALCCSID** 시스템 초기화 매개변수의 값을 사용하여 애플리케이션 프로그램 데이터를 인코딩합니다. 그러나, 웹 서비스 바인딩 파일은 US EBCDIC(Cp037)에서 인코딩됩니다. 이는 애플리케이션 프로그램이 사용하는 코드 페이지가 US EBCDIC 코드 페이지에 다르게 문자를 인코딩할 때 데이터를 변환하는 문제점으로 이어질 수 있습니다. 이 문제점을 피하기 위해 웹 서비스 지원 일괄처리 작업에서 **CCSID** 매개변수를 사용하여 애플리케이션 프로그램과 웹 서비스 바인딩 파일 사이에 데이터를 인코딩하기 위한 다른 코드 페이지를 지정할 수 있습니다. 이 매개변수의 값은 해당 특별 WEBSERVICE 자원에 대한 **LOCALCCSID** 시스템 초기화 매개변수를 대체합니다. **CCSID**의 값은 EBCDIC CCSID여야 합니다.

변환 오류에 대한 SOAP 결함 메시지

변환 오류가 런타임 시 발생하고 CICS가 웹 서비스 제공자 역할을 하는 경우 SOAP 결함 메시지가 서비스 요청자에게 발행됩니다. 이 SOAP 결함 메시지에는 CICS에 의해 발행되는 메시지가 포함됩니다.

서비스 요청자는 다음 SOAP 결함 메시지 중 하나를 수신할 수 있습니다.

- SOAP 메시지를 변환할 수 없음

이 결함 메시지는 SOAP 메시지가 올바르지 않게 구성되거나 해당 값이 범위 밖에 있음을 의미합니다.

- 아웃바운드 데이터를 변환할 수 없음

이 결함 메시지는 COMMAREA 또는 컨테이너의 콘텐츠가 일치하지 않음을 의미합니다.

- 조작이 웹 서비스의 일부가 아님

이 결함 메시지는 올바르지 않은 SOAP 메시지를 CICS에서 수신할 때의 특수한 변형입니다.

CICS가 웹 서비스 요청자인 경우 **INVOKE SERVICE** 명령은 RESP 코드로 INVREQ를 리턴하고 RESP2 값으로 14를 리턴합니다.

제 16 장 CICS 카탈로그 관리자 예제 애플리케이션

CICS 카탈로그 관리자 예제 애플리케이션은 CICS 애플리케이션을 외부 클라이언트와 서버에 연결할 때 우수 사례를 표시하도록 설계되는 작업 COBOL 애플리케이션입니다.

예제는 단순한 판매 카탈로그 및 주문 처리 애플리케이션 주위에 구성되며 사용자가 이러한 태스크를 수행할 수 있습니다.

- 카탈로그에 항목을 나열하십시오.
- 카탈로그에서 개별 항목에 대해 조사하십시오.
- 카탈로그에서 항목 주문하십시오.

카탈로그는 VSAM 파일로 구현됩니다.

기본 애플리케이션은 3270 사용자 인터페이스를 가지고 있지만, 구성요소 사이의 잘 정의된 인터페이스를 사용하여 모듈러 구조가 추가적 구성요소를 추가하는 것을 가능하게 합니다. 특히, 애플리케이션에는 웹 서비스 지원이 제공되고, 웹 서비스 환경으로 기존 애플리케이션을 확장할 수 있는 방법을 설명하도록 설계됩니다.

이 예제의 경우, CICS 탐색기는 애플리케이션을 설치하고 배치하는데 사용됩니다. CICS 탐색기는 CICS에 대한 Eclipse-기반 그래픽 도구 인터페이스입니다.

기본 애플리케이션

3270 사용자 인터페이스를 가진 기본 애플리케이션에서는 저장된 카탈로그의 콘텐츠를 나열하고 목록에서 항목을 선택하며 주문할 수량을 입력할 수 있는 기능을 제공합니다. 애플리케이션에는 모듈 디자인이 있고, 이는 웹 서비스와 같은 더 새로운 기술을 지원하기 위해 애플리케이션 확장을 단순하게 합니다.

그림 40에서는 기본 애플리케이션의 구조가 표시됩니다.

그림 40. 기본 애플리케이션의 구조가 표시됩니다.

기본 애플리케이션의 구성요소는 다음과 같습니다.

- 3270 터미널 또는 에뮬레이터를 지원하고 기본 카탈로그 관리자 프로그램과 상호작용하는 BMS 프리젠테이션 관리자(DFH0XGUI).
- 예제 애플리케이션의 코어이고 여러 백엔드 구성요소와 상호작용하는 카탈로그 관리자 프로그램(DFH0XCMN).

- 카탈로그 관리자 프로그램과 데이터 저장소 사이에 인터페이스를 제공하는 데이터 핸들러 프로그램. 기본 애플리케이션에서는 이 프로그램의 두 개의 버전을 제공합니다. VSAM 데이터 세트에 데이터를 저장하는 VSAM 데이터 핸들러 프로그램(DFH0XVDS)과 데이터를 저장하지 않지만 호출자에게 올바른 응답을 리턴하는 더미 데이터 핸들러(DFH0XSDS)가 있습니다. 구성 옵션은 두 개의 프로그램 사이에 선택하도록 합니다.
 - 주문을 디스패치하기 위한 인터페이스를 고객에게 제공하는 디스패치 관리자 프로그램. 다시, 구성 옵션은 이 프로그램의 두 버전 사이에 선택하도록 합니다. DFHX0WOD는 원격 주문 디스패치 엔드포인트를 호출하는 웹 서비스 요청자이고 DFHX0SOD는 호출자에게 올바른 응답을 리턴하는 더미 프로그램입니다.
- 두 개의 동등한 주문 디스패치 엔드포인트가 있습니다. DFH0XODE는 CICS 서비스 제공자 프로그램입니다. DispatchOrderV7.war는 CICS Liberty JVM Server 또는 유사한 환경에 배치될 수 있는 Java 웹 아카이브 파일입니다.
- 호출자에게 올바른 응답을 리턴하지만 기타 조치를 취하지 않는 더미 재고 관리자 프로그램(DFH0XSSM).

BMS 프리젠테이션 관리자

프리젠테이션 관리자는 3270 BMS 패널을 통해 사용자와의 모든 상호작용에 대한 책임이 있습니다. 이 프로그램에서 수행할 비즈니스 의사결정이 없습니다.

BMS 프리젠테이션 관리자는 두 가지 방법으로 사용될 수 있습니다.

- 기본 애플리케이션의 파트로.
- SOAP 메시지를 사용하여 기본 애플리케이션과 통신하는 CICS 웹 서비스 클라이언트로.

데이터 핸들러

데이터 핸들러는 카탈로그 관리자와 데이터 저장소 사이의 인터페이스를 제공합니다.

예제 애플리케이션에서는 두 가지 버전의 데이터 핸들러를 제공합니다.

- 첫 번째 버전에서는 데이터 저장소로 VSAM 파일을 사용합니다.
- 두 번째 버전에서는 문의 시 동일한 데이터를 항상 리턴하지만 갱신 요청 결과는 저장하지 않는 더미 프로그램입니다.

디스패치 관리자

디스패치 관리자는 주문이 확인되었을 때 주문을 고객에게 디스패치하는 것에 책임이 있습니다.

예제 애플리케이션에서는 두 가지 버전의 디스패치 관리자 프로그램을 제공합니다.

- 첫 번째 버전은 호출자에게 정확한 응답을 리턴하지만 기타 조치를 수행하지 않는 더미 프로그램입니다.
- 두 번째 버전은 구성 파일에 정의된 엔드포인트 주소를 요청하는 웹 서비스 요청자 프로그램입니다.

주문 디스패치 프로그램

주문 디스패치 프로그램은 고객에게 항목을 디스패치하는 데 책임이 있는 웹 서비스 제공자 프로그램입니다.

예제 애플리케이션에서 주문 디스패치는 정확한 응답을 호출자에게 리턴하지만 기타 조치는 취하지 않는 더미 프로그램입니다. 이는 예제 웹 서비스의 모든 구성이 작동될 수 있도록 합니다.

재고 관리자

재고 관리자는 재고의 보충을 관리할 책임이 있습니다.

예제 프로그램에서 재고 관리자는 정확한 응답을 호출자에게 리턴하지만 기타 조치는 취하지 않는 더미 프로그램입니다.

애플리케이션 구성

예제 애플리케이션에는 기본 애플리케이션을 구성할 수 있는 프로그램이 들어 있습니다.

기본 애플리케이션 설치 및 설정

기본 애플리케이션을 실행하려면 두 개의 VSAM 데이터 세트를 정의하고 채우며 두 개의 TRANSACTION 자원을 작성해야 합니다.

VSAM 데이터 세트 작성 및 정의

두 개의 KSDS VSAM 데이터 세트는 예제 애플리케이션을 정의하고 채우기 위해 사용됩니다. 하나의 데이터 세트에는 예제 애플리케이션에 대한 구성 정보가 포함됩니다. 다른 데이터 세트에는 영업 카탈로그가 포함됩니다.

프로시저

1. VSAM 데이터 세트를 작성하기 위해 JCL을 찾으십시오. CICS 설치 동안, JCL은 *hlq.SDFHINST* 라이브러리에 위치합니다.
 - DFH\$ECNF 멤버에는 구성 데이터 세트를 생성하기 위한 JCL이 포함됩니다.
 - DFH\$ECAT 멤버에는 카탈로그 데이터 세트를 생성하기 위한 JCL이 포함됩니다.
2. JCL을 수정하고 메소드 서비스 명령에 액세스하십시오.
 - a. 올바른 JOB 카드를 넣으십시오.

- b. 액세스 메소드 서비스 명령의 데이터 세트 이름에 적절한 상위 레벨 규정자를 제공하십시오. 제공된 것처럼, JCL은 HLQ의 상위 레벨 규정자를 사용합니다.

다음 명령은 구성 파일을 정의합니다.

```
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) -  
    )
```

여기서, *hlq*는 사용자 선택사항의 상위 레벨 규정자입니다.

다음 명령은 카탈로그 파일을 정의합니다.

```
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) -  
    )
```

여기서,

- *hlq*는 사용자 선택사항의 상위 레벨 규정자입니다
- *catname*은 사용자 선택사항의 이름입니다. 제공된 것처럼 예제 애플리케이션에서 사용되는 이름은 EXMPCAT입니다.

3. 데이터 세트를 작성하고 채우기 위해 양쪽 작업 모두를 실행하십시오.
4. CICS 탐색기를 사용하여 카탈로그 파일에 대한 FILE 정의를 작성하십시오.
 - a. 정의 > 파일 정의를 선택하십시오. 이름 열을 마우스 오른쪽 단추로 클릭하고 새로 작성을 클릭하여 새 파일 정의를 작성하십시오. EXAMPLE을 자원 그룹 텍스트 상자에 입력하고 EXMPCAT를 이름 텍스트 상자에 입력하십시오. 완료를 클릭하여 FILE 정의를 정의하십시오. 그렇지 않으면, CICS 제공 그룹 DFH\$EXBS에서 파일 정의를 복사할 수 있습니다.
 - b. 새 EXMPCAT 파일을 두 번 클릭하십시오. 파일 정의(EXMPCAT) CICS 예제 애플리케이션 편집기에서 **VSAM** 탭을 선택하십시오. *hlq.EXMPLAPP.EXMPCAT*를 사용할 데이터 세트 이름 텍스트 상자에 입력하십시오.
 - c. 속성 탭을 선택하고 다음 속성의 조작을 예로 설정하십시오.

추가
찾아보기
삭제
읽기
갱신

- d. 기타 모든 속성에 기본값을 사용하십시오.
5. CICS 탐색기를 사용하여 구성 파일에 대한 FILE 정의를 작성하십시오.
- a. 정의 > 파일 정의를 선택하십시오. 이름 열을 마우스 오른쪽 단추로 클릭하고 새로 작성을 클릭하여 새 파일 정의를 작성하십시오. EXAMPLE을 자원 그룹 텍스트 상자에 입력하고 EXMPCONF를 이름 텍스트 상자에 입력하십시오. 완료를 클릭하여 FILE 정의를 정의하십시오. 그렇지 않으면, CICS 제공 그룹 DFH\$EXBS에서 파일 정의를 복사할 수 있습니다.
- b. 새 EXMPCONF 파일을 두 번 클릭하십시오. 파일 정의(EXMPCONF) CICS 예제 애플리케이션 창에서 VSAM 탭을 선택하십시오. hlq.EXMPLAPP.EXMPCONF를 사용할 데이터 세트 이름 텍스트 상자에 입력하십시오.
- c. 속성 탭을 선택하고 다음 속성의 조작을 예로 설정하십시오.

추가
찾아보기
삭제
읽기
갱신

- d. 기타 모든 속성에 기본값을 사용하십시오.

결과

데이터 세트가 채워지고 카탈로그 파일 및 구성 파일에 대한 FILE 정의가 작성되고 설치될 준비가 되었습니다.

3270 인터페이스 정의

애플리케이션을 실행하고 이를 사용자 정의하기 위해 예제 애플리케이션에 3270 사용자 인터페이스가 제공됩니다. 사용자 인터페이스는 두 개의 트랜잭션(EGUI 및 ECFG)으로 구성됩니다. 세 번째 트랜잭션(ECLI)이 CICS 웹 서비스 클라이언트에 사용됩니다.

프로시저

1. CICS 탐색기를 사용하여 다음 트랜잭션을 위한 TRANSACTION 정의를 작성하십시오. 예제 애플리케이션의 올바른 조작은 트랜잭션의 이름에 따라 다르지 않습니다. 따라서, 다른 이름을 사용할 수 있습니다.

- a. 자원 그룹 정의 보기에서 DFH\$EXBS를 마우스 오른쪽 단추로 클릭하여 CICS 제공 그룹 DFH\$EXBS로부터 트랜잭션 EGUI에 대한 정의를 복사하십시오. 새로 작성 > 트랜잭션 정의를 선택하십시오. EGUI를 이름 텍스트 상자에 입력하고 DFH0XGUI를 프로그램 이름 텍스트 상자에 입력하십시오. 완료를 클릭하여 EGUI TRANSACTION 정의를 작성하십시오.
- b. 자원 그룹 정의 창에서 DFH\$EXBS를 마우스 오른쪽 단추로 클릭하여 CICS 제공 그룹 DFH\$EXBS로부터 트랜잭션 ECFG에 대한 정의를 복사하십시오. 새로 작성 > 트랜잭션 정의를 선택하십시오. ECFG를 이름 텍스트 상자에 입력하고 DFH0XCFG를 프로그램 이름 텍스트 상자에 입력하십시오. 완료를 클릭하여 ECFG TRANSACTION 정의를 작성하십시오.
- c. 옵션: 자원 그룹 정의 보기에서 DFH\$EXWS를 마우스 오른쪽 단추로 클릭하여 CICS 제공 그룹 DFH\$EXWS로부터 트랜잭션 ECLI에 대한 정의를 복사하십시오. 새로 작성 > 트랜잭션 정의를 선택하십시오. ECLI를 이름 텍스트 상자에 입력하고 DFH0XCUI를 프로그램 이름 텍스트 상자에 입력하십시오. 완료를 클릭하여 ECLI 트랜잭션 정의를 작성하십시오.

기타 모든 속성에 기본값을 사용하십시오.

2. 옵션: 프로그램 자동 설치를 사용하지 않으려면 CICS 제공 그룹 DFH\$EXBS에서 기본 애플리케이션 프로그램에 대한 PROGRAM 정의를 복사하고 BMS 맵에 대한 MAPSET 정의를 복사하십시오.
 - a. DFH0XS1, DFH0XS2 및 DFH0XS3 멤버에서 BMS 맵에 대한 MAPSET 자원 정의를 복사하십시오. 각 멤버에 있는 세부사항의 경우, 754 페이지의 『기본 애플리케이션의 컴포넌트』의 내용을 참조하십시오.
 - b. 다음 COBOL 프로그램에 대한 PROGRAM 자원 정의를 복사하십시오.

표 21. 기본 애플리케이션에 대한 COBOL 소스를 포함하는 SDFHSAMP 멤버

멤버명	설명
DFH0XCFG	VSAM 구성 파일을 읽고 갱신하기 위해 트랜잭션 ECFG가 호출한 프로그램.
DFH0XCMN	카탈로그 애플리케이션에 대한 제어기 프로그램. 제어기 프로그램을 통해 모든 요청이 전달됩니다.
DFH0XGUI	터미널 사용자로 BMS 맵을 전송하고 터미널 사용자로부터 맵을 수신하는 것을 관리하기 위해 트랜잭션 EGUI가 호출하는 프로그램. 이 프로그램은 프로그램 DFH0XCMN에 링크됩니다.
DFH0XODE	주문 디스패치 웹 서비스에 대한 엔드포인트의 두 가지 버전 중 하나. 이는 CICS에서 실행되는 버전입니다. 이 프로그램은 리턴 COMMAREA에서 "디스패치에서 주문" 텍스트를 설정합니다.
DFH0XSDS	VSAM 카탈로그 파일이 설정되지 않을 때 애플리케이션이 작동하도록 허용하는 데이터 저장소 프로그램의 스텝 또는 더미 버전. DFH0XSDS는 VSAM 파일에 저장된 데이터보다 프로그램에 정의된 데이터를 사용합니다.
DFH0XSOD	주문 디스패치 프로그램의 스텝 버전. COMMAREA의 리턴 코드를 0으로 설정하고 해당 호출자로 리턴합니다. DFH0XSOD는 아웃바운드 웹 서비스가 필요하지 않을 때 사용됩니다.

표 21. 기본 애플리케이션에 대한 COBOL 소스를 포함하는 SDFHSAMP 멤버 (계속)

멤버명	설명
DFH0XSSM	재고 관리자(보충) 프로그램의 스텝 버전. DFH0XSSM은 COMMAREA의 리턴 코드를 0으로 설정하고 해당 호출자로 리턴합니다.
DFH0XVDS	데이터 저장소 프로그램의 VSAM 버전. DFH0XVDS는 카탈로그의 읽기 및 갱신을 수행하기 위해 VSAM 파일에 액세스합니다.
DFH0XWOD	주문 디스패치 프로그램의 웹 서비스 버전. DFH0XWOD는 주문 디스패처에 대한 아웃바운드 웹 서비스를 호출하기 위해 EXEC CICS INVOKE WEBSERVICE를 실행합니다.

기타 모든 속성에 기본값을 사용하십시오.

