IBM
Washington
Systems
Center

Web Services in CICS

IBM Washington Systems Center

Steve Fowlkes – fowlkes@us.ibm.com

Leigh Compton – lcompton@us.ibm.com

IBM
Washington
Systems
Center

Abstract

- This topic discusses CICS's Web services support. The topic starts with the flow of Web service through CICS and the resource definitions needed to implement Web services. Application develop of Web services is discussed both using the CICS Web Services Assistant and IBM Developer for z System (IDz). We will also touch on special processing via Handlers.

- Closely related to web services is CICS's ability to transform XML, and JSON web services.

- This presentation has been updated for CICS TS V5.3 and IDz V14.

IBM
Washington
Systems
Center

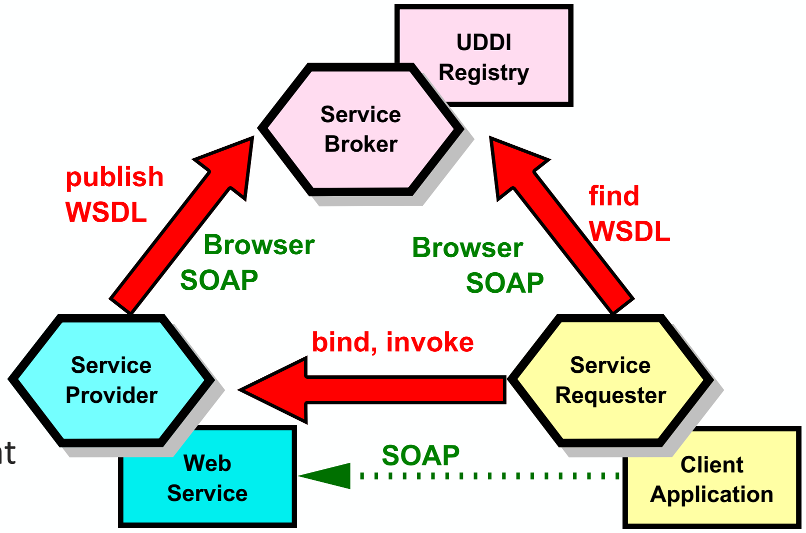
Agenda

- High Level Overview of SOAP, Web Services, and CICS Pipeline processing
- Development Styles
- Web Services Artifacts
  - WSDL
  - WSBind file
- CICS Web Services Assistant
  - DFHLS2WS / DFHWS2LS
- IBM Developer for z System (IDz)
  - Web Services, JSON Service, and XML transforms
- Service Flow Feature (just a few words)
- Java-based Web Services

IBM Washington Systems Center

Web Services

- Architecture for
  - Application to application
    - Communication
    - Interoperation
- Definition:
  - Web Services are **software components described via WSDL** that are capable of being accessed via **standard** network protocols such as SOAP over HTTP
- WS-I.org (Web Services Interoperability Organization)
  - Ensure interoperability

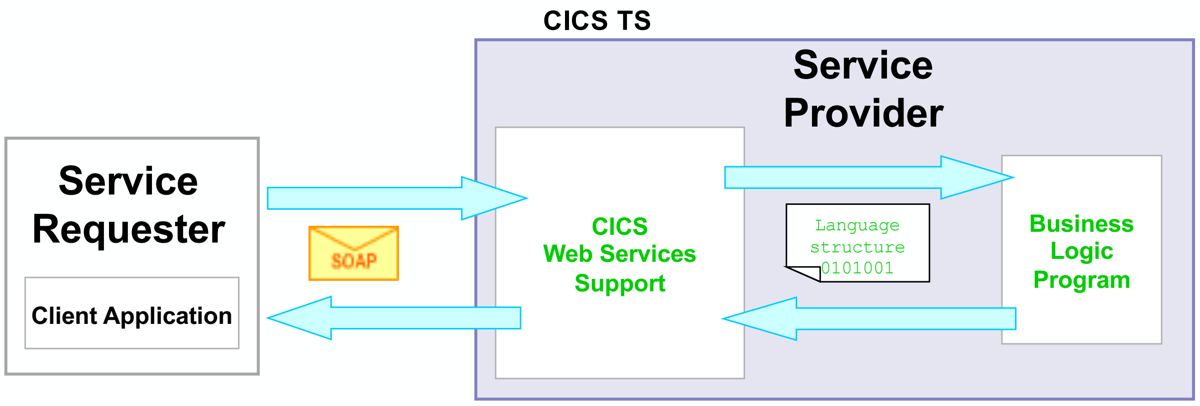


WS-I WEB SERVICES INTEROPERABILITY ORGANIZATION

uddi.org

The entire industry is agreeing on one set of standards !!