- c. 옵션: CICS 제공 그룹 DFH\$EXWS로부터 DFH0XCUI에 대한 PROGRAM 정의를 복사하십시오. 기타 모든 속성에 기본값을 사용하십시오. 웹 서비스 클라이언트를 시작하는 트랜잭션 ECLI를 사용하려는 경우 이 프로그램이 필요합니다.

```
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
```

설치 완료

설치를 완료하려면 자원 정의를 포함하는 RDO 그룹을 설치하십시오.

프로시저

자원 그룹 정의 창에서 자원 그룹을 마우스 오른쪽 단추로 클릭하십시오. 설치를 선택하십시오. 사용자의 CICSplex가 정확한지 확인하고 사용자가 대상 리전을 선택하는지 확인한 후 확인을 클릭하십시오.

결과

RDO가 설치되고 애플리케이션이 사용 준비됩니다.

예제 애플리케이션 구성

기본 애플리케이션에 예제 애플리케이션을 구성하기 위해 사용할 수 있는 트랜잭션 (ECFG)이 포함됩니다.

시작하기 전에

구성 트랜잭션은 혼합 문자 정보를 사용합니다. 혼합 문자 정보를 올바르게 처리할 수 있는 터미널을 사용해야 합니다.

이 태스크 정보

예제 애플리케이션의 많은 측면을 지정할 수 있습니다. 다음이 포함됩니다.

- 웹 서비스 사용과 같은 애플리케이션의 전체 구성
- 애플리케이션의 웹 서비스 구성요소가 사용하는 네트워크 주소
- 데이터 저장소에 사용되는 파일과 같은 자원 이름
- 애플리케이션의 각 구성요소에 사용하는 프로그램 이름

구성 트랜잭션을 사용하여 예제 애플리케이션의 CICS 제공 구성요소를 애플리케이션을 재시작하지 않고 사용자 자체 것으로 대체할 수 있습니다.

프로시저

1. 구성 애플리케이션을 시작하려면 트랜잭션 ECFG를 입력하십시오. CICS에는 다음 화면이 표시됩니다.

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:9999
Outbound WebService URI ==> http://myserver:80/exampleApp/dispatchOrder
                        ==>
                        ==>
                        ==>
                        ==>
```

PF

3 END

12 CNCL

2. 필드를 완료하십시오.

데이터 저장소 유형

데이터 저장소 스텝 프로그램을 사용하려면 STUB을 지정하십시오.

VSAM 데이터 저장소 프로그램을 사용하려면 VSAM을 지정하십시오.

아웃바운드 WebService

사용자 주문 디스패치 기능에 원격 웹 서비스를 사용하려면 YES를 지정하십시오. 즉, 카탈로그 관리자 프로그램이 주문 디스패치 웹 서비스 프로그램에 링크 되도록 하려는 경우입니다.

사용자 주문 디스패치 기능에 스텝 프로그램을 사용하려면 NO를 지정하십시오. 즉, 카탈로그 관리자 프로그램이 주문 디스패치 스텝 프로그램에 링크되도록 하려는 경우입니다.

카탈로그 관리자

카탈로그 관리자 프로그램의 이름을 지정하십시오. 예제 애플리케이션과 함께 제공되는 프로그램은 DFH0XCMN입니다.

데이터 저장소 스텝

데이터 저장소 유형 필드에 STUB을 지정한 경우, 데이터 저장소 스텝 프로그램의 이름을 지정하십시오. 예제 애플리케이션과 함께 제공되는 프로그램은 DFH0XSDS입니다.

데이터 저장소 VSAM

데이터 저장소 유형 필드에 VSAM을 지정한 경우, VSAM 데이터 저장소 프로그램의 이름을 지정하십시오. 예제 애플리케이션과 함께 제공되는 프로그램은 DFH0XVDS입니다.

주문 디스패치 스텝

아웃바운드 WebService 필드에 NO를 지정한 경우, 주문 디스패치 스텝 프로그램의 이름을 지정하십시오. 예제 애플리케이션과 함께 제공되는 프로그램은 DFH0XSOD입니다.

주문 디스패치 WebService

아웃바운드 WebService 필드에서 YES를 지정한 경우, 서비스 요청자로 기능하는 프로그램의 이름을 지정하십시오. 예제 애플리케이션과 함께 제공되는 프로그램은 DFH0XWOD입니다.

재고 관리자

재고 관리자 프로그램의 이름을 지정하십시오. 예제 애플리케이션과 함께 제공되는 프로그램은 DFH0XSSM입니다.

VSAM 파일 이름

데이터 저장소 유형 필드에 VSAM을 지정한 경우, CICS FILE 정의의 이름을 지정하십시오. 제공된 것처럼 예제 애플리케이션에서 사용되는 이름은 EXMPCAT입니다.

서버 주소 및 포트

CICS 웹 서비스 클라이언트를 사용 중인 경우, 예제 애플리케이션이 웹 서비스로 배치되는 시스템의 IP 주소 및 포트를 지정하십시오.

아웃바운드 WebService URI

아웃바운드 WebService 필드에 YES를 지정한 경우, 디스패치 주문 기능을 구현하는 웹 서비스의 위치를 지정하십시오. 제공된 CICS 엔드포인트를 사용 중

인 경우, 아웃바운드 **WebService**를 `http://myserver:myport/exampleApp/dispatchOrder`로 설정하십시오. 여기서, `myserver` 및 `myport`는 CICS 서버 주소 및 포트입니다.

BMS 인터페이스로 예제 애플리케이션 실행

기본 애플리케이션은 BMS 인터페이스를 사용하여 실행될 수 있습니다.

프로시저

1. CICS 터미널로부터 트랜잭션 EGUI를 입력하십시오. 예제 애플리케이션 메뉴가 표시됩니다.

```
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
```

카탈로그에 항목을 나열하고 항목을 주문하거나 메뉴에서 옵션을 사용하여 애플리케이션을 종료할 수 있습니다.

2. 1을 입력하고 Enter를 눌러 항목 나열 옵션을 선택하십시오. 애플리케이션은 카탈로그에 항목의 목록을 표시합니다.

CICS EXAMPLE CATALOG APPLICATION - Inquire Catalog

Select a single item to order with /, then press ENTER

Item	Description	Cost	Order
0010	Ball Pens Black 24pk	2.90	/
0020	Ball Pens Blue 24pk	2.90	-
0030	Ball Pens Red 24pk	2.90	-
0040	Ball Pens Green 24pk	2.90	-
0050	Pencil with eraser 12pk	1.78	-
0060	Highlighters Assorted 5pk	3.89	-
0070	Laser Paper 28-lb 108 Bright 500/ream	7.44	-
0080	Laser Paper 28-lb 108 Bright 2500/case	33.54	-
0090	Blue Laser Paper 201b 500/ream	5.35	-
0100	Green Laser Paper 201b 500/ream	5.35	-
0110	IBM Network Printer 24 - Toner cart	169.56	-
0120	Standard Diary: Week to view 8 1/4x5 3/4	25.99	-
0130	Wall Planner: Eraseable 36x24	18.85	-
0140	70 Sheet Hard Back wire bound notepad	5.89	-
0150	Sticky Notes 3x3 Assorted Colors 5pk	5.35	-

F3=EXIT F7=BACK F8=FORWARD F12=CANCEL

3. /를 주문 열에 입력하고 Enter를 눌러 항목을 주문하십시오. 애플리케이션에는 주문할 항목의 세부사항이 표시됩니다.

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. 주문을 이행하기 위해 충분한 재고가 있으면, 다음 정보를 입력하십시오.
- 주문 수량 필드를 완료하십시오. 주문하려는 항목의 수를 지정하십시오.
 - User Name** 필드를 완료하십시오. 1-8자 문자열을 입력하십시오. 기본 애플리케이션은 여기에 입력한 값을 확인하지 않습니다.
 - 담당 부서 필드를 완료하십시오. 1-8자 문자열을 입력하십시오. 기본 애플리케이션은 여기에 입력한 값을 확인하지 않습니다.
5. Enter를 눌러 주문을 제출하고 기본 메뉴로 돌아가십시오.

6. F3을 눌러 애플리케이션을 종료하십시오.

예제 애플리케이션의 웹 서비스 지원

웹 서비스 지원은 주문 디스패처 구성요소에 대한 웹 서비스 엔드포인트의 두 가지 버전 및 웹 클라이언트 프론트 엔드 두 가지 버전을 제공하여 예제 애플리케이션을 확장합니다.

웹 클라이언트 프론트 엔드의 두 가지 버전 및 웹 서비스 엔드포인트의 한 가지 버전은 다음 환경에서 실행되는 Java 웹 아카이브 파일(WAR)로 제공됩니다.

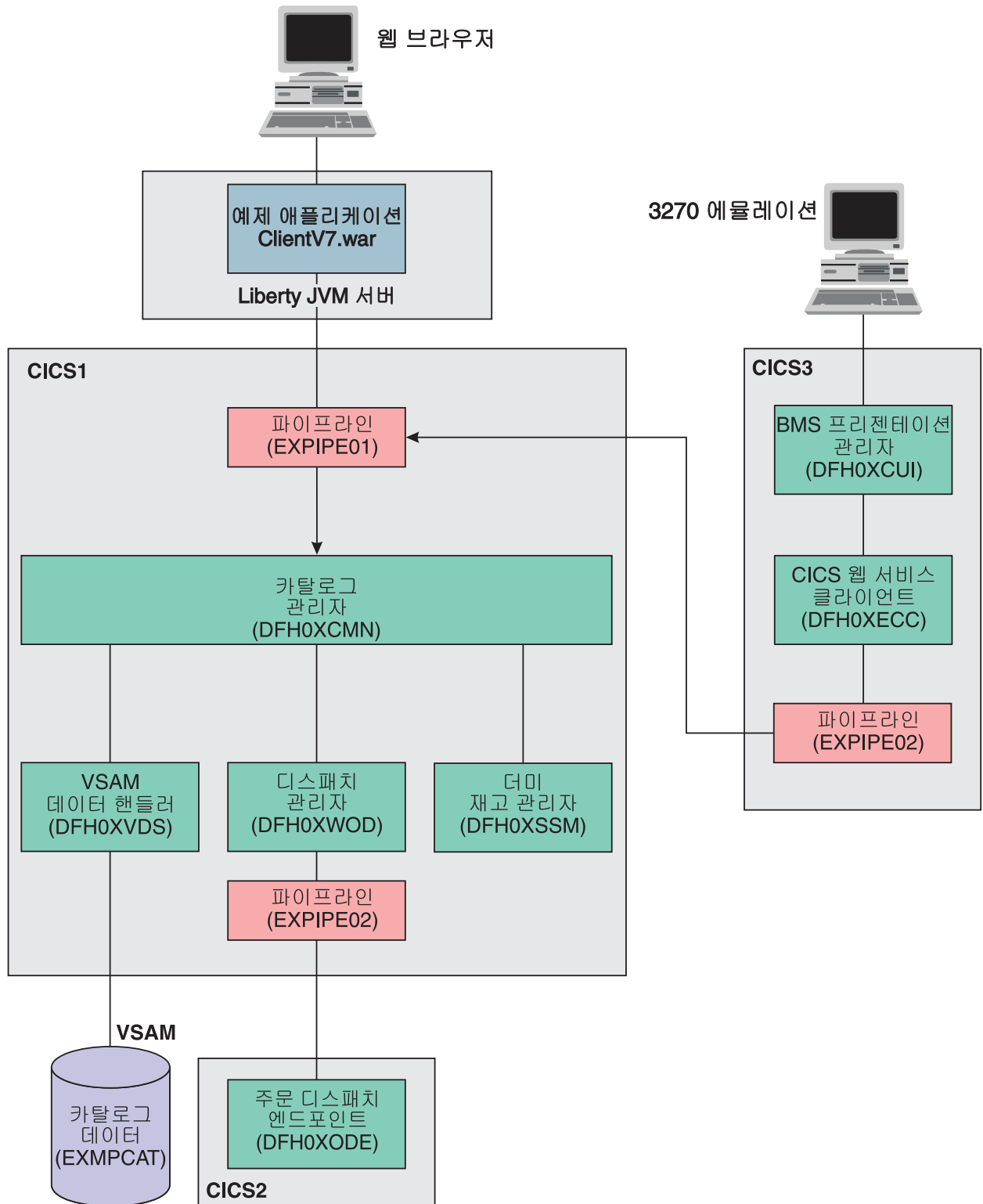
환경	WAR 파일
CICS Liberty JVM 서버 V8.5.5	ExampleAppClientV7.war ExampleAppWrapperClientV7.war DispatchOrderV7.war

해당 WAR는 동적 웹 프로젝트에서 내보냈습니다. WAR 파일 배치는 다음을 참조하십시오.

JVM 서버 구성 파일(예: `server.xml`.)에 추가하여 (예: `</feature> jaxws-2.2 </feature>`) Liberty JVM 서버에서 `jax-ws` 기능을 활성화해야 합니다.

웹 서비스 엔드포인트의 두 번째 버전은 CICS 서비스 제공자 애플리케이션 프로그램 (DFH0XODE)으로 제공됩니다.

그림 1에서는 주문 디스패치 웹 서비스 엔드 포인트로 웹 클라이언트 프론트 엔드의 한 가지 버전과 CICS 서비스 제공자를 사용하여 예제 애플리케이션의 구성을 보여줍니다. 또한 CICS 시스템에는 웹 서비스 클라이언트도 포함되어 있습니다.

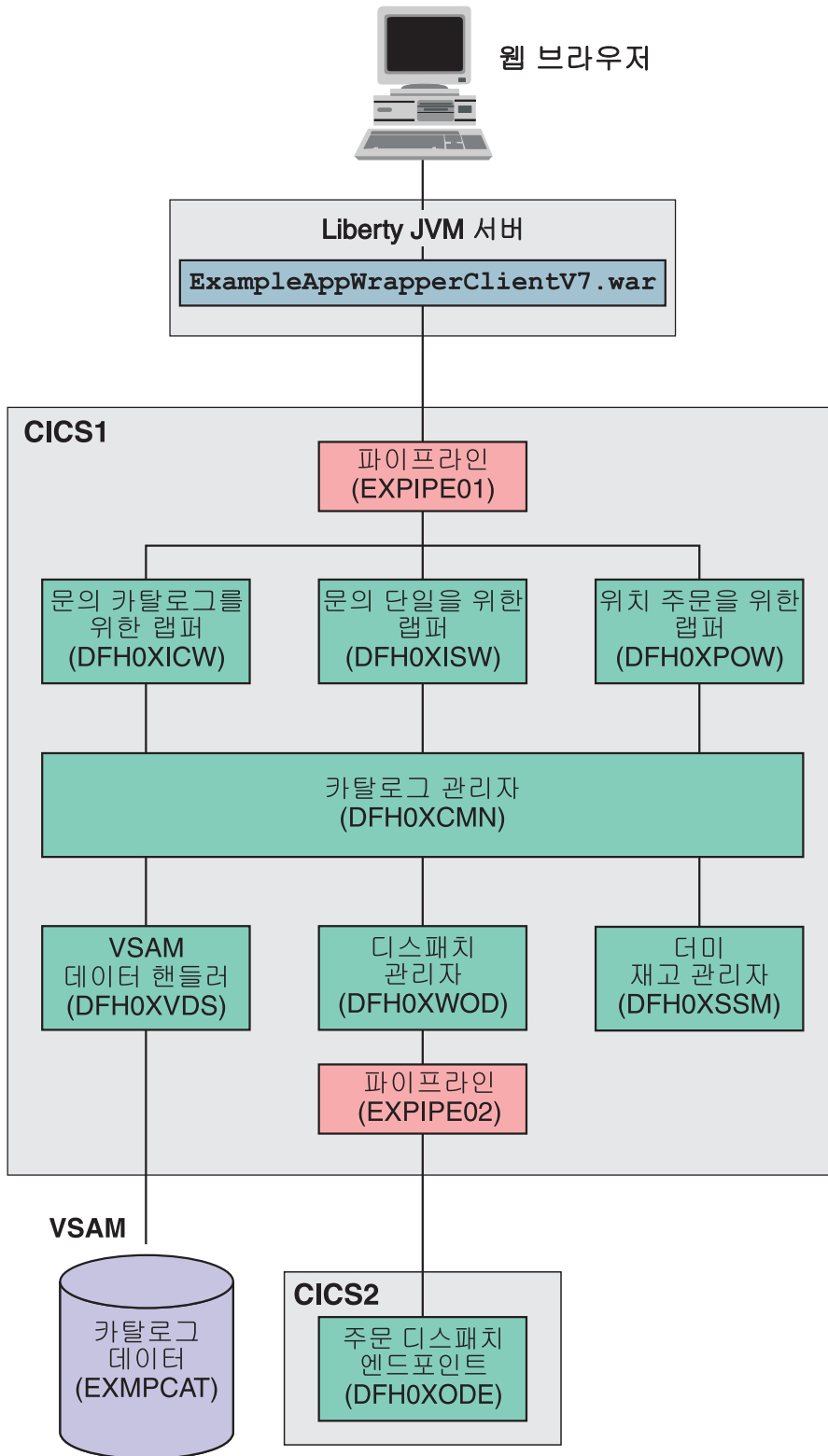


이 구성에서 애플리케이션은 두 개의 다른 클라이언트를 통해서 액세스됩니다.

- ExampleAppClientV7.War가 배치되는 Liberty JVM 서버에 연결된 웹 브라우저 클라이언트

- CICS 웹 서비스 클라이언트 DFH0XECC. 이 클라이언트는 기본 애플리케이션과 동일한 BMS 프리젠테이션 로직을 사용하지만 DFH0XGUI 대신 모듈 DFH0XCUI를 사용합니다.

그림 2에서는 주문 디스패치 웹 서비스 엔드 포인트로 CICS 서비스 제공자와 웹 클라이언트 프론트 엔드의 다른 버전을 보여줍니다.



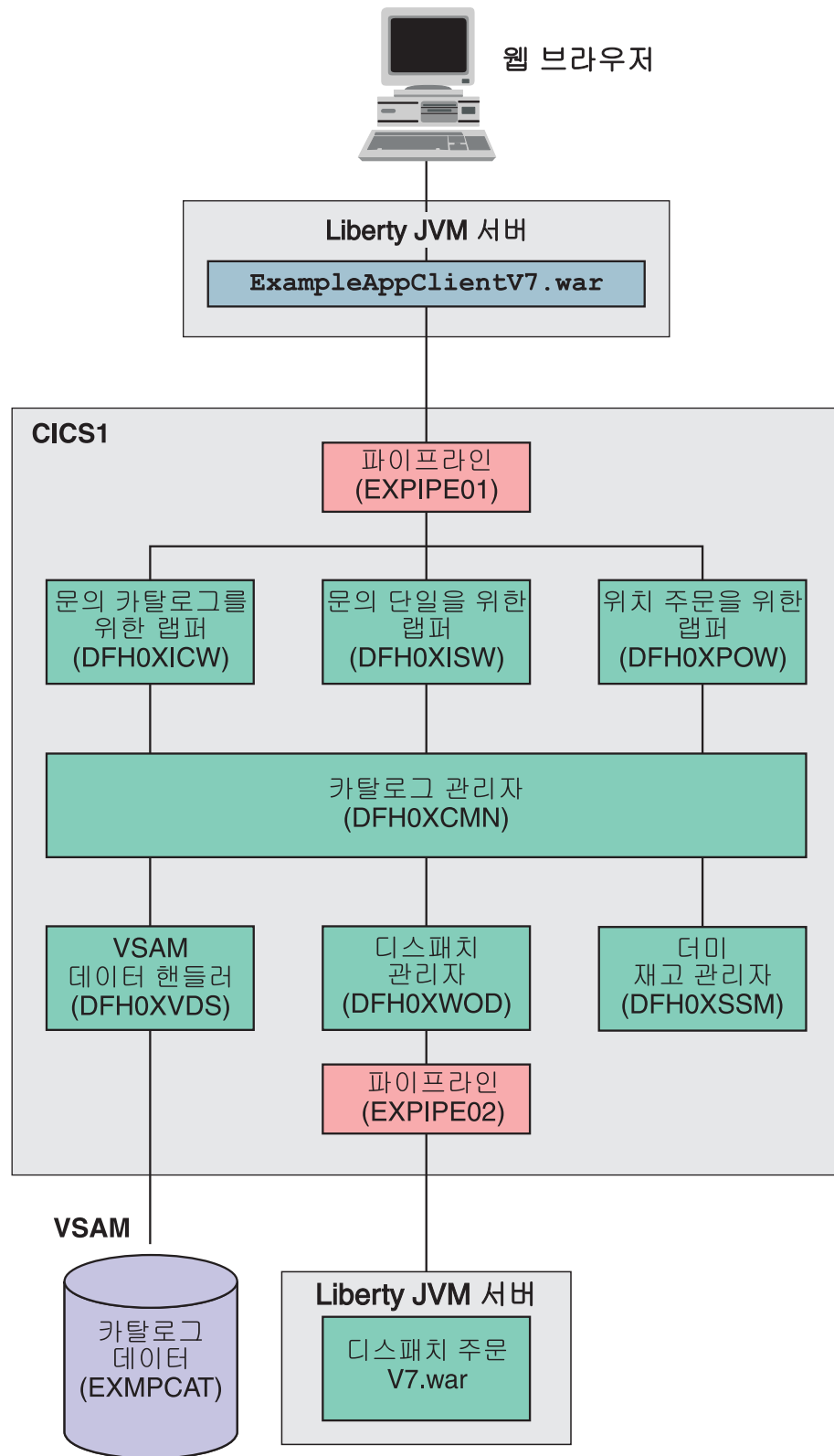
이 구성에서 웹 브라우저 클라이언트가 Liberty JVM 서버에 연결되고 여기에 ExampleAppWrapperClientV7.war가 배치됩니다. CICS에서 세 개의 랩퍼 애플리케이션

이션(문의 카탈로그, 문의 단일 및 위치 주문 기능)이 서비스 제공자 애플리케이션으로 배치됩니다. 이는 차례로 기본 애플리케이션에 링크됩니다.

CICS 시스템에서 디스패치 관리자가 이 엔드 포인트를 호출하려면 다음 구성을 변경해야 합니다.

- 아웃바운드 웹 서비스입니까? 예
- 디스패치 주문 엔드 포인트가 배치된 URL에 대한 아웃바운드 웹 서비스 URL(예: `http://cics2:8080/exampleApp/dispatchOrder`)

그림 3에서는 Liberty JVM 서버의 웹 클라이언트 프론트 엔드와 주문 디스패치 웹 서비스 엔드 포인트 모두로 예제 애플리케이션의 구성을 보여줍니다.



이 구성에서 웹 브라우저 클라이언트가 Liberty JVM 서버에 연결되고 여기에 ExampleAppClientV7.war가 배치됩니다. 주문 디스패치 웹 서비스 엔드 포인트 DispatcherV7.war는 Liberty JVM 서버에 설치되어 있습니다.

CICS 시스템에서 디스패치 관리자가 이 엔드 포인트를 호출하려면 다음 구성을 변경해야 합니다. 구성 방법에 대한 예제 애플리케이션 구성을 참조하십시오.

- Outbound WebService? 예
- 디스패치 주문 엔드 포인트가 배치된 URL에 대한 아웃바운드 웹 서비스 URL(예: `http://mylibertyserver:9080/DispatcherV7/Dispatcher`)

구성 방법에 대한 예제 애플리케이션 구성은 의 내용을 참조하십시오.

코드 페이지 지원 구성

제공한 바와 같이, 예제 애플리케이션은 두 개의 CCS(Coded Character Set)를 사용합니다. 두 문자 세트 간에 데이터 변환이 가능하게 시스템을 구성해야 합니다.

이 태스크 정보

예제 애플리케이션에서 사용하는 CCS(Coded Character Set)는 다음과 같습니다.

037 EBCDIC 그룹 1: 미국, 캐나다(z/OS), 네덜란드, 포르투갈, 브라질, 호주, 뉴질랜드

1208 UTF-8 레벨 3

프로시저

z/OS 시스템의 변환 이미지에 다음 명령문을 추가하십시오.