Very High Level: CICS Web Services



SOAP Message - XML, tag delimited data, one body, zero or more headers

zero or more headers	body containing application data

Languages Structure – e.g. COBOL copybook

```
01 DFHCOMMAREA.
        03  CUSTOMER-FIRST-NAME  PIC X(30).
        03  CUSTOMER-LAST-NAME   PIC X(30).
        …
```

Web Services Processing in CICS

- We would like for CICS to …
  - Listen for and receive incoming requests
  - Decide what to do with the request (invoke program)
  - Allow user written programs to look or massage the message before it reaches our application program
  - Allow user written programs to process SOAP headers
  - Deal with WS-Security and WS-Atomic Transaction standards
  - Parse the user payload, perform conversion, and prepare a COMMAREA or container
  - Invoke our application business logic program

IBM
Washington
Systems
Center

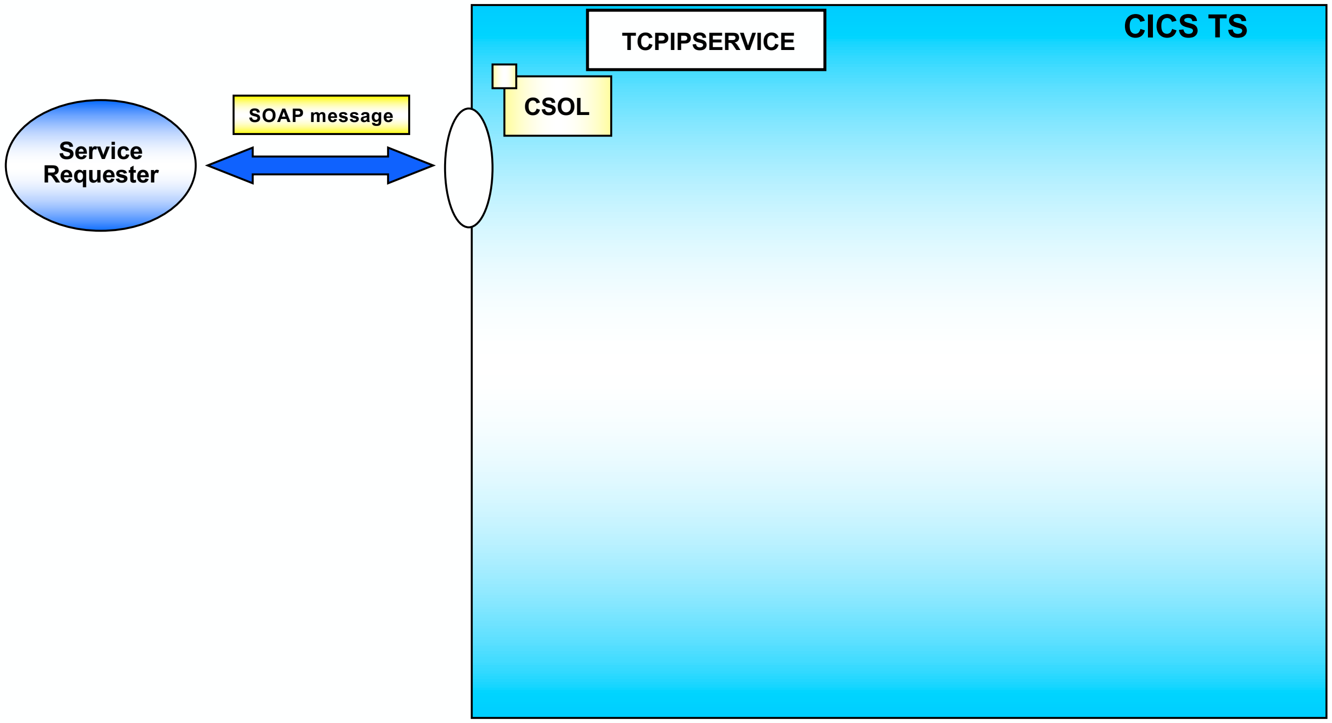
The URL - Directing Web Service Requests to CICS

- The ‘endpoint’ is the target of the Web Service request
- Specified in WSDL
- Contains a scheme
- Host name or address
- Port (optional)
- Path

```
http://demomvs.demopkg.ibm.com:8091/account
scheme://host:port/path
```

IBM
Washington
Systems
Center

CICS listens for HTTP requests (1 of 7)



- TCPIPSERVICE resource establishes listener task for incoming requests

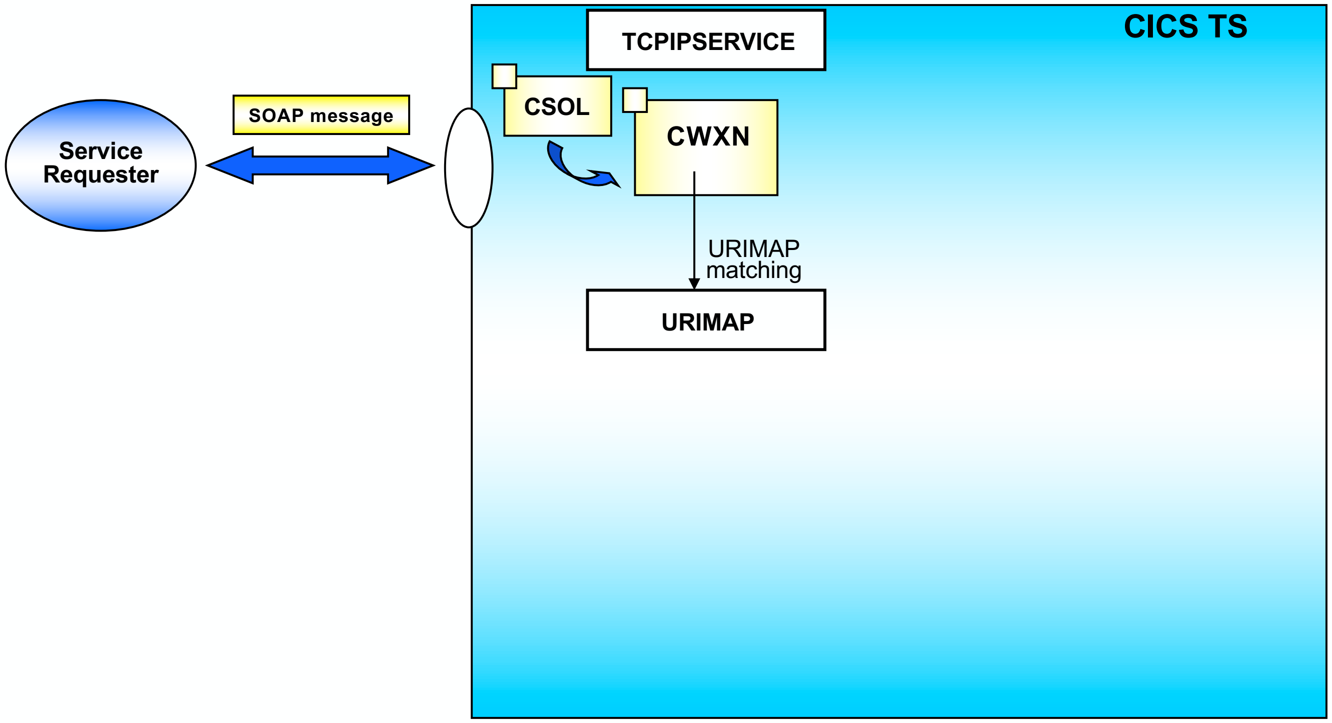
IBM
Washington
Systems
Center

TCPIPSERVICE

- TCPIPService definition tells CICS to listen for connections
- TCPIPSERVICE definition created by Systems Programmer
  - Tells CICS what port to listen on
  - Identifies host name or address if z/OS is multi-homed
  - Specifies whether encryption should be used (SSL/TLS)
    - And whether CICS or AT-TLS will be responsible for handling SSL/TLS protocol
- Receiving message is performed under the CSOL transaction