```
CONVERSION 037,1208;  
CONVERSION 1208,037;
```

자세한 정보는 z/OS용 CICS Transaction Server 설치 안내서의 내용을 참조하십시오.

웹 서비스 클라이언트 및 랩퍼 프로그램 정의

프로그램 자동 설치를 사용 중이지 않은 경우, 웹 서비스 클라이언트 및 랩퍼 프로그램에 대한 자원 정의를 정의해야 합니다.

프로시저

1. 정의 > 프로그램 정의를 선택하여 랩퍼 프로그램에 대한 PROGRAM 자원 정의를 정의하기 위해 CICS 탐색기를 사용하십시오. 프로그램 정의 보기에서 마우스 오른쪽 단추를 클릭하고 새로 작성을 선택하여 새 프로그램 정의를 작성하십시오.

DFH0XECC를 자원 그룹 텍스트 상자에 입력하고 PROGRAM1을 이름 텍스트 상자에 입력하십시오. 완료를 클릭하여 프로그램 정의를 정의하십시오. 다음 COBOL 프로그램에 대한 정의를 작성하십시오.

표 22. 랩퍼 모듈에 대한 COBOL 소스 코드를 포함하는 SDFHSAMP 멤버

멤버명	설명
DFH0XICW	inquireCatalog 서비스에 대한 랩퍼 프로그램.
DFH0XISW	inquireSingle 서비스에 대한 랩퍼 프로그램.
DFH0XPOW	purchaseOrder 서비스에 대한 랩퍼 프로그램.

- 정의 > 프로그램 정의를 선택하여 웹 서비스 클라이언트 프로그램 DFH0XECC에 대한 PROGRAM 자원 정의를 정의하기 위해 CICS 탐색기를 사용하십시오. 프로그램 정의 보기에서 마우스 오른쪽 단추를 클릭하고 새로 작성을 선택하여 새 프로그램 정의를 작성하십시오. DFH0XECC를 자원 그룹 텍스트 상자에 입력하고 PROGRAM1을 이름 텍스트 상자에 입력하십시오. 완료를 클릭하여 프로그램 정의를 정의하십시오. 웹 서비스 클라이언트 프로그램 DFH0XECC는 COBOL 프로그램입니다. 기타 모든 속성에 대해 기본값을 사용할 수 있습니다.

웹 서비스 자원 설치

예제 애플리케이션에 대한 웹 서비스 지원을 실행할 수 있으려면 두 개의 z/OS UNIX 디렉토리를 작성하고 필수 CICS 자원을 작성해야 합니다.

z/OS UNIX 디렉토리

예제 애플리케이션에 대한 웹 서비스 지원은 z/OS UNIX에서 셸프 디렉토리와 픽업 디렉토리가 필요합니다.

셸프 디렉토리는 WEBSERVICE 자원과 연관된 WEBSERVICE 바인딩 파일을 저장 하는데 사용됩니다. 각 WEBSERVICE 자원은 차례로 PIPELINE과 연관됩니다. 셸프 디렉토리는 PIPELINE 자원이 관리하며 콘텐츠를 직접 수정하면 안됩니다. CICS가 각 PIPELINE에 대한 셸프 디렉토리의 밑에 고유 디렉토리 구조를 확인하는 것처럼, 여러 PIPELINE은 동일한 셸프 디렉토리를 사용할 수 있습니다.

픽업 디렉토리는 PIPELINE과 연관된 WEBSERVICE 바인딩 파일을 포함하는 디렉토리입니다. PIPELINE을 설치하거나 **PERFORM PIPELINE SCAN** 명령에 응답하면 바인딩 파일에 있는 정보를 사용하여 PIPELINE과 연관된 WEBSERVICE 및 URIMAP 정의를 동적으로 작성합니다.

예제 애플리케이션에서는 셸프 디렉토리에 /var/cicsts를 사용합니다.

파이프라인 정의 작성

파이프라인의 완전한 정의는 PIPELINE 자원 및 PIPELINE 구성 파일로 구성됩니다. 해당 파일에는 파이프라인을 통해 전달되는 것처럼 웹 서비스 요청 및 응답에서 실행되는 메시지 핸들러의 세부사항이 포함됩니다.

이 태스크 정보

예제 애플리케이션은 제공된 SOAP 1.1 핸들러를 사용하여 인바운드 및 아웃바운드 요청의 SOAP 엔벨로프로 처리합니다. CICS에서는 서비스 제공자 및 서비스 요청자에서 사용할 수 있는 샘플 파이프라인 구성 파일을 제공합니다.

둘 이상의 웹 서비스는 단일 파이프라인을 공유할 수 있습니다. 따라서, 예제 애플리케이션의 인바운드 요청에 대한 하나의 파이프라인만 정의해야 합니다. 그러나, 단일 파이프라인이 동시에 제공자와 요청자 파이프라인이 되도록 구성될 수 없으므로 아웃바운드 요청에 대한 두 번째 파이프라인을 정의해야 합니다.

Java 기반 파이프라인을 사용하려면 1b 단계에서 `basicsoap11provider.xml` 대신 샘플 서비스 제공자 구성 파일 `basicsoap11javaprovider.xml`을 지정해야 합니다. 그리고, 2b 단계에서 `basicsoap11requester.xml` 대신 샘플 서비스 요청자 구성 파일 `basicsoap11javarequester.xml`을 지정하십시오. 샘플 구성 파일에 대한 자세한 정보는 파이프라인 구성 파일을 참조하십시오. Java 기반 파이프라인에서 Axis2 애플리케이션 핸들러를 사용하려면, 1a 단계에서 EXPIPE01을 EXPIPE03으로 대체하고 2a 단계에서 EXPIPE02를 EXPIPE04로 대체해야 합니다.

프로시저

1. CICS 탐색기를 사용하여 서비스 제공자에 대한 파이프라인 정의를 작성하십시오.
 - a. 정의 > 파이프라인 정의를 선택하여 CICS 탐색기를 사용하는 랩퍼 프로그램에 대한 PIPELINE 정의를 작성하십시오. 파이프라인 정의 보기에서 마우스 오른쪽 쪽 단추를 클릭하고 새로 작성을 선택하여 새 파이프라인 정의를 작성하십시오. DFH\$EXWS를 자원 그룹 텍스트 상자에 입력하고 EXPIPE01을 이름 텍스트 상자에 입력하십시오. 완료를 클릭하여 PIPELINE 정의를 작성하십시오. 그렇지 않으면, CICS 제공 그룹 DFH\$EXWS에서 PIPELINE 정의를 복사할 수 있습니다. 자원 그룹 정의 보기에서 DFH\$EXWS를 마우스 오른쪽 단추로 클릭하고 새로 작성 > 파이프라인 정의를 선택하십시오.
 - b. PIPELINE 정의를 두 번 클릭하고 파이프라인 정의(EXPIPE01) 편집기에서 속성 탭을 선택하십시오. 세부사항에서 구성 파일은 샘플 파일 `/usr/lpp/cicsts/samples/pipelines/basicsoap11provider.xml`의 위치로 설정되어야 합니다. 여기서, `/usr/lpp/cicsts`는 디렉토리의 파일에 대한 경로이고, 셀프는 `/var/cicsts`여야 하고, 상태는 `ENABLED`여야 하며, **WS** 디렉토리는 `/usr/lpp/cicsts/samples/webservices/wsbind/provider`여야 합니다.

z/OS UNIX 항목은 대소문자 구분이고 `/usr/lpp/cicsts`의 기본 CICS z/OS UNIX 설치 루트를 추정합니다.
2. CICS 탐색기를 사용하여 서비스 요청자에 대한 PIPELINE 정의를 작성하십시오.
 - a. 정의 > 파이프라인 정의를 선택하여 CICS 탐색기를 사용하는 랩퍼 프로그램에 대한 PIPELINE 정의를 작성하십시오. 파이프라인 정의 보기에서 마우스 오른쪽

쪽 단추를 클릭하고 새로 작성을 선택하여 새 파이프라인 정의를 작성하십시오. DFH\$EXWS를 자원 그룹 텍스트 상자에 입력하고 EXPIPE02를 이름 텍스트 상자에 입력하십시오. 완료를 클릭하여 PIPELINE 정의를 작성하십시오. 그렇지 않으면, CICS 제공 그룹 DFH\$EXWS에서 PIPELINE 정의를 복사할 수 있습니다.

- b. PIPELINE 정의를 두 번 클릭하고 파이프라인 정의(EXPIPE02) 편집기에서 속성 탭을 선택하십시오. **Details**에서 구성 파일은 샘플 파일 `/usr/lpp/cicsts/samples/pipelines/basicsoap11requester.xml`의 위치로 설정되어야 합니다. 여기서, `/usr/lpp/cicsts`는 디렉토리의 파일에 대한 경로이고, 셀프는 `/var/cicsts/`여야 하고, 상태는 **ENABLED**이며, **WS** 디렉토리는 `/usr/lpp/cicsts/samples/webservices/wsbind/requester/`여야 합니다.

TCP/IP 서비스 작성

클라이언트가 HTTP 전송을 통해 웹 서비스에 연결되기 때문에 인바운드 HTTP 트래픽을 수신하기 위해 TCP/IP 서비스를 정의해야 합니다.

프로시저

CICS 탐색기를 사용하여 인바운드 HTTP 요청을 처리하도록 TCPIPSERVICE 정의를 작성하십시오.

1. 정의 > **TCP/IP** 서비스 정의를 선택하여 TCPIPSERVICE 정의를 작성하십시오. TCP/IP 서비스 정의 보기를 마우스 오른쪽 단추로 클릭하고 새로 작성을 선택하여 새 정의를 작성하십시오. DFH\$EXWS를 자원 그룹 텍스트 상자에 입력하고 EXMPPORT를 이름 텍스트 상자에 입력하십시오. 포트 번호를 지정해야 합니다. CICS 시스템에서 사용하지 않은 포트 번호를 입력하십시오. 완료를 클릭하여 TCPIPSERVICE 정의를 작성하십시오.
2. TCPIPSERVICE 정의를 두 번 클릭하십시오. TCP/IP 서비스 정의(EXMPPORT) 편집기의 속성 탭에서 다음 속성을 설정하십시오.

Urm은 DFHWBAAX여야 합니다

프로토콜은 HTTP여야 합니다

트랜잭션은 CWXN이어야 합니다

3. 기타 모든 속성에 기본값을 사용하십시오.

WEBSERVICE 및 URIMAP 자원 동적으로 설치

웹 서비스로 노출되는 각 기능에는 프로그램의 COMMAREA 인터페이스 및 SOAP BODY의 수신 XML 사이에 매핑하기 위한 WEBSERVICE 자원이 필요하고, 올바른 파이프라인 및 웹 서비스에 수신 요청을 라우팅하는 URIMAP 자원이 필요합니다. 자원 정의 온라인(RDO)를 사용하여 WEBSERVICE 및 URIMAP 자원을 정의하고 설치할 수 있다고 해도 파이프라인 자원을 설치할 때 CICS가 이를 동적으로 작성할 수도 있습니다.

프로시저

1. CICS 탐색기를 사용하여 PIPELINE 자원을 설치하십시오.

- a. 정의 > 파이프라인 정의를 선택하십시오. 파이프라인 정의 보기에서 EXPIPE01 PIPELINE 정의를 마우스 오른쪽 단추로 클릭하고 설치를 선택하십시오. 선택란을 선택하여 대상 CICS 리전을 선택하십시오. 확인을 클릭하여 PIPELINE을 설치하십시오.

참고: 735 페이지의 『파이프라인 정의 작성』에서 Java 기반 파이프라인 정의를 작성한 경우, Pipeline Definitions 보기에서 EXPIPE03 PIPELINE 정의를 마우스 오른쪽 단추로 클릭하십시오.

- b. Java 기반 파이프라인에 대한 EXPIPE02 PIPELINE 정의 또는 EXPIPE04에 대해 이 프로세스를 반복하십시오.

각 PIPELINE 자원을 설치할 때, CICS는 PIPELINE WSDIR 속성에 지정된 디렉토리(픽업 디렉토리)를 스캔합니다. 디렉토리의 각 WEBSERVICE 바인딩 파일의 경우, 이는 .wsbind 접두어를 가진 각 파일에 대한 것이고, CICS는 이러한 자원이 존재하지 않으면 WEBSERVICE 자원 및 하나의 URIMAP 자원을 설치합니다.

URIMAP 자원은 CICS에 WEBSERVICE 자원과 특정 URI를 연관시키기 위한 정보를 제공합니다. 바인딩 파일의 정보가 기존 자원 보다 더 새로운 것이면 기존 자원이 대체됩니다.

WSDL 파일 또는 WSDL 아카이브 파일이 픽업 디렉토리로 복사되면 두 번째 선택적 URIMAP 자원이 설치됩니다. 이 URIMAP 자원에서는 외부 요청자가 URI를 사용하여 WSDL 아카이브 파일 또는 WSDL 문서를 감지할 수 있도록 WSDL 아카이브 파일 또는 WSDL 문서를 특정 URI와 연관시키기 위해 CICS에 정보를 제공합니다.

PIPELINE이 나중에 사용 안함으로 설정되고 제거될 때, 연관된 모든 WEBSERVICE 및 URIMAP 자원도 제거됩니다.

2. PIPELINE 자원을 이미 설치한 경우, **PERFORM PIPELINE SCAN** 명령을 사용하여 PIPELINE 픽업 디렉토리의 스캔을 시작하십시오.

PIPELINE 자원을 설치할 때, 제공자 파이프라인에 대한 다음 WEBSERVICE 자원 및 연관된 URIMAP 자원이 시스템에 설치됩니다.

```
dispatchOrder
dispatchOrderEndpoint
inquireCatalog
inquireCatalogClient
```

inquireCatalogWrapper
inquireSingle
inquireSingleClient
inquireSingleWrapper
placeOrder
placeOrderClient
placeOrderWrapper

WEBSERVICE 자원의 이름이 WEBSERVICE 바인딩 파일의 이름에서 파생됩니다. URIMAP 자원의 이름이 동적으로 생성됩니다. 추가 URIMAP은 파이프라인의 픽업 디렉토리에 있는 각 WSDL 문서를 위해 생성됩니다. 웹 서비스 보기를 열려면 **Operations** > 웹 서비스를 선택하여 자원을 볼 수 있습니다. WEBSERVICE 자원을 마우스 오른쪽 단추로 클릭하고 **관련된 열기** > **URI** 맵을 선택하십시오.

CICS 탐색기 보기는 PIPELINE 자원의 이름, URIMAP 자원 및 각 웹 서비스와 연관되는 대상 프로그램을 표시합니다. 이 예제에서, WEBSERVICE 자원이 아웃바운드 요청을 위한 것이므로 WEBSERVICE(dispatchOrder)에 대한 URIAP 또는 대상 프로그램이 없습니다.

WEBSERVICE(dispatchOrderEndpoint)는 디스패치 주문 서비스의 로컬 CICS 구현을 나타냅니다.

자원 정의 온라인(RDO)로 WEBSERVICE 자원 작성

파이프라인 스캐닝 메커니즘을 사용하여 WEBSERVICE 자원을 설치하는 것에 대한 대체로서 자원 정의 온라인(RDO)을 사용하여 이를 작성하고 설치할 수 있습니다.

시작하기 전에

중요사항: RDO를 사용하여 WEBSERVICE 및 URIMAP 자원을 정의하면 PIPELINE의 픽업 디렉토리에 웹 서비스 바인딩 파일이 없다는 것을 확인해야 합니다. 이는 WEBSERVICE 및 URIMAP 자원이 픽업 디렉토리의 파이프라인 스캔 동안 동적으로 설치되지 않는다는 것을 확인합니다. 그렇지 않으면, PIPELINE에서 WSDIR에 지정된 값이 없다는 것을 확인할 수 있습니다. 그러나, WSDIR에 대한 값을 지정하지 않으면 픽업 디렉토리의 어떤 파이프라인 스캔이 발생하지 않습니다. 따라서, 모든 WEBSERVICE 및 URIMAP 자원은 RDO를 사용하여 작성되고 설치되어야 합니다.

프로시저

1. CICS 탐색기를 사용하여 예제 애플리케이션의 문의 카탈로그 기능에 대한 WEBSERVICE 정의를 작성하십시오.
 - a. 정의 > 웹 서비스 정의를 선택하여 CICS 탐색기를 사용하는 WEBSERVICE 정의를 작성하십시오.

- b. 웹 서비스 정의 보기에서 마우스 오른쪽 단추를 클릭하고 새로 작성을 선택하여 새 **WEBSERVICE** 정의를 작성하십시오.
 - c. **DFH\$EXWS**를 자원 그룹 텍스트 상자에 입력하고, **EXINQCWS**를 이름 텍스트 상자에 입력하며, **EXPIPE01**을 파이프라인 텍스트 상자에 입력하거나 Java 기반 파이프라인에 대해 **EXPIPE03**을 입력하십시오. **WEBSERVICE** 정의를 작성할 수 있으려면 **WSBind** 속성을 입력해야 합니다. **WSBind** 파일 텍스트 상자에 **/usr/lpp/cicsts/samples/webservices/wsbind/provider/inquireCatalog.wsbind**를 입력하십시오.
 - d. 완료를 클릭하여 **WEBSERVICE** 정의를 작성하십시오.
2. 예제 애플리케이션의 각각의 다음 기능에 대해 이전 단계를 반복하십시오.

기능	WEBSERVICE 이름	PIPELINE 속성	WSBind 속성
INQUIRE SINGLE ITEM	EXINQSWWS	EXPIPE01 또는 EXPIPE03	/usr/lpp/cicsts/samples /webservices/wsbind /provider/inquireSingle.wsbind
PLACE ORDER	EXORDRWS	EXPIPE01 또는 EXPIPE03	/usr/lpp/cicsts/samples /webservices/wsbind /provider/placeOrder.wsbind
DISPATCH STOCK	EXODRQWS	EXPIPE02 또는 EXPIPE04	/usr/lpp/cicsts/samples /webservices/wsbind /requester/dispatchOrder.wsbind
DISPATCH STOCK 엔드포인트 (선택사항)	EXODEPWS	EXPIPE01 또는 EXPIPE03	/usr/lpp/cicsts/samples /webservices/wsbind /provider/dispatchOrderEndpoint.wsbind

자원 정의 온라인(RDO)으로 URIMAP 자원 작성

파이프라인 스캐닝 메커니즘을 사용하여 URIMAP 자원을 설치하는 것에 대한 대체로써 자원 정의 온라인(RDO)을 사용하여 이를 작성하고 설치할 수 있습니다.

시작하기 전에

중요사항: RDO를 사용하여 WEBSERVICE 및 URIMAP 자원을 정의하면 PIPELINE의 픽업 디렉토리에 웹 서비스 바인딩 파일이 없다는 것을 확인해야 합니다. 이는 WEBSERVICE 및 URIMAP 자원이 픽업 디렉토리의 파이프라인 스캔 동안 동적으로 설치되지 않는다는 것을 확인합니다. 그렇지 않으면, PIPELINE에서 WSDIR에 지정된 값이 없다는 것을 확인할 수 있습니다. 그러나, WSDIR에 대한 값을 지정하지 않으면 픽업 디렉토리의 어떤 파이프라인 스캔이 발생하지 않습니다. 따라서, 모든 WEBSERVICE 및 URIMAP 자원은 RDO를 사용하여 작성되고 설치되어야 합니다.

프로시저

1. CICS 탐색기를 사용하여 예제 애플리케이션의 문의 카탈로그 기능에 대한 URIMAP 정의를 작성하십시오.
 - a. 정의 > **URI 맵** 정의를 선택하여 CICS 탐색기에서 URIMAP 정의를 작성하십시오.

- b. URI 맵 정의 보기에서 마우스 오른쪽 단추를 클릭하고 새로 작성을 선택하여 새 URIMAP 정의를 작성하십시오.
 - c. INQCURI를 이름 텍스트 상자에 입력하고 *를 호스트 텍스트 상자에 입력하십시오. URIMAP 정의를 작성할 수 있으면 Path 속성을 입력해야 합니다. 경로 텍스트 상자에 /exampleApp/inquireCatalog를 입력하십시오. 사용법은 파이프라인으로 설정되어야 합니다. PIPELINE 자원은 Java 기반 파이프라인에 대해 EXPIPE01 또는 EXPIPE03입니다.
 - d. 완료를 클릭하여 URIMAP 정의를 완료하십시오.
 - e. 편집기를 열려면 새 URIMAP 자원을 두 번 클릭하십시오. 편집기의 속성 탭에서 웹 서비스 속성을 EXINQWS로 설정하고 TCP/IP 서비스를 SOAPPORT로 설정하십시오.
2. 예제 애플리케이션의 각각의 남은 기능에 대해 이전 단계를 반복하십시오. URIMAP에 다음 이름을 사용하십시오.

기능	URIMAP 이름
INQUIRE SINGLE ITEM	INQSURI
PLACE ORDER	ORDRURI
DISPATCH STOCK	필요 없음
DISPATCH STOCK 엔드포인트(선택사항)	ODEPURI

3. 각 URIMAP에 다음 고유 속성을 지정하십시오.

기능	URIMAP 이름	PATH	WEBSERVICE
INQUIRE SINGLE ITEM	INQSURI	/exampleApp/inquireSingle	EXINQWS
PLACE ORDER	ORDRURI	/exampleApp/placeOrder	EXORDRWS
DISPATCH STOCK 엔드포인트(선택사항)	ODEPURI	/exampleApp/dispatchOrder	EXODEPWS

설치 완료

설치를 완료하려면 자원 정의를 포함하는 RDO 그룹을 설치하십시오.

프로시저

자원 그룹 정의 창에서 자원 그룹을 마우스 오른쪽 단추로 클릭하십시오. 설치를 선택하십시오. 사용자의 CICSplex가 정확한지 확인하고 사용자가 대상 리전을 선택하는지 확인한 후 확인을 클릭하십시오.

결과

RDO가 설치되고 애플리케이션이 사용 준비됩니다.

웹 클라이언트 구성

웹 클라이언트를 사용할 수 있으려면 웹 클라이언트 프론트 엔드에 대한 Java 웹 아카이브(WAR)를 지원되는 환경 중 하나에 배치하고 CICS 시스템에서 적절한 엔드포인트를 호출하기 위해 이를 구성해야 합니다.

이 태스크 정보

다음 환경이 웹 클라이언트 프론트 엔드 애플리케이션 ExampleAppClientV7.war 및 ExampleAppWrapperClientV7.war의 두 가지 버전에 지원됩니다.

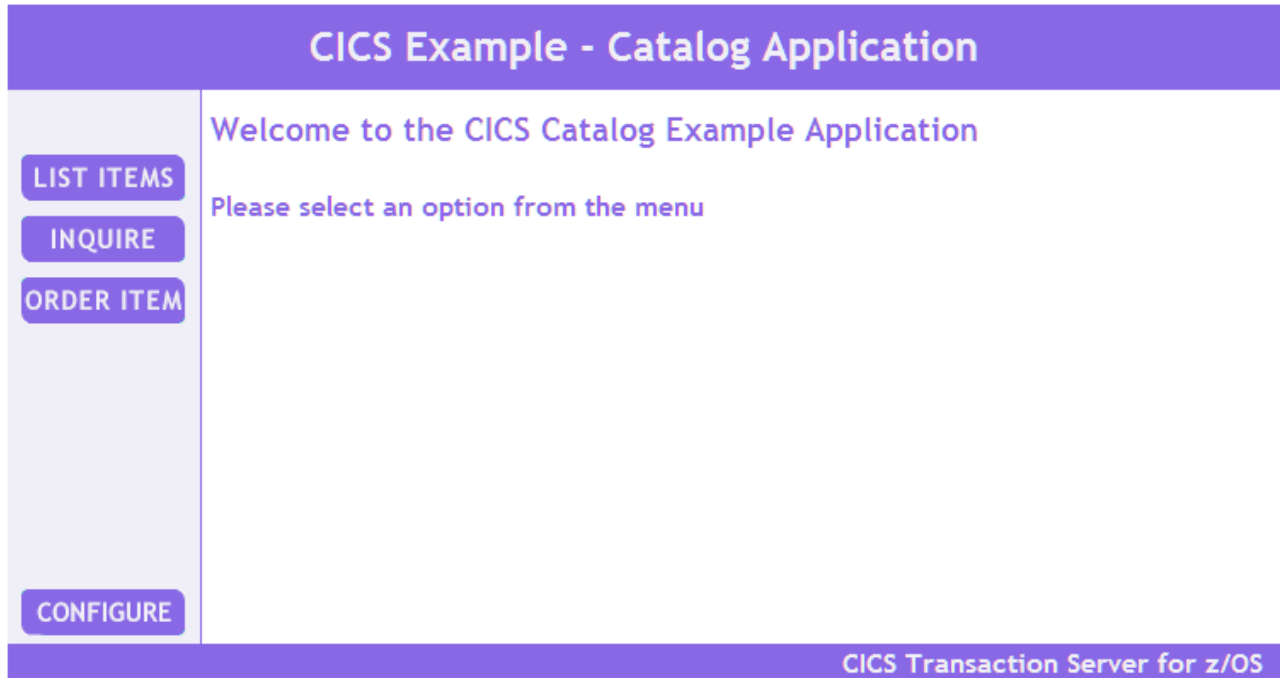
- WebSphere Liberty Profile V8.5.5가 있는 CICS Liberty JVM 서버

WAR 파일은 z/OS UNIX의 *hlq/samples/webservices/client* 디렉토리에 있습니다.

프로시저

1. 웹 클라이언트를 시작하려면 웹 브라우저에 다음 URL을 입력하십시오. 여기서 *mylibertyserver*는 웹 클라이언트가 설치되는 JVM Liberty 서버의 호스트 이름입니다.
 - ExampleAppClientV7.war의 경우 URL `http://mylibertyserver:9080/ExampleAppClientV7/`을 사용하십시오.
 - ExampleAppWrapperClientV7.war의 경우 URL `http://mylibertyserver:9080/ExampleAppWrapperClientV7/`을 사용하십시오.

예제 애플리케이션에는 다음 페이지가 표시됩니다.



2. **CONFIGURE**를 클릭하여 구성 페이지를 표시하십시오. 구성 페이지가 표시됩니다.
3. 문의 카탈로그 문의 항목 및 위치 주문의 웹 서비스에 대해 새 엔드 포인트를 입력하십시오.
 - a. URL에서 myCicsServer 문자열을 CICS가 실행 중인 시스템의 이름으로 대체하십시오.
 - b. 포트 번호 8080을 TCPIP SERVICE 정의 자원에 구성된 포트 번호로 대체하십시오.
4. **SUBMIT**를 클릭하십시오.

결과

웹 애플리케이션이 실행될 준비가 됩니다.

다음에 수행할 작업

웹 서비스 호출에 대한 URL이 HTTP 세션에 저장됩니다. 따라서, 웹 브라우저가 처음 클라이언트에 연결될 때마다 이 구성 단계를 반복할 필요가 있습니다.

웹 서비스 사용 애플리케이션 실행

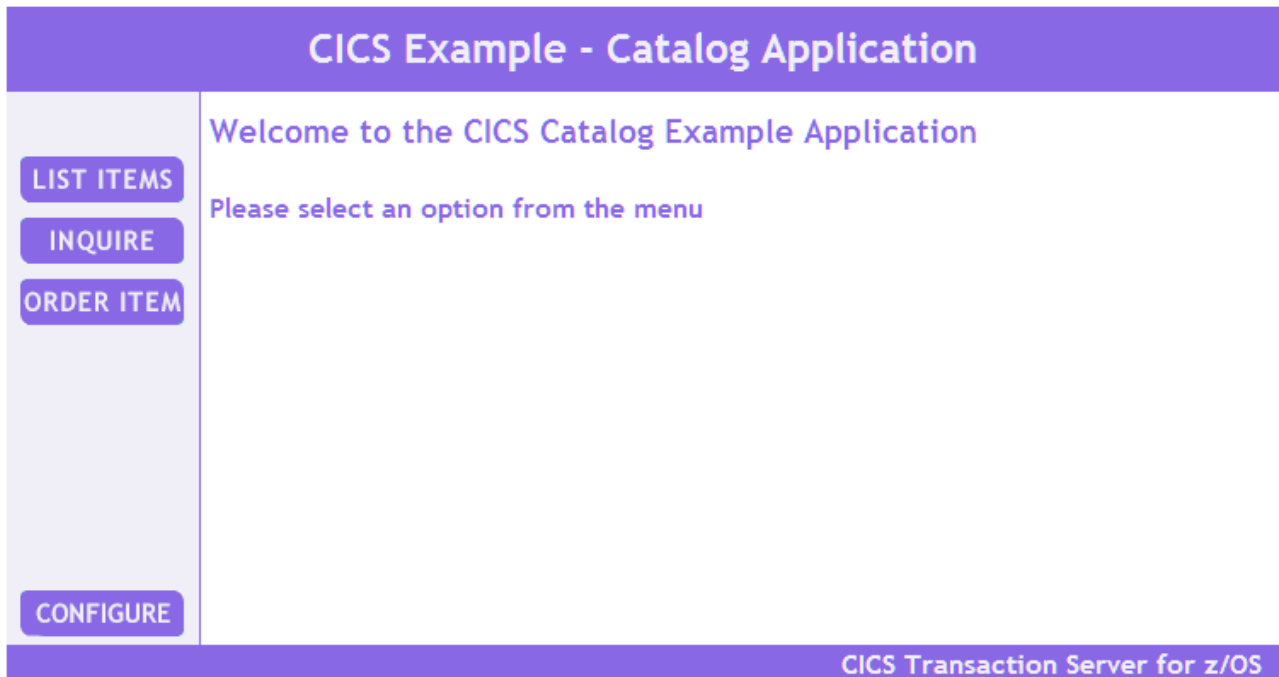
웹 브라우저로부터 예제 애플리케이션을 호출할 수 있습니다.

이 태스크 정보

진행하기 전에 웹 클라이언트가 구성되어 있어야 합니다. 예제 애플리케이션 구성의 내용을 참조하십시오.

프로시저

1. 사용자의 웹 브라우저에 URL(<http://myserver:9080/ExampleAppClientWeb/V7/>)을 입력하십시오. 여기서, *myserver*는 웹 서비스 클라이언트가 설치되는 서버의 호스트 이름입니다. 예제 애플리케이션에는 다음 페이지가 표시됩니다.



2. **INQUIRE** 단추를 클릭하십시오. 예제 애플리케이션에는 다음 페이지가 표시됩니다.

CICS Example - Catalog Application	
	Enter Catalog Item Reference Number
INQUIRE	Start List From Item Number 0010
ORDER ITEM	SUBMIT
BACK	
CONFIGURE	
CICS Transaction Server for z/OS	

3. 항목 번호를 입력하고 **SUBMIT** 단추를 클릭하십시오.

팁: 기본 애플리케이션은 순서 0010, 0020.. 0210까지 항목 번호를 미리 제공받습니다.

애플리케이션은 입력한 항목 번호로 시작하는 카탈로그에 항목 목록이 들어 있는 다음 페이지를 표시합니다.

CICS Example - Catalog Application

LIST ITEMS

INQUIRE

ORDER ITEM

BACK

CONFIGURE

Item Details - Select Item to Place Order

Item	Description	In Stock	On Order	Cost	Select
0010	Ball Pens Black 24pk	13	0	£2.90	☐
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

CICS Transaction Server for z/OS

4. 주문할 항목을 선택하십시오.
 - a. 주문하려는 항목의 선택 열에서 단일 선택 단추를 클릭하십시오.
 - b. **SUBMIT** 단추를 클릭하십시오.

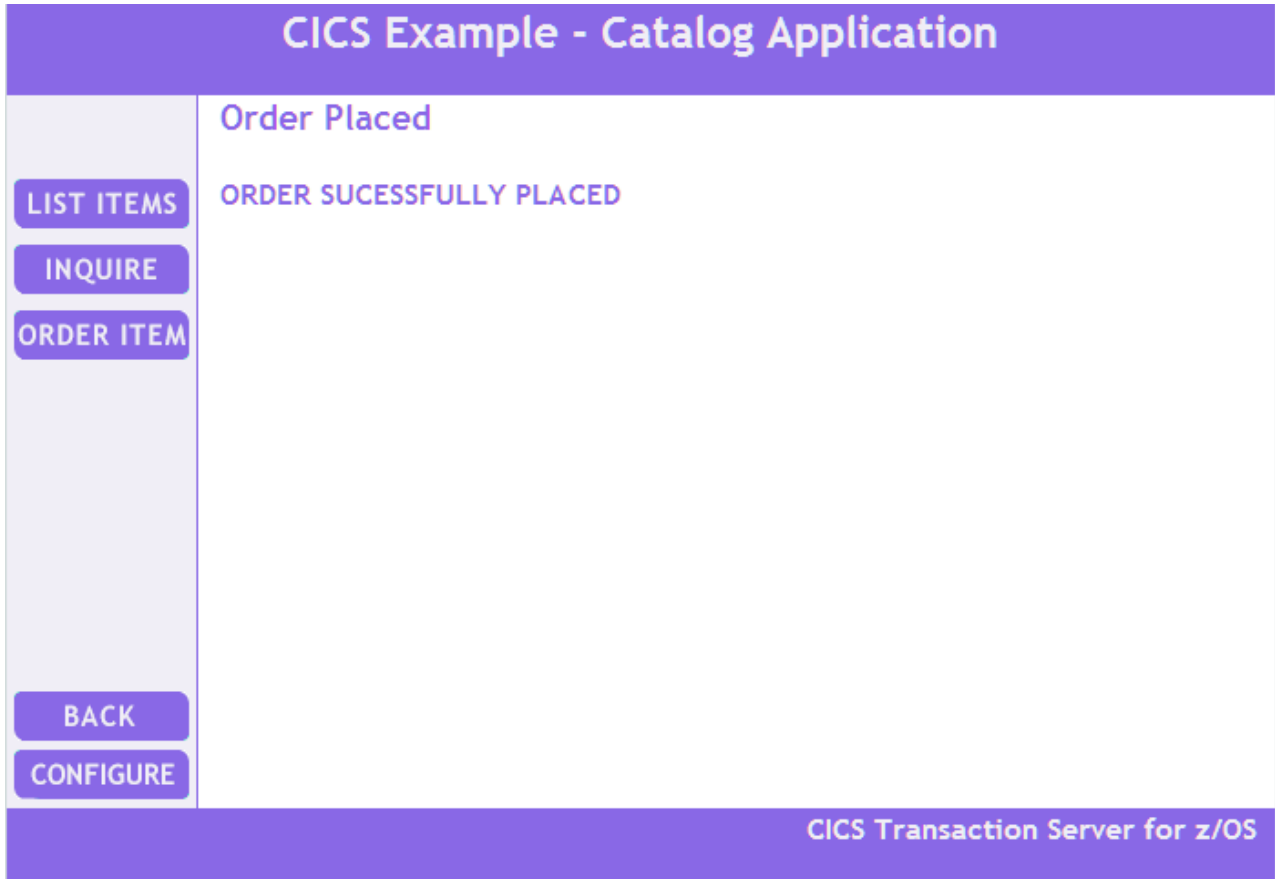
애플리케이션에 다음 페이지가 표시됩니다.

CICS Example - Catalog Application		
Enter Order Details		
LIST ITEMS	Item Reference Number	0040
INQUIRE	Quantity	001
	User Name	AUSER
	Department Name	CICS1
	SUBMIT	
BACK		
CONFIGURE		
CICS Transaction Server for z/OS		

5. 주문을 하려면 다음 정보를 입력하십시오.

- Quantity** 필드를 완료하십시오. 주문하려는 항목의 수를 지정하십시오.
- User Name** 필드를 완료하십시오. 1 - 8자의 문자열을 입력하십시오. 기본 애플리케이션은 여기에 입력한 값을 확인하지 않습니다.
- Department Name** 필드를 완료하십시오. 1 - 8자의 문자열을 입력하십시오. 기본 애플리케이션은 여기에 입력한 값을 확인하지 않습니다.
- SUBMIT** 단추를 클릭하십시오.

애플리케이션에서는 주문을 확인하기 위해 다음 페이지가 표시됩니다.



예제 애플리케이션 배치

웹 서비스 보조를 사용하여 웹 서비스로 예제 애플리케이션의 파트를 배치할 수 있습니다. 애플리케이션이 이 태스크를 수행하지 않고 작동할지라도 예제 애플리케이션을 확장하기 위해 자체 애플리케이션을 배치하려는 경우 유사한 태스크를 수행해야 합니다.

프로그램 인터페이스 추출

CICS 웹 서비스 지원으로 프로그램을 배치하려면 COMMAREA 또는 컨테이너 인터페이스와 일치하는 카피북을 작성해야 합니다.

이 태스크 정보

이 예제에서, 중앙 카탈로그 관리자 프로그램(DFH0XCMN)의 INQUIRE SINGLE ITEM 기능이 웹 서비스로 배치됩니다. 이 프로그램에 대한 인터페이스는 COMMAREA이고, COMMAREA의 구조가 카피북 DFH0XCP1에 정의됩니다.

```
* Catalogue COMMAREA structure
      03 CA-REQUEST-ID           PIC X(6).
      03 CA-RETURN-CODE          PIC 9(2).
      03 CA-RESPONSE-MESSAGE     PIC X(79).
```

```

03 CA-REQUEST-SPECIFIC          PIC X(911).
* Fields used in Inquire Catalog
03 CA-INQUIRE-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
05 CA-LIST-START-REF            PIC 9(4).
05 CA-LAST-ITEM-REF            PIC 9(4).
05 CA-ITEM-COUNT                PIC 9(3).
05 CA-INQUIRY-RESPONSE-DATA PIC X(900).
05 CA-CAT-ITEM REDEFINES CA-INQUIRY-RESPONSE-DATA
                                OCCURS 15 TIMES.
                                07 CA-ITEM-REF            PIC 9(4).
                                07 CA-DESCRIPTION        PIC X(40).
                                07 CA-DEPARTMENT        PIC 9(3).
                                07 CA-COST              PIC X(6).
                                07 IN-STOCK              PIC 9(4).
                                07 ON-ORDER              PIC 9(3).
* Fields used in Inquire Single
03 CA-INQUIRE-SINGLE REDEFINES CA-REQUEST-SPECIFIC.
05 CA-ITEM-REF-REQ              PIC 9(4).
05 FILLER                      PIC 9(4).
05 FILLER                      PIC 9(3).
05 CA-SINGLE-ITEM.
                                07 CA-SNGL-ITEM-REF      PIC 9(4).
                                07 CA-SNGL-DESCRIPTION    PIC X(40).
                                07 CA-SNGL-DEPARTMENT    PIC 9(3).
                                07 CA-SNGL-COST          PIC X(6).
                                07 IN-SNGL-STOCK         PIC 9(4).
                                07 ON-SNGL-ORDER         PIC 9(3).
                                05 FILLER                PIC X(840).
* Fields used in Place Order
03 CA-ORDER-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
05 CA-USERID                   PIC X(8).
05 CA-CHARGE-DEPT              PIC X(8).
05 CA-ITEM-REF-NUMBER          PIC 9(4).
05 CA-QUANTITY-REQ             PIC 9(3).
05 FILLER                      PIC X(888).

```

카피북은 INQUIRE CATALOG, INQUIRE SINGLE ITEM 및 PLACE ORDER 기능에 대한 세 개의 개별 인터페이스를 정의합니다. 이는 카피북에서 서로에게 오버레이됩니다. 그러나, DFHLS2WS 유틸리티는 REDEFINES 명령문을 지원하지 않습니다. 따라서, 문의 단일 기능과 관련되는 해당 섹션을 결합된 카피북에서 추출해야 합니다.

```

* Catalogue COMMAREA structure
03 CA-REQUEST-ID                PIC X(6).
03 CA-RETURN-CODE               PIC 9(2) DISPLAY.
03 CA-RESPONSE-MESSAGE         PIC X(79).
* Fields used in Inquire Single
03 CA-INQUIRE-SINGLE.
05 CA-ITEM-REF-REQ              PIC 9(4) DISPLAY.
05 FILLER                      PIC X(4) DISPLAY.
05 FILLER                      PIC X(3) DISPLAY.
05 CA-SINGLE-ITEM.
                                07 CA-SNGL-ITEM-REF      PIC 9(4) DISPLAY.
                                07 CA-SNGL-DESCRIPTION    PIC X(40).
                                07 CA-SNGL-DEPARTMENT    PIC 9(3) DISPLAY.
                                07 CA-SNGL-COST          PIC X(6).

```

	07 IN-SNGL-STOCK	PIC 9(4) DISPLAY.
	07 ON-SNGL-ORDER	PIC 9(3) DISPLAY.
05 FILLER		PIC X(840).

재정의된 요소 CA-REQUEST-SPECIFIC이 제거되고 문의 단일 기능에 대해 이를 재정의한 카피북의 섹션에 의해 대체됩니다. 카피북은 이제 웹 서비스 지원용으로 적합합니다.

카피북은 DFH0XCP4 카피북으로 예제 애플리케이션과 함께 제공됩니다.

웹 서비스 지원 프로그램 DFHLS2WS 실행

CICS 웹 서비스 지원은 사용자가 기존 CICS 애플리케이션을 웹 서비스로 변환시키도록 도움을 줄 수 있고, CICS 애플리케이션이 외부 제공자가 제공하는 웹 서비스를 사용할 수 있도록 도움을 줄 수 있는 두 개의 일괄처리 프로그램으로 구성됩니다. 프로그램 DFHLS2WS는 웹 서비스 바인딩 파일 및 웹 서비스 설명을 생성하기 위해 언어 구조를 변환합니다.

프로시저

1. 제공된 샘플 JCL을 적합한 작업 파일로 복사하십시오. JCL이 samples/webservices/JCL/LS2WS에서 제공됩니다.
2. 유효한 JOB 카드를 JCL에 추가하십시오.
3. DFHLS2WS에 대한 매개변수를 코드화하십시오. 예제 애플리케이션의 INQUIRE SINGLE ITEM 기능에 다음 매개변수가 필요합니다.

```
//INPUT.SYSUT1 DD *
LOGFILE=/u/exampleapp/wsbind/inquireSingle.log
PDSLIB=CICSHLQ.SDFHSAMP
REQMEM=DFH0XCP4
RESPMEM=DFH0XCP4
LANG=COBOL
PGMNAME=DFH0XCMN
PGMINT=COMMAREA
URI=mycicsserver:myport/exampleApp/inquireSingle
WSBIND=/u/exampleapp/wsbind/inquireSingle.wsbind
WSDL=/u/exampleapp/wsd1/inquireSingle.wsd1
*/
```

LOGFILE=/u/exampleapp/wsbind/inquireSingle.log

DFHLS2WS로부터 진단 정보를 기록하기 위해 사용되는 파일. 파일은 IBM 소프트웨어 지원 조직에서만 사용됩니다.

PDSLIB=CICSHLQ.SDFHSAMP

웹 서비스 지원이 요청 및 응답 구조를 정의하는 카피북을 찾는 파티션된 데이터 세트(PDS)의 이름. 이 예제에서, 이는 CICS 설치된 데이터 세트 SDFHSAMP입니다.

REQMEM=DFH0XCP4

RESPMEM=DFH0XCP4

이러한 매개변수는 요청에 대한 언어 구조 및 프로그램에 대한 응답을 정의합니다. 예제에서, 요청 및 응답에 동일한 구조가 있고 동일한 카피북에 의해 정의됩니다.

LANG=COBOL

대상 프로그램 및 데이터 구조가 COBOL로 작성됩니다.

PGMNAME=DFH0XCMN

웹 서비스 요청이 수신될 때 시작되는 대상 프로그램의 이름.

PGMINT=COMMAREA

대상 프로그램이 COMMAREA 인터페이스로 호출됩니다.

URI=mycicsserver:myport/exampleApp/inquireSingle

생성된 웹 서비스 정의에 사용되고, 수신 요청을 올바른 웹 서비스에 매핑하는 URIMAP 자원을 작성하기 위해 사용되는 URI의 고유 파트. 지정된 값은 외부 클라이언트에 사용 가능한 서비스의 결과로 생깁니다.

`http://mycicsserver:myport/exampleApp/inquireSingle`

여기서, *mycicsserver* 및 *myport*는 이 WEBSERVICE 자원이 설치되는 CICS 서버 주소 및 포트입니다.

참고: 매개변수는 선두 '/'를 가지고 있지 않습니다.

WSBIND=/u/exampleapp/wsbind/inquireSingle.wsbind

웹 서비스 바인딩 파일이 작성되는 z/OS UNIX의 위치.

참고: 파일이 파이프라인 스캐닝 메커니즘과 함께 사용되는 경우, 확장자 `.wsbind`가 있어야 합니다.

WSDL=/u/exampleapp/wsd1/inquireSingle.wsd1

생성된 웹 서비스 설명을 포함하는 파일이 작성되는 z/OS UNIX의 위치. 웹 서비스 바인딩 파일 및 해당 웹 서비스 설명에 대한 일치하는 이름을 사용하는 것은 좋은 관행입니다.

통상적으로, 웹 서비스 설명을 포함하는 파일에는 확장자 `.wsdl`이 있습니다.

웹 서비스 설명에서는 클라이언트가 웹 서비스에 액세스하기 위해 사용해야 하는 정보를 제공합니다. 요청 및 응답의 XML 스키마 정의 및 서비스에 대한 위치 정보를 포함합니다.

4. 작업을 실행하십시오. 웹 서비스 설명 및 웹 서비스 바인딩 파일이 지정된 위치에서 작성됩니다.

생성된 WSDL 문서의 예제

웹 서비스 지원 프로그램 DFHLS2WS가 실행될 때 생성되는 웹 서비스 설명(WSDL) 문서의 예제.

```
<?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">
      ... most of the schema for the request 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>
</definitions>
```

```

</binding>
<service name="DFH0XCMNService">
  <port binding="tns:DFH0XCMNHTTPSoapBinding" name="DFH0XCMNPort">
    <!-- This soap:address indicates the location of the web service over HTTP.
    Please replace "my-server" with the TCP/IP host name of your CICS region.
    Please replace "my-port" with the port number of your CICS TCP/IP SERVICE. -->
    <soap:address location="http://my-server:my-port/exampleApp/inquireSingles.log"/>
    <!-- This soap:address indicates the location of the web service over HTTPS. -->
    <!-- <soap:address location="https://my-server:my-port/exampleApp/inquireSingles.log"/> -->
    <!-- This soap:address indicates the location of the web service over Websphere MQSeries.
    Please replace "my-queue" with the appropriate queue name. -->
    <!-- <soap:address location="jms:/queue?destination=my-queue&connectionFactory=()&
    targetService=/exampleApp/inquireSingles.log&initialContextFactory=com.ibm.mq.jms.NoJndi" /> -->
  </port>
</service>
</definitions>

```

웹 서비스 바인딩 파일 배치

DFHLS2WS에 의해 작성된 WEBSERVICE 바인딩 파일은 PIPELINE 자원을 설치할 때 동적으로 CICS 리전에 배치됩니다.

이 태스크 정보

파이프라인 스캔 명령이 실행될 때, CICS는 .wsbind 확장자로 WEBSERVICE 바인딩 파일을 검색하기 위해 픽업 디렉토리를 스캔합니다. 발견된 각 바인딩 파일의 경우, CICS는 WEBSERVICE 자원을 설치할지 여부를 판별합니다.

웹 서비스가 설치된 PIPELINE 및 설치된 WEBSERVICE 자원에 JCL에 제공된 것처럼 URI를 맵핑하기 위해 URIMAP 자원도 작성됩니다. 스캐닝된 WEBSERVICE 자원이 삭제될 때, 이와 연관된 URIMAP 자원도 삭제됩니다.

프로시저

1. 정의 > 파이프라인 정의를 선택하여 CICS 탐색기에서 제공자 파이프라인 PIPELINE (EXPIPE01)에 대한 PIPELINE 정의를 수정하십시오. EXPIPE01을 두 번 클릭하여 파이프라인 정의 (EXPIPE01) 편집기를 여십시오. 속성 탭에서 **WS** 디렉토리 매개변수를 /u/exampleapp/wsbind로 변경하십시오. 이 픽업 디렉토리에는 DFHLS2WS로 생성한 WEBSERVICE 바인딩이 포함됩니다.
2. 애플리케이션이 사용하는 기타 WEBSERVICE 바인딩을 동일한 디렉토리에 복사하십시오. 이 예제에서 다음 파일이 복사됩니다.

inquireCatalog

placeOrder

/usr/lpp/cicsts/samples/webservices/wsbind/provider 디렉토리에서 제공됩니다.

3. PIPELINE 자원을 설치하십시오.

결과

CICS는 두 개의 URIMAP 자원을 작성합니다. 첫 번째 URIMAP 정의는 인바운드 웹 서비스 요청의 URI를 요청을 제공하는 기타 자원(예: PIPELINE 자원)으로 맵핑하는 정보를 포함할 때 서비스 제공자에 필요합니다. 두 번째 URIMAP에는 웹 서비스와 연관되는 문서 및 WSDL 문서에 대한 인바운드 요청의 URI를 맵핑하는 정보가 포함됩니다.

기본 애플리케이션의 컴포넌트

이러한 테이블을 사용하여 SDFHSAMP 샘플에 제공되는 기본 애플리케이션 및 멤버의 컴포넌트를 이해하십시오. 나열된 SDFHSAMP 멤버에는 BMS 맵, COBOL 소스 및 기본 애플리케이션, 웹 서비스 클라이언트 애플리케이션과 랩퍼 모듈에 대한 카피북이 포함됩니다.

표 23. BMS 맵을 포함하는 SDFHSAMP 멤버

멤버명	설명
DFH0XS1	기본 메뉴 화면에 대한 맵(EXMENU) 및 사용자 주문에 대한 세부사항에 대한 맵(EXORDR)으로 구성되는 맵핑 세트에 대한 BMS 매크로.
DFH0XS2	문의 카탈로그 화면에 대한 맵(EXINQC)으로 구성하는 맵핑 세트에 대한 BMS 매크로.
DFH0XS3	CICS 예제 카탈로그 애플리케이션 구성 화면에 대한 맵(EXCONF)으로 구성되는 맵핑 세트에 대한 BMS 매크로.
DFH0XM1	DFH0XS1을 조립하여 생성된 COBOL 카피북. DFH0XGUI 및 DFH0XCUI에는 이 복사본이 포함됩니다.
DFH0XM2U	카피북 프로그래밍이 용이하도록 색인화된 어레이 구조를 포함하도록 결과를 편집하고 DFH0XS2를 조립하여 생성된 COBOL 카피북. DFH0XGUI 및 DFH0XCUI에 이 카피북이 포함됩니다.
DFH0XM3	DFH0XS3을 조립하여 생성된 COBOL 카피북. DFH0XCFG에는 이 카피북이 포함됨

표 24. 기본 애플리케이션에 대한 COBOL 소스를 포함하는 SDFHSAMP 멤버

멤버명	설명
DFH0XCFG	VSAM 구성 파일을 읽고 갱신하기 위해 트랜잭션 ECFG가 호출한 프로그램.
DFH0XCMN	카탈로그 애플리케이션에 대한 제어기 프로그램. 제어기 프로그램을 통해 모든 요청이 전달됩니다.
DFH0XGUI	터미널 사용자로 BMS 맵을 전송하고 터미널 사용자로부터 맵을 수신하는 것을 관리하기 위해 트랜잭션 EGUI가 호출하는 프로그램. 이 프로그램은 프로그램 DFH0XCMN에 링크됩니다.
DFH0XODE	주문 디스패치 웹 서비스에 대한 엔드포인트의 두 가지 버전 중 하나. 이는 CICS에서 실행되는 버전입니다. 이 프로그램은 리턴 COMMAREA에서 "디스패치에서 주문" 텍스트를 설정합니다.
DFH0XSDS	VSAM 카탈로그 파일이 설정되지 않을 때 애플리케이션이 작동하도록 허용하는 데이터 저장소 프로그램의 스텝 또는 더미 버전. DFH0XSDS는 VSAM 파일에 저장된 데이터보다 프로그램에 정의된 데이터를 사용합니다.

표 24. 기본 애플리케이션에 대한 COBOL 소스를 포함하는 SDFHSAMP 멤버 (계속)

멤버명	설명
DFH0XSOD	주문 디스패치 프로그램의 스텝 버전. COMMAREA의 리턴 코드를 0으로 설정하고 해당 호출자로 리턴합니다. DFH0XSOD는 아웃바운드 웹 서비스가 필요하지 않을 때 사용됩니다.
DFH0XSSM	재고 관리자(보충) 프로그램의 스텝 버전. DFH0XSSM은 COMMAREA의 리턴 코드를 0으로 설정하고 해당 호출자로 리턴합니다.
DFH0XVDS	데이터 저장소 프로그램의 VSAM 버전. DFH0XVDS는 카탈로그의 읽기 및 갱신을 수행하기 위해 VSAM 파일에 액세스합니다.
DFH0XWOD	주문 디스패치 프로그램의 웹 서비스 버전. DFH0XWOD는 주문 디스패처에 대한 아웃바운드 웹 서비스를 호출하기 위해 EXEC CICS INVOKE WEBSERVICE를 실행합니다.

표 25. 기본 애플리케이션에 대한 COBOL 카피북을 포함하는 SDFHSAMP 멤버

멤버명	설명
DFH0XCP1	문의 카탈로그, 문의 단일 및 위치 주문 기능에 대한 요청 및 응답을 포함하는 COMMAREA 구조를 정의합니다. DFH0XCMN, DFH0XCUI, DFH0XECC, DFH0XGUI, DFH0XICW, DFH0XISW, DFH0XPOW, DFH0XSDS 및 DFH0XVDS 프로그램에 이 카피북이 포함됩니다.
DFH0XCP2	주문 디스패처 및 재고 관리자 모듈에 대한 COMMAREA 구조를 정의합니다. DFH0XCMN, DFH0XSOD, DFH0XSSM 및 DFH0XWOD 프로그램에 이 카피북이 포함됩니다.
DFH0XCP3	문의 카탈로그 요청 및 응답에 대한 데이터 구조를 정의합니다. inquireCatalog.wsd1 및 inquireCatalog.wsbinding을 생산하기 위해 DFHLS2WS에 대한 입력으로 사용됩니다.
DFH0XCP4	문의 단일 요청 및 응답에 대한 데이터 구조를 정의합니다. inquireSingle.wsd1 및 inquireSingle.wsbinding을 생산하기 위해 DFHLS2WS에 대한 입력으로 사용됩니다.
DFH0XCP5	위치 주문 요청 및 응답에 대한 데이터 구조를 정의합니다. placeOrder.wsd1 및 placeOrder.wsbinding을 생산하기 위해 DFHLS2WS에 대한 입력으로 사용됩니다.
DFH0XCP6	디스패치 주문 요청 및 응답에 대한 데이터 구조를 정의합니다. dispatchOrder.wsd1 및 dispatchOrder.wsbinding을 생산하기 위해 DFHLS2WS에 대한 입력으로 사용됩니다.
DFH0XCP7	디스패치 주문 요청에 대한 데이터 구조를 정의합니다. DFH0XODE 및 DFH0XWOD 프로그램에 이 카피북이 포함됩니다.
DFH0XCP8	디스패치 주문 응답에 대한 데이터 구조를 정의합니다. DFH0XODE 및 DFH0XWOD 프로그램에 이 카피북이 포함됩니다.

표 26. CICS에서 실행되는 웹 서비스 클라이언트 애플리케이션에 대한 COBOL 소스 코드를 포함하는 SDFHSAMP 멤버

멤버명	설명
DFH0XCUI	터미널 사용자로 BMS 맵을 전송하고 터미널 사용자로부터 맵을 수신하는 것을 관리하기 위해 트랜잭션 ECLI가 호출하는 프로그램. DFH0XECC 프로그램에 링크됩니다.

표 26. CICS에서 실행되는 웹 서비스 클라이언트 애플리케이션에 대한 COBOL 소스 코드를 포함하는 SDFHSAMP 멤버 (계속)

멤버명	설명
DFH0XECC	EXEC CICS INVOKE WEBSERVICE 명령을 사용하여 기본 애플리케이션에 대한 아웃바운드 웹 서비스를 요청합니다. 지정된 웹 서비스는 다음 중 하나입니다. inquireCatalogClient inquireSingleClient placeOrderClient

표 27. CICS에서 실행되는 웹 서비스 클라이언트 애플리케이션에 대한 COBOL 카피북을 포함하는 SDFHSAMP 멤버. 이는 모두 DFHWS2LS에 의해 생성되고 DFH0XECC 프로그램에 의해 포함됩니다.

멤버명	설명
DFH0XCPA	문의 카탈로그 요청에 대한 데이터 구조를 정의합니다.
DFH0XCPB	문의 카탈로그 응답에 대한 데이터 구조를 정의합니다.
DFH0XCPC	문의 단일 요청에 대한 데이터 구조를 정의합니다.
DFH0XCPD	문의 단일 응답에 대한 데이터 구조를 정의합니다.
DFH0XCPE	위치 주문 요청에 대한 데이터 구조를 정의합니다.
DFH0XCPF	위치 주문 응답에 대한 데이터 구조를 정의합니다.

표 28. 랩퍼 모듈에 대한 COBOL 소스 코드를 포함하는 SDFHSAMP 멤버

멤버명	설명
DFH0XICW	inquireCatalog 서비스에 대한 랩퍼 프로그램.
DFH0XISW	inquireSingle 서비스에 대한 랩퍼 프로그램.
DFH0XPOW	purchaseOrder 서비스에 대한 랩퍼 프로그램.

표 29. 랩퍼 모듈에 대한 COBOL 카피북을 포함하는 SDFHSAMP 멤버

멤버명	설명
DFH0XWC1	문의 카탈로그 요청에 대한 데이터 구조를 정의합니다. DFH0XICW 프로그램에 이 카피북이 포함됩니다.
DFH0XWC2	문의 카탈로그 응답에 대한 데이터 구조를 정의합니다. DFH0XICW 프로그램에 이 카피북이 포함됩니다.
DFH0XWC3	문의 단일 요청에 대한 데이터 구조를 정의합니다. DFH0XISW 프로그램에 이 카피북이 포함됩니다.
DFH0XWC4	문의 단일 응답에 대한 데이터 구조를 정의합니다. DFH0XISW 프로그램에 이 카피북이 포함됩니다.
DFH0XWC5	위치 주문 요청에 대한 데이터 구조를 정의합니다. DFH0XPOW 프로그램에 이 카피북이 포함됩니다.
DFH0XWC6	위치 주문 응답에 대한 데이터 구조를 정의합니다. DFH0XPOW 프로그램에 이 카피북이 포함됩니다.

표 30. CICS 자원 정의

자원 이름	자원 유형	주석
EXAMPLE	CICS 자원 정의 그룹	예제 애플리케이션에 필요한 CICS 자원 정의.
EGUI	TRANSACTION	애플리케이션(사용자 정의할 수 있는)에 대한 BMS 인터페이스를 시작하도록 DFH0XGUI 프로그램을 호출하기 위한 트랜잭션.
ECFG	TRANSACTION	예제 구성 BMS 인터페이스(사용자 정의할 수 있는)를 시작하도록 DFH0XCFG 프로그램을 호출하기 위한 트랜잭션.
EXMPCAT	FILE	애플리케이션 카탈로그(사용자 정의할 수 있음)에 대한 EXMPCAT VSAM 파일의 파일 정의.
EXMPCONF	FILE	EXMPCONF 애플리케이션 구성 파일의 파일 정의.

카탈로그 관리자 프로그램

카탈로그 관리자는 예제 애플리케이션의 비즈니스 로직에 대한 제어 프로그램이고 예제 애플리케이션과의 모든 상호작용이 이를 통해 전달됩니다.

프로그램 로직이 단순한지 확인하기 위해 카탈로그 관리자가 수행하는 유형 검사 및 오류 복구가 제한됩니다. .

카탈로그 관리자는 다양한 조작을 지원합니다. 각 조작에 대한 입력 및 출력 매개변수가 단일 데이터 구조에 정의되고, 이는 COMMAREA의 프로그램과 상호작용합니다.

COMMAREA 구조

표준 CICS 통신 영역(COMMAREA)을 사용하여 서비스 프로그램 및 샘플 클라이언트 사이에 데이터가 전달됩니다.

다음 코드 추출은 카탈로그 관리자 애플리케이션 COMMAREA 구조를 표시합니다.

```
*   Catalogue COMMAREA structure
      03 CA-REQUEST-ID          PIC X(6).
      03 CA-RETURN-CODE         PIC 9(2).
      03 CA-RESPONSE-MESSAGE    PIC X(79).
      03 CA-REQUEST-SPECIFIC    PIC X(911).
*   Fields used in Inquire Catalog
      03 CA-INQUIRE-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
          05 CA-LIST-START-REF    PIC 9(4).
          05 CA-LAST-ITEM-REF     PIC 9(4).
          05 CA-ITEM-COUNT        PIC 9(3).
          05 CA-INQUIRY-RESPONSE-DATA PIC X(900).
          05 CA-CAT-ITEM REDEFINES CA-INQUIRY-RESPONSE-DATA
              OCCURS 15 TIMES.
              07 CA-ITEM-REF      PIC 9(4).
              07 CA-DESCRIPTION   PIC X(40).
```



```

07 CA-DEPARTMENT      PIC 9(3).
07 CA-COST             PIC X(6).
07 IN-STOCK           PIC 9(4).
07 ON-ORDER           PIC 9(3).
*   Fields used in Inquire Single
03 CA-INQUIRE-SINGLE REDEFINES CA-REQUEST-SPECIFIC.
05 CA-ITEM-REF-REQ     PIC 9(4).
05 FILLER              PIC 9(4).
05 FILLER              PIC 9(3).
05 CA-SINGLE-ITEM.
07 CA-SNGL-ITEM-REF    PIC 9(4).
07 CA-SNGL-DESCRIPTION PIC X(40).
07 CA-SNGL-DEPARTMENT PIC 9(3).
07 CA-SNGL-COST        PIC X(6).
07 IN-SNGL-STOCK       PIC 9(4).
07 ON-SNGL-ORDER       PIC 9(3).
05 FILLER              PIC X(840).
*   Fields used in Place Order
03 CA-ORDER-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
05 CA-USERID           PIC X(8).
05 CA-CHARGE-DEPT      PIC X(8).
05 CA-ITEM-REF-NUMBER  PIC 9(4).
05 CA-QUANTITY-REQ     PIC 9(3).
05 FILLER              PIC X(888).

*   Dispatcher/Stock Manager COMMAREA structure
03 CA-ORD-REQUEST-ID   PIC X(6).
03 CA-ORD-RETURN-CODE  PIC 9(2).
03 CA-ORD-RESPONSE-MESSAGE PIC X(79).
03 CA-ORD-REQUEST-SPECIFIC PIC X(23).
*   Fields used in Dispatcher
03 CA-DISPATCH-ORDER REDEFINES CA-ORD-REQUEST-SPECIFIC.
05 CA-ORD-ITEM-REF-NUMBER PIC 9(4).
05 CA-ORD-QUANTITY-REQ    PIC 9(3).
05 CA-ORD-USERID          PIC X(8).
05 CA-ORD-CHARGE-DEPT     PIC X(8).
*   Fields used in Stock Manager
03 CA-STOCK-MANAGER-UPDATE REDEFINES CA-ORD-REQUEST-SPECIFIC.
05 CA-STK-ITEM-REF-NUMBER PIC 9(4).
05 CA-STK-QUANTITY-REQ    PIC 9(3).
05 FILLER                 PIC X(16).

```

리턴 코드

카탈로그 관리자의 각 조작은 다양한 리턴 코드를 리턴할 수 있습니다.

표 31. 카탈로그 관리자 리턴 코드

유형	코드	설명
일반	00	오류없이 완료된 기능
카탈로그 파일	20	항목 참조를 찾을 수 없음
	21	카탈로그 파일의 브라우저 열기, 읽기 또는 종료 오류
	22	파일 갱신 오류

표 31. 카탈로그 관리자 리턴 코드 (계속)

유형	코드	설명
구성 파일	50	구성 파일 열기 오류
	51	데이터 저장소 유형이 STUB도 VSAM도 아님
	52	아웃바운드 웹 서비스 전환은 Y도 N도 아님
원격 웹 서비스	30	EXEC CICS INVOKE WEBSERVICE 명령은 INVREQ 조건을 리턴함
	31	EXEC CICS INVOKE WEBSERVICE 명령은 NOTFND 조건을 리턴함
	32	EXEC CICS INVOKE WEBSERVICE 명령은 INVREQ 또는 NOTFND 이외의 조건을 리턴함
애플리케이션	97	주문을 완료하기 위해 충분하지 않은 재고
	98	주문 수량이 양수가 아님
	99	DFH0XCMN은 CA-REQUEST-ID 필드가 01INQC, 01INQS 또는 01ORDR 중 하나로 설정되지 않는 COMMAREA를 수신함

INQUIRE CATALOG 조작

이 조작에서는 호출자가 지정하는 항목을 시작하여 최대 15개의 카탈로그 항목 목록을 리턴합니다.

입력 매개변수

CA-REQUEST-ID

조작을 식별하는 문자열. INQUIRE CATALOG 명령의 경우, 문자열에 01INQC가 포함됩니다.

CA-LIST-START-REF

리턴할 첫 번째 항목의 참조 번호

출력 매개변수

CA-RETURN-CODE

조작을 식별하는 문자열.

CA-RESPONSE-MESSAGE

num ITEMS RETURNED를 포함하여 사용자가 읽을 수 있는 문자열입니다. 여기서 *num*은 리턴한 항목 수입니다.

CA-LAST-ITEM-REF

리턴한 마지막 항목의 참조 번호

CA-ITEM-COUNT

리턴한 항목 수

CA-CAT-ITEM

리턴한 카탈로그 항목 목록이 들어 있는 배열. 배열에는 15개 요소가 있으며, 15개 이하의 요소가 리턴되면, 나머지 배열 요소에는 공백이 포함됩니다.

INQUIRE SINGLE ITEM 조작

이 조작은 호출자가 지정하는 단일 카탈로그 항목을 리턴합니다.

입력 매개변수

CA-REQUEST-ID

조작을 식별하는 문자열. INQUIRE SINGLE ITEM 명령의 경우, 문자열에 01INQS가 포함됩니다.

CA-ITEM-REF-REQ

리턴할 항목의 참조 번호

출력 매개변수

CA-RETURN-CODE

조작을 식별하는 문자열.

CA-RESPONSE-MESSAGE

RETURNED ITEM: REF=*item-reference*를 포함하는 사용자가 읽을 수 있는 문자열. 여기서 *item-reference*는 리턴된 항목의 참조 번호입니다.

CA-SINGLE-ITEM

리턴한 카탈로그 항목 중 첫 번째 요소에 들어 있는 배열

PLACE ORDER 조작

이 조작은 단일 항목에 대한 주문을 작성합니다. 필요 수량을 사용할 수 없는 경우, 사용자에게 메시지가 리턴됩니다. 주문이 완료되면, 주문한 항목과 수량이 재고 관리자에게 통보됩니다.

입력 매개변수

CA-REQUEST-ID

조작을 식별하는 문자열. PLACE ORDER 조작의 경우, 문자열에 01ORDR이 포함됩니다.

CA-USERID

애플리케이션이 디스패치 및 대금 청구에 사용하는 8자로 된 사용자 ID

CA-CHARGE-DEPT

애플리케이션이 디스패치 및 대금 청구에 사용하는 8자로 된 부서 ID

CA-ITEM-REF-NUMBER

주문할 항목의 참조 번호

CA-QUANTITY-REQ

필수 항목 수

출력 매개변수**CA-RETURN-CODE**

조작을 식별하는 문자열.

CA-RESPONSE-MESSAGE

ORDER SUCCESSFULLY PLACED를 포함하는 사용자가 읽을 수 있는 문자열.

DISPATCH STOCK 조작

이 조작에서 재고 디스패치 프로그램을 호출하면 차례로 고객에게 주문을 전달합니다.

입력 매개변수**CA-ORD-REQUEST-ID**

조작을 식별하는 문자열. DISPATCH ORDER 조작의 경우, 문자열에 01DSP0이 포함됩니다.

CA-ORD-USERSID

애플리케이션이 디스패치 및 대금 청구에 사용하는 8자로 된 사용자 ID

CA-ORD-CHARGE-DEPT

애플리케이션이 디스패치 및 대금 청구에 사용하는 8자로 된 부서 ID

CA-ORD-ITEM-REF-NUMBER

주문할 항목의 참조 번호

CA-ORD-QUANTITY-REQ

필수 항목 수

출력 매개변수**CA-ORD-RETURN-CODE**

조작을 식별하는 문자열.

NOTIFY STOCK MANAGER 조작

이 조작에서는 재고 보충이 필요한지 여부를 결정하기 위해 주문 세부내역을 작성합니다.

입력 매개변수**CA-ORD-REQUEST-ID**

조작을 식별하는 문자열. NOTIFY STOCK MANAGER 조작의 경우, 문자열에 01STK0이 포함됩니다.

CA-STK-ITEM-REF-NUMBER

주문할 항목의 참조 번호

CA-STK-QUANTITY-REQ

필수 항목 수

출력 매개변수

CA-ORD-RETURN-CODE

조작을 식별하는 문자열.

파일 구조 및 정의

예제 애플리케이션은 비축된 모든 항목과 재고 레벨에 대한 세부사항을 포함하는 카탈로그 파일과 애플리케이션에 대한 사용자 선택 옵션을 가지고 있는 구성 파일과 같은 두 개의 VSAM 파일을 사용합니다.

카탈로그 파일

카탈로그 파일은 제품 인벤토리와 관련되는 모든 정보가 포함된 KSDS VSAM 파일입니다.

카탈로그 파일 레코드

파일의 레코드에는 다음 구조가 있습니다.

이름	COBOL 데이터 유형	설명
WS-ITEM-REF-NUM	PIC 9(4)	항목 참조 번호
WS-DESCRIPTION	PIC X(40)	항목 설명
WS-DEPARTMENT	PIC 9(3)	부서 ID 번호
WS-COST	PIC ZZZ.99	항목 가격
WS-IN-STOCK	PIC 9(4)	재고에 있는 항목 수
WS-ON-ORDER	PIC 9(3)	주문한 항목 수

파일 구성

구성 파일은 예제 애플리케이션을 구성하기 위해 사용되는 정보를 포함하는 KSDS VSAM 파일입니다.

구성 파일 레코드

구성 파일은 네 개의 특정 레코드를 가진 KSDS VSAM 파일입니다.

표 32. 일반 정보 레코드

이름	COBOL 데이터 유형	설명
PROGS-KEY	PIC X(9)	EXMP-CONF를 포함하는 일반 정보 레코드에 대한 키 필드.
필러	PIC X	
DATASTORE	PIC X(4)	사용되는 데이터 저장소 프로그램 유형을 지정하는 문자열. 값은 다음과 같습니다. STUB VSAM
필러	PIC X	
DO-OUTBOUND-WS	PIC X	디스패치 관리자인지 아닌지를 지정하는 문자는 아웃바운드 웹 서비스 요청을 작성합니다. 값은 다음과 같습니다. Y N
필러	PIC X	
CATMAN-PROG	PIC X(8)	카탈로그 관리자 프로그램의 이름.
필러	PIC X	
DSSTUB-PROG	PIC X(8)	더미 데이터 핸들러 프로그램의 이름.
필러	PIC X	
DSVSAM-PROG	PIC X(8)	VSAM 데이터 핸들러 프로그램의 이름.
필러	PIC X	
ODSTUB-PROG	PIC X(8)	더미 주문 디스패치 모듈의 이름.
필러	PIC X	
ODWEBS-PROG	PIC X(8)	아웃바운드 웹 서비스 주문 디스패치 프로그램의 이름.
필러	PIC X	
STKMAN-PROG	PIC X(8)	재고 관리자 프로그램의 이름.
필러	PIC X(10)	

표 33. 아웃바운드 URL 레코드

이름	COBOL 데이터 유형	설명
URL-KEY	PIC X(9)	OUTBNDURL을 포함하는 일반 정보 레코드에 대한 키 필드.
필러	PIC X	

표 33. 아웃바운드 URL 레코드 (계속)

이름	COBOL 데이터 유형	설명
OUTBOUND-URL	PIC X(255)	주문 디스패처 웹 서비스 요청에 대한 아웃바운드 URL.

표 34. 카탈로그 파일 정보 레코드

이름	COBOL 데이터 유형	설명
URL-FILE-KEY	PIC X(9)	VSAM-NAME을 포함하는 일반 정보 레코드에 대한 키 필드.
필러	PIC X	
CATALOG-FILE-NAME	PIC X(8)	카탈로그 파일에 사용되는 CICS FILE 자원의 이름.

표 35. 서버 정보 레코드

이름	COBOL 데이터 유형	설명
WS-SERVER-KEY	PIC X(9)	WS-SERVER를 포함하는 서버 정보 레코드에 대한 키 필드.
필러	PIC X	
CATALOG-FILE-NAME	PIC X(8)	CICS 웹 서비스 클라이언트 전용 예제 애플리케이션이 웹 서비스로서 배치되는 시스템의 포트 및 IP 주소.

주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다. IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이선스까지 부여하는 것은 아닙니다. 라이선스에 대한 의문사항은 다음으로 문의하십시오.

150-945

서울특별시 영등포구 국제금융로 10, 3IFC,

한국 아이.비.엠 주식회사

대표전화서비스: 02-3781-7114

2바이트(DBCS) 정보에 관한 라이선스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan, Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다.

IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증 없이 이 책을 "현상태대로" 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 책에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

- (i) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함) 간의 정보 교환 및
- (ii) 교환된 정보의 상호 이용을 목적으로 정보를 원하는 프로그램 라이선스 사용자는 다음 주소로 문의하십시오.

150-945

서울특별시 영등포구 국제금융로 10, 3IFC,

한국 아이.비.엠 주식회사

대표전화서비스: 02-3781-7114

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등)하에서 사용될 수 있습니다.

이 정보에 기술된 라이선스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이선스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이선스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

개인정보 보호정책 고려사항

SaaS(Software as a Service) 솔루션을 포함한 IBM 소프트웨어 제품(이하 "소프트웨어 오퍼링")은 제품 사용 정보를 수집하거나 일반 사용자의 사용 경험을 개선하거나 일반 사용자와의 상호 작용을 조정하거나 그 외의 용도로 쿠키나 기타 다른 기술을 사용할 수 있습니다. 대부분의 경우 소프트웨어 오퍼링은 개인 식별 정보를 수집하지 않습니다. IBM 소프트웨어 오퍼링 중 일부는 개인 식별 정보를 수집할 수 있도록 도와 줍니다. 이 소프트웨어 오퍼링이 쿠키를 사용하여 개인 식별 정보를 수집하는 경우, 이 오퍼링의 쿠키 사용에 대한 자세한 정보는 다음과 같습니다.

CICSplex SM 웹 사용자 인터페이스:

WUI 기본 인터페이스: 전개된 구성에 따라 이 소프트웨어 오퍼링은 세션 관리, 인증, 사용자 용이성 향상 또는 기타 사용법 추적이나 기능을 위해 각 사용자의 사용자 이름 및 개인 식별 정보를 수집하는 세션 및 지속적 쿠키를 사용할 수 있습니다. 이러한 쿠키는 사용 안함으로 설정할 수 없습니다.

WUI 데이터 인터페이스: 전개된 구성에 따라 이 소프트웨어 오퍼링은 세션 관리, 인증 또는 기타 사용법 추적이나 기능을 위해 각 사용자의 사용자 이름 및 개인 식별 정보를 수집하는 세션 쿠키를 사용할 수 있습니다. 이러한 쿠키는 사용 안함으로 설정할 수 없습니다.

WUI Hello World 페이지: 전개된 구성에 따라 이 소프트웨어 오퍼링은 개인 식별 정보를 수집하지 않는 세션 쿠키를 사용할 수 있습니다. 이러한 쿠키는 사용 안함으로 설정할 수 없습니다.

CICS 탐색기: 전개된 구성에 따라 이 소프트웨어 오퍼링은 세션 관리, 인증 및 싱글 사인온 구성을 위해 사용자의 사용자 이름 및 비밀번호를 수집하는 세션 및 지속적 환경 설정을 사용할 수 있습니다. 사인온 중에 선택란을 선택하여 사용자 비밀번호를 암

호화된 양식으로 저장하면 사용자의 명시적인 조치에 의해서만 사용으로 설정할 수 있지만 이러한 환경 설정은 사용 안함으로 설정할 수 없습니다.

이 소프트웨어 오퍼링을 위해 전개되는 구성이 귀하(고객)에게 쿠키 등의 기술로 일반 사용자의 개인 식별 정보를 수집하는 기능을 제공하는 경우, 통지 및 동의에 필요한 모든 요구사항을 포함하여 그러한 데이터 수집에 적용되는 모든 법률에 대해 직접 법적 자문을 구해야 합니다.

이러한 목적을 위해 쿠키를 비롯한 여러 기술을 사용하는 데 대한 자세한 정보는 IBM의 개인정보 보호정책(<http://www.ibm.com/privacy/kr/ko/>), IBM의 온라인 개인정보 보호정책(<http://www.ibm.com/privacy/details/kr/ko/>) 『쿠키, 웹 비콘 및 기타 기술』절과 『IBM 소프트웨어 제품 및 SaaS(Software-as-a-Service) 개인정보 보호정책』(<http://www-01.ibm.com/software/info/product-privacy/>)을 참조하십시오.

상표

IBM, IBM 로고 및 [ibm.com](http://www.ibm.com)[®]은 전세계 여러 국가에 등록된 International Business Machines Corp.의 상표 또는 등록상표입니다. 기타 제품 및 서비스 이름은 IBM 또는 타사의 상표입니다. IBM 상표의 최신 목록은 웹 사이트의 저작권 및 상표 정보(www.ibm.com/legal/copytrade.shtml)에서 확인할 수 있습니다.

Java 및 모든 Java 기반 상표와 로고는 Oracle 및/또는 그 계열사의 상표 또는 등록상표입니다.

Linux는 미국 또는 기타 국가에서 사용되는 Linus Torvalds의 등록상표입니다.

Microsoft 및 Windows는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

UNIX는 미국 또는 기타 국가에서 사용되는 Open Group의 등록상표입니다.

참고문헌

z/OS용 CICS Transaction Server의 CICS 서적

일반

z/OS용 CICS Transaction Server 프로그램 디렉토리 - 기본, GI13-3375
z/OS용 CICS Transaction Server 프로그램 디렉토리 활성화 모듈 - 기본, GI13-3376
z/OS용 CICS Transaction Server 프로그램 디렉토리 활성화 모듈 - 개발자 평가판, GI13-3377
z/OS용 CICS Transaction Server 프로그램 디렉토리 활성화 모듈 - Value Unit Edition, GI13-3378
z/OS용 CICS Transaction Server 새로운 기능, GC34-7437
z/OS용 CICS Transaction Server CICS TS 버전 5.3으로 업그레이드, GC34-7436
z/OS용 CICS Transaction Server 설치 안내서, GC34-7414

CICS 액세스

CICS 인터넷 안내서, SC34-7416
CICS 웹 서비스 안내서, SC34-7452

관리

CICS 시스템 정의 안내서, SC34-7428
CICS 사용자 정의 안내서, SC34-7404
CICS 자원 정의 안내서, SC34-7425
CICS 운영 및 유틸리티 안내서, SC34-7420
CICS RACF 보안 안내서, SC34-7423
CICS 제공된 트랜잭션, SC34-7427

프로그래밍

CICS 애플리케이션 프로그래밍 안내서, SC34-7401
CICS 애플리케이션 프로그래밍 참조서, SC34-7402
CICS 시스템 프로그래밍 참조서, SC34-7429
CICS FEPI(Front End Programming Interface) 사용자 안내서, SC34-7412
CICS C++ OO 클래스 라이브러리, SC34-7405
CICS 분산 트랜잭션 프로그래밍 안내서, SC34-7410
CICS 비즈니스 트랜잭션 서비스, SC34-7403
CICS의 Java 애플리케이션, SC34-7417

진단

CICS 문제점 판별 안내서, GC34-7422
CICS 성능 안내서, SC34-7421
CICS 메시지 및 코드 제 1권, GC34-7418
CICS 메시지 및 코드 제 2권, GC34-7419
CICS 진단 참조서, GC34-7409
CICS 복구 및 다시 시작 안내서, SC34-7424
CICS 데이터 영역, GC34-7406
CICS 추적 항목, SC34-7430
CICS 디버깅 도구 인터페이스 참조서, GC34-7408

통신

CICS 상호 통신 안내서, SC34-7415
CICS 외부 인터페이스 안내서, SC34-7411

데이터베이스

CICS DB2[®] 안내서, SC34-7407
CICS IMS[™] 데이터베이스 제어 안내서, SC34-7413
CICS 공유 데이터 테이블 안내서, SC34-7426

z/OS용 *CICS* Transaction Server에 대한 *CICS*Plex SM 서적

일반

*CICS*Plex SM 개념 및 계획, SC34-7441
*CICS*Plex SM 웹 사용자 인터페이스 안내서, SC34-7451

관리(Administration and Management)

*CICS*Plex SM 관리, SC34-7438
*CICS*Plex SM 운영 보기 참조서, SC34-7447
*CICS*Plex SM 모니터 보기 참조서, SC34-7446
*CICS*Plex SM 워크로드 관리, SC34-7444
*CICS*Plex SM 자원 사용 관리, SC34-7443
*CICS*Plex SM 비즈니스 애플리케이션 관리, SC34-7442

프로그래밍

*CICS*Plex SM 애플리케이션 프로그래밍 안내서, SC34-7439
*CICS*Plex SM 애플리케이션 프로그래밍 참조서, SC34-7440

진단

*CICS*Plex SM 자원 테이블 참조서 제 1권, SC34-7449
*CICS*Plex SM 자원 테이블 참조서 제 2권, SC34-7450

CICSplex SM 메시지 및 코드, GC34-7445

CICSplex SM 문제점 판별, GC34-7448

기타 CICS 서적

다음 서적에는 CICS에 대한 자세한 정보가 포함되어 있으나 z/OS용 CICS Transaction Server, 버전 5 릴리스 3 과 함께 제공되지는 않습니다.

Designing and Programming CICS Applications, SR23-9692

CICS Application Migration Aid Guide, SC33-0768

CICS 제품군: API Structure, SC33-1007

CICS 제품군: Client/Server Programming, SC33-1435

CICS 제품군: Interproduct Communication, SC34-6853

CICS 제품군: Communicating from CICS on System/390, SC34-6854

z/OS용 CICS Transaction Gateway 관리, SC34-5528

CICS 제품군: 일반 정보, GC33-0155

CICS 4.1 Sample Applications Guide, SC33-1173

CICS/ESA 3.3 XRF Guide, SC33-0661

내게 필요한 옵션

내게 필요한 옵션 기능은 거동 장애나 시각 장애 같은 신체적 장애가 있는 사용자가 소프트웨어 제품을 사용할 수 있도록 도와줍니다.

다음 중 한 가지 방법으로 CICS 시스템을 설정, 실행, 유지보수하는 데 필요한 대부분의 작업을 수행할 수 있습니다.

- CICS에 로그인된 3270 에뮬레이터 사용
- TSO에 로그인된 3270 에뮬레이터 사용
- 3270 에뮬레이터를 MVS 시스템 콘솔로 사용

IBM Personal Communications는 장애를 가진 사람들을 위한 내게 필요한 옵션 기능을 3270 에뮬레이션에 제공합니다. 이 제품을 사용하여 CICS 시스템에서 필요한 내게 필요한 옵션 기능을 제공할 수 있습니다.

색인

[가]

가변 배열 132, 509
결합, SOAP 15
고급 언어 구조
 JSON으로 변환 181, 560
 WSDL로 변환 63, 441
구문 표기법 96, 474
구성 265
구성 파일, 파이프라인 269
글로벌 사용자 종료 385, 617
기본 EPR 421, 658
 WSDL 1.1 421, 658

[나]

네임스페이스
 파이프라인 구성 요소 287

[다]

다이어그램
 구문 96, 474
동적 라우팅
 서비스 제공자 344
 터미널 핸들러 344

[라]

런타임 시 제한사항 382, 614

[마]

메시지 교환 패턴(MEP) 36
메시지 핸들러
 비터미널 334, 335, 336
 신뢰 클라이언트 호출 697
문제점 진단
 서비스 요청자 708
 서비스 제공자 706

[바]

반복 콘텐츠 147, 525
보안 컨테이너 372
 DFHSAML-AnnnVmmm 374
 DFHSAML-ASSQNAME 374
 DFHSAML-ATTRAnnn 375
 DFHSAML-ATTRFnnn 375
 DFHSAML-ATTRNnnn 375
 DFHSAML-ATTRSnnn 375
 DFHSAML-ATTRYnnn 375
 DFHSAML-AUDNRnnn 375
 DFHSAML-AUTHMETH 376
 DFHSAML-CERTIDN 376
 DFHSAML-CERTSDN 376
 DFHSAML-CERTSNUM 376
 DFHSAML-CONFMETH 376
 DFHSAML-COUNTS 376
 DFHSAML-FLAGS 376
 DFHSAML-ISSUER 376
 DFHSAML-NAMID 377
 DFHSAML-NAMIDF 377
 DFHSAML-NAMIDQ 377
 DFHSAML-NAMIDSP 377
 DFHSAML-NAMIDSPQ 377
 DFHSAML-OUTTOKEN 377
 DFHSAML-PROXYnnn 377
 DFHSAML-RESPONSE 377
 DFHSAML-SAMLID 377
 DFHSAML-SUBJADDR 378
 DFHSAML-SUBJDNS 378
 DFHSAML-TIMES 378
보안 핸들러
 자체 작성 696
본문, SOAP 15
비터미널 메시지 핸들러 334, 335, 336

[사]
사용자 정의 보안 핸들러 696
사용자 컨테이너 379
상표 767
서비스
 파이프라인 구성 요소 303

서비스 매개변수 목록
 <service_parameter_list> 305
서비스 요청자
 문제점 진단 708
 파이프라인 정의 280
서비스 요청자 애플리케이션
 원자적 트랜잭션 사용 397, 631
서비스 제공자
 문제점 진단 706
서비스 제공자 애플리케이션 217, 597
 데이터 구조에서 작성 154, 219, 532, 599
 원자적 트랜잭션 사용 396, 630
스키마
 채널 설명 158, 221, 535, 601
신뢰 클라이언트
 인터페이스 682
 호출 697
신뢰 클라이언트 호출 697
실행 시간 제한사항 382, 614

[아]

알고리즘 683, 685
애플리케이션 핸들러
 파이프라인 구성 요소 281
언어 구조
 JSON으로 변환 181, 560
 WSDL로 변환 63, 441
엔드포인트 참조
 기본값 421, 658
엔드포인트 참조(EPR) 413, 650
엔벨로프, SOAP 15
워크로드 관리
 서비스 제공자 344
 터미널 핸들러 344
원자적 트랜잭션 391, 399, 625, 633
 등록 서비스 392, 625
 상태 400, 634
서비스 요청자 구성 397, 631
서비스 제공자 구성 396, 630
CICS 구성 394, 628

웹 서비스

구성

Kerberos 436

웹 서비스 발견 235

웹 서비스 보안

파이프라인 구성 313

웹 서비스 보안(WSS) 675, 687, 692

웹 서비스 보안: SOAP 메시지 보안 42

웹 서비스 오류 706, 708

웹 서비스 지원 62, 440

서비스 제공자 애플리케이션 작성 154,
219, 532, 599

웹 서비스를 위한 보안 675

유틸리티 프로그램

웹 서비스 지원 62, 440

JSON 지원 180, 560

인증

파이프라인 구성 요소 316

일괄처리 유틸리티 180, 560

웹 서비스 지원 62, 440

[자]

제어 컨테이너 347

주소 지정

파이프라인 구성 요소 287

지속적 메시지 242

지속적 메시지 지원 243

지원, JSON 180, 560

지원, 웹 서비스 62, 440

직접 모드 403, 639

[차]

채널 설명 158, 221, 535, 601

채널 인터페이스 158, 221, 535, 601

[카]

카탈로그 샘플카탈로그 샘플 717

컨테이너

제어 컨테이너

DFHERROR 347

DFHFUNCTION 348

DFHHTTPSTATUS 351

DFHMEDIATYPE 351

DFHNORESPONSE 352

DFHREQUEST 352

컨테이너 (계속)

제어 컨테이너 (계속)

DFHRESPONSE 352

DFHWS-CCSID 353

채널 설명 158, 221, 535, 601

컨텍스트 컨테이너

DFHWS-APPHANDLER 357, 358,
360

DFHWS-DATA 358

DFHWS-FAULT 358

DFHWS-PIPELINE 360

DFHWS-SOAPLEVEL 361

DFHWS-STSREASON 373

DFHWS-TRANID 361

DFHWS-URI 361

DFHWS-USERID 366

DFHWS-WEBSERVICE 366

DFH-HANDLERPLIST 357

DFH-SERVICEPLIST 357

파이프라인에서 사용 345

DFHERROR 347

DFHFUNCTION 348

DFHHTTPSTATUS 351

DFHMEDIATYPE 351

DFHNORESPONSE 352

DFHREQUEST 352

DFHRESPONSE 352

DFHSAML-AnnnVmmm 374

DFHSAML-ASSQNAME 374

DFHSAML-ATTRAnnn 375

DFHSAML-ATTRFnnn 375

DFHSAML-ATTRNnnn 375

DFHSAML-ATTRSnnn 375

DFHSAML-ATTRYnnn 375

DFHSAML-AUDNRnnn 375

DFHSAML-AUTHMETH 376

DFHSAML-CERTIDN 376

DFHSAML-CERTSDN 376

DFHSAML-CERTSNUM 376

DFHSAML-CONFMETH 376

DFHSAML-COUNTS 376

DFHSAML-FLAGS 376

DFHSAML-ISSUER 376

DFHSAML-NAMID 377

DFHSAML-NAMIDF 377

DFHSAML-NAMIDQ 377

DFHSAML-NAMIDSP 377

DFHSAML-NAMIDSPQ 377

컨테이너 (계속)

DFHSAML-OUTTOKEN 377

DFHSAML-PROXYnnn 377

DFHSAML-RESPONSE 377

DFHSAML-SAMLID 377

DFHSAML-SUBJADDR 378

DFHSAML-SUBJDNS 378

DFHSAML-TIMES 378

DFHWS-APPHANDLER 357, 358, 360

DFHWS-CCSID 353

DFHWS-CID-DOMAIN 367

DFHWS-DATA 358

DFHWS-FAULT 358

DFHWS-IDTOKEN 372

DFHWS-LOCATION 359

DFHWS-MEP 359

DFHWS-MTOM-IN 367

DFHWS-MTOM-OUT 368

DFHWS-PIPELINE 360

DFHWS-RESPWAIT 360

DFHWS-RESTOKEN 372

DFHWS-SERVICEURI 373

DFHWS-SOAPLEVEL 361

DFHWS-STSACTION 373

DFHWS-STSFAULT 373

DFHWS-STSREASON 373

DFHWS-STSURI 374

DFHWS-TOKENTYPE 374

DFHWS-TRANID 361

DFHWS-URI 361

DFHWS-USERID 366

DFHWS-WEBSERVICE 366

DFHWS-XOP-IN 369

DFHWS-XOP-OUT 369

DFH-EXIT-HEADER1 356

DFH-HANDLERPLIST 357

DFH-SERVICEPLIST 357

SAML 374

컨테이너 DFHSAML-AnnnVmmm 374

컨테이너 DFHSAML-ASSQNAME 374

컨테이너 DFHSAML-ATTRAnnn 375

컨테이너 DFHSAML-ATTRFnnn 375

컨테이너 DFHSAML-ATTRNnnn 375

컨테이너 DFHSAML-ATTRSnnn 375

컨테이너 DFHSAML-ATTRYnnn 375

컨테이너 DFHSAML-AUDNRnnn 375

컨테이너 DFHSAML-AUTHMETH 376

컨테이너 DFHSAML-CERTIDN 376

컨테이너 DFHSAML-CERTSDN 376
 컨테이너 DFHSAML-CERTSNUM 376
 컨테이너 DFHSAML-CONFMETH 376
 컨테이너 DFHSAML-COUNTS 376
 컨테이너 DFHSAML-FLAGS 376
 컨테이너 DFHSAML-ISSUER 376
 컨테이너 DFHSAML-NAMID 377
 컨테이너 DFHSAML-NAMIDF 377
 컨테이너 DFHSAML-NAMIDQ 377
 컨테이너 DFHSAML-NAMIDSP 377
 컨테이너 DFHSAML-NAMIDSPQ 377
 컨테이너 DFHSAML-OUTTOKEN 377
 컨테이너 DFHSAML-PROXYnnn 377
 컨테이너 DFHSAML-RESPONSE 377
 컨테이너 DFHSAML-SAMLID 377
 컨테이너 DFHSAML-SUBJADDR 378
 컨테이너 DFHSAML-SUBJDNS 378
 컨테이너 DFHSAML-TIMES 378
 컨테이너 DFHWS-CID-DOMAIN 367
 컨테이너 DFHWS-IDTOKEN 372
 컨테이너 DFHWS-LOCATION 359
 컨테이너 DFHWS-MEP 359
 컨테이너 DFHWS-MTOM-IN 367
 컨테이너 DFHWS-MTOM-OUT 368
 컨테이너 DFHWS-RESPWAIT 360
 컨테이너 DFHWS-RESTOKEN 372
 컨테이너 DFHWS-SERVICEURI 373
 컨테이너 DFHWS-STSACTION 373
 컨테이너 DFHWS-STSFault 373
 컨테이너 DFHWS-STSURI 374
 컨테이너 DFHWS-TOKENTYPE 374
 컨테이너 DFHWS-WSDL-CTX 369
 컨테이너 DFHWS-XOP-IN 369
 컨테이너 DFHWS-XOP-OUT 369
 컨테이너 DFH-EXIT-HEADER1 356
 컨텍스트 컨테이너 356
 DFHWS-CID-DOMAIN 367
 DFHWS-IDTOKEN 372
 DFHWS-LOCATION 359
 DFHWS-MEP 359
 DFHWS-MTOM-IN 367
 DFHWS-MTOM-OUT 368
 DFHWS-RESPWAIT 360
 DFHWS-RESTOKEN 372
 DFHWS-SERVICEURI 373
 DFHWS-STSACTION 373
 DFHWS-STSFault 373
 DFHWS-STSURI 374

컨텍스트 컨테이너 (계속)

DFHWS-TOKENTYPE 374
 DFHWS-WSDL-CTX 369
 DFHWS-XOP-IN 369
 DFHWS-XOP-OUT 369
 DFH-EXIT-HEADER1 356

컨텐츠 유형 vii

컨텐츠 유형 맵핑 vii

[파]

파이프라인 구성 692

웹 서비스 보안 313

MTOM/XOP 306

파이프라인 구성 요소

<addressing> 287
 <apphandler> 283
 <auth_token_type> 323
 <cics_json_handler_java> 288
 <cics_mtom_handler> 307
 <cics_soap_1.1_handler> 288
 <cics_soap_1.1_handler_java> 291
 <cics_soap_1.2_handler> 294
 <cics_soap_1.2_handler_java> 296
 <default_http_transport_handler_list> 299
 <default_mq_transport_handler_list> 299
 <default_transport_handler_list> 300
 <dfhmtom_configuration> 308
 <dfhwsse_configuration> 314
 <encrypt_body> 326
 <handler> 301
 <jvmserver> 302
 <mime_options> 312
 <mtom> 307
 <mtom_options> 309
 <named_transport_entry> 282
 <namespace> 287
 <provider_pipeline> 283
 <provider_pipeline_json> 284
 <repository> 302
 <requester_pipeline> 286
 <service> 303
 <service_handler_list> 303
 <sign_body> 325
 <sts_authentication> 320
 <sts_endpoint> 324

파이프라인 구성 요소 (계속)

<terminal_handler> 285
 <transport> 305
 <transport_handler_list> 286
 <wsse_handler> 313
 <xop_options> 311
 <인증> 316

파이프라인 구성 파일 269

파이프라인 정의

서비스 요청자 280

파이프라인 처리

사용자 정의 385, 617

URI 대체 388, 620

파이프라인 처리 사용자 정의 385, 617

표기법

구문 96, 474

표준

SAML 39

[하]

핸들러

파이프라인 구성 요소 301

헤더, SOAP 15

호환 모드 403, 639

[숫자]

2진 첨부 파일

파이프라인 구성 306

A

apphandler

파이프라인 구성 요소 283

auth_token_type

파이프라인 구성 요소 323

Axis2 249

C

C 및 C++

XML 스키마로 맵핑 115, 118, 493, 496

C 및 C++에 대한 맵핑 115, 118, 493, 496

cics_json_handler_java

파이프라인 구성 요소 288

cics_mtom_handler

파이프라인 구성 요소 307

cics_soap_1.1_handler
 파이프라인 구성 요소 288

cics_soap_1.1_handler_java
 파이프라인 구성 요소 291

cics_soap_1.2_handler
 파이프라인 구성 요소 294

cics_soap_1.2_handler_java
 파이프라인 구성 요소 296

COBOL
 가변 반복 컨테츠 147, 525
 XML 스키마로 맵핑 105, 109, 483, 487
COBOL로 맵핑 105, 109, 483, 487

D

default_http_transport_handler_list
 파이프라인 구성 요소 299, 302

default_mq_transport_handler_list
 파이프라인 구성 요소 299

default_target
 파이프라인 구성 요소 305

default_transport_handler_list
 파이프라인 구성 요소 300

DFHERROR 컨테이너 347

DFHFHFUNCTION 컨테이너 348

DFHHTTPSTATUS 컨테이너 351

DFHJS2LS
 카탈로그된 프로시저 192, 204, 572, 584

DFHLS2JS
 카탈로그된 프로시저 181, 560

DFHLS2WS
 카탈로그된 프로시저 63, 441

DFHMEDIATYPE 컨테이너 351

dfhmtom_configuration
 파이프라인 구성 요소 308

DFHNORESPONSE 컨테이너 352

DFHREQUEST 컨테이너 352

DFHRESPONSE 컨테이너 352

DFHWS2LS
 카탈로그된 프로시저 79, 457

dfhwsse_configuration
 파이프라인 구성 요소 314

DFHWS-APPHANDLER 컨테이너 357,
 358, 360

DFHWS-CCSID 컨테이너 353

DFHWS-DATA 컨테이너 358

DFHWS-FAULT 컨테이너 358

DFHWS-PIPELINE 컨테이너 360

DFHWS-SOAPLEVEL 컨테이너 361

DFHWS-STSREASON 컨테이너 373

DFHWS-TRANID 컨테이너 361

DFHWS-URI 컨테이너 361

DFHWS-USERID 컨테이너 366

DFHWS-WEBSERVICE 컨테이너 366

DFH-HANDLERPLIST 컨테이너 357

DFH-SERVICEPLIST 컨테이너 357

E

encrypt_body
 파이프라인 구성 요소 326

EXEC CICS SOAPFAULT CREATE 명령
 162, 539

G

GLUE 385, 617

I

Information Center vii

Information Center 컨테츠 유형 vii

J

Java 249
 웹 서비스 호출 176, 554

Java 기반 SOAP 파이프라인 416, 653

Java에서 웹 서비스 호출 176, 554

JSON 265, 365

JSON 스키마 217, 597

JSON 지원 180, 560

JVM 서버 176, 249, 554

K

Kerberos
 웹 서비스 구성 436

M

maxOccurs
 XML 스키마 132, 509

MEP 36

MIME 메시지
 파이프라인 구성 306

mime_options
 파이프라인 구성 요소 312

minOccurs
 XML 스키마 132, 509

mtom
 파이프라인 구성 요소 307

MTOM/XOP
 파이프라인 구성 306

mtom_options
 파이프라인 구성 요소 309

N

named_transport_entry
 파이프라인 구성 요소 282

P

PL/I
 XML 스키마로 맵핑 122, 126, 499, 503

PL/I로 맵핑 122, 126, 499, 503

provider_pipeline
 파이프라인 구성 요소 283

R

RACF 구성 687

requester_pipeline
 파이프라인 구성 요소 286
 파이프라인 정의의 요소 280

S

SAML
 컨테이너 374

service_handler_list
 파이프라인 구성 요소 303

service_parameter_list
 서비스 매개변수 목록 305

sign_body
 파이프라인 구성 요소 325

SOAP
 개요 15
 결함 15
 본문 15
 엔벨로프 15
 헤더 15
 SOAP 개요 15

SOAP 결합 162, 539

SOAP 메시지

구조 15

서명 683

암호화 685

예제 15

XML 스키마

SOAP 메시지 유효성 검증 177, 555

XML 스키마에 대한 유효성 검증 177,
555

SOAP 메시지 경로 23

SOAP 메시지 보안 42

SOAP 메시지 유효성 검증 177, 555

SOAP 파이프라인 249

STS(Security Token Service)

신뢰 클라이언트 인터페이스 682

sts_authentication

파이프라인 구성 요소 320

sts_endpoint

파이프라인 구성 요소 324

T

terminal_handler

파이프라인 구성 요소 285

transport

파이프라인 구성 요소 305

transport_handler_list

파이프라인 구성 요소 286

U

URI

WebSphere MQ 전송 241

URI 대체 388, 620

W

Web Services Addressing

기본 조치 420, 423, 424, 657, 660,
662

기본 EPR 420, 421, 657, 658

명시적 조치 420, 422, 657, 659

스펙 412, 649

요청자 서비스 420, 657

요청자 파이프라인 구성 416, 653

제공자 파이프라인 구성 418, 655

주소 지정 핸들러 416, 418, 653, 655

Web Services Addressing (계속)

지원 412, 649

DFHWSADH 416, 418, 653, 655

DFHWS-URI 413, 650

EPR 413, 650

MAP 413, 650

WSDL 1.1 423, 660

WSDL 2.0 424, 662

<wsa:Action> 422, 659

WSDL

및 애플리케이션 데이터 구조 34

언어 구조로 변환 79, 192, 204, 457,
572, 584

조회 235

Web Services Addressing 420, 657

WSDL 1.1

기본 EPR 421, 658

WSDL 스펙 42

wsse_handler

파이프라인 구성 요소 313

WSS: SOAP 메시지 보안 42

WS-Addressing

기본 조치 423, 424, 660, 662

기본 EPR 421, 658

명시적 조치 422, 659

스펙 412, 649

요청자 파이프라인 구성 416, 653

제공자 파이프라인 구성 418, 655

주소 지정 핸들러 416, 418, 653, 655

DFHWSADH 416, 418, 653, 655

DFHWS-URI 413, 650

EPR 413, 650

MAP 413, 650

WSDL 1.1 423, 660

WSDL 2.0 424, 662

<wsa:Action> 422, 659

WS-AT 391, 625

X

XML 스키마 105, 109, 115, 118, 122,
126, 483, 487, 493, 496, 499, 503

xop_options

파이프라인 구성 요소 311

Z

zAAP 249

zosConnectService 265

z/OS Connect 265, 268
구성 263

[특수 문자]

<addressing>

파이프라인 구성 요소 287

<apphandler>

파이프라인 구성 요소 281, 283

<apphandler_class>

파이프라인 구성 요소 281

<auth_token_type>

파이프라인 구성 요소 323

<cics_json_handler_java>

파이프라인 구성 요소 288

<cics_mtom_handler>

파이프라인 구성 요소 307

<cics_soap_1.1_handler>

파이프라인 구성 요소 288

<cics_soap_1.1_handler_java>

파이프라인 구성 요소 291

<cics_soap_1.2_handler>

파이프라인 구성 요소 294

<cics_soap_1.2_handler_java>

파이프라인 구성 요소 296

<default_http_transport_handler_list>

파이프라인 구성 요소 299

<default_mq_transport_handler_list>

파이프라인 구성 요소 299

<default_target>

파이프라인 구성 요소 305

<default_transport_handler_list>

파이프라인 구성 요소 300

<dfhmtom_configuration>

파이프라인 구성 요소 308

<dfhwsse_configuration>

파이프라인 구성 요소 314

<encrypt_body>

파이프라인 구성 요소 326

<handler>

파이프라인 구성 요소 301

<jvmserver>

파이프라인 구성 요소 302

<mime_options>	
파이프라인 구성 요소	312
<mtom>	
파이프라인 구성 요소	307
<mtom_options>	
파이프라인 구성 요소	309
<named_transport_entry>	
파이프라인 구성 요소	282
<namespace>	
파이프라인 구성 요소	287
<provider_pipeline>	
파이프라인 구성 요소	283
<provider_pipeline_json>	
파이프라인 구성 요소	284
<repository>	
파이프라인 구성 요소	302
<requester_pipeline>	
파이프라인 구성 요소	286
<service>	
파이프라인 구성 요소	303
<service_handler_list>	
파이프라인 구성 요소	303
<service_parameter_list>	
파이프라인 구성 요소	305
<sign_body>	
파이프라인 구성 요소	325
<sts_authentication>	
파이프라인 구성 요소	320
<sts_endpoint>	
파이프라인 구성 요소	324
<terminal_handler>	
파이프라인 구성 요소	285
<transport>	
파이프라인 구성 요소	305
<transport_handler_list>	
파이프라인 구성 요소	286
<wsse_handler>	
파이프라인 구성 요소	313
<xop_options>	
파이프라인 구성 요소	311
<인증>	
파이프라인 구성 요소	316



SC43-2760-00