IBM
Washington
Systems
Center

CICS decides what to do with the request (2 of 7)



- CICS finds a match on endpoint path based on URIMAP resource definitions

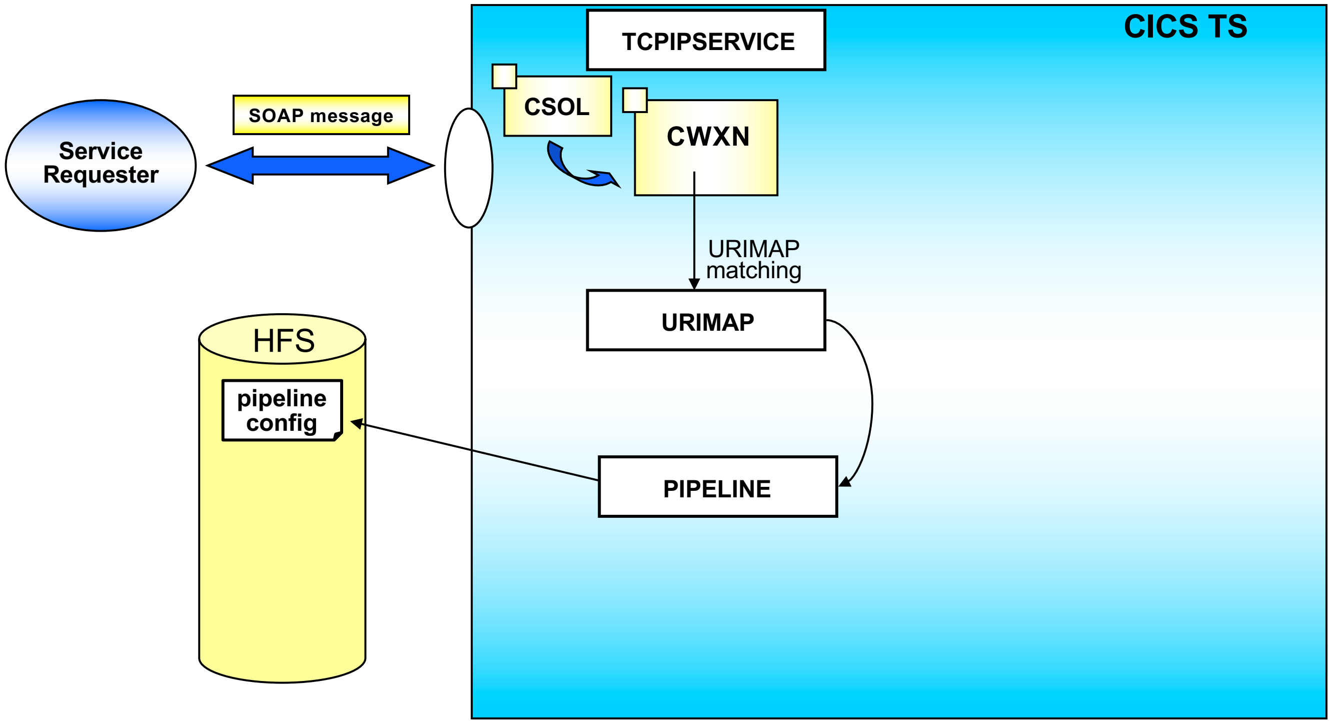
IBM Washington Systems Center

URIMAP matching …

- CICS searches installed URIMAP definitions to find a match on endpoint path
- URIMAP resource typically created dynamically by CICS
  - Can be defined by Systems Programmer
- If no matching URIMAP is found, CICS returns an error to client
- Work is performed under the CSOL or CWXN transaction (by default)
  - If configuration is eligible for optimized processing, all the work can be done by CSOL and CWXN transaction is not attached

IBM Washington Systems Center

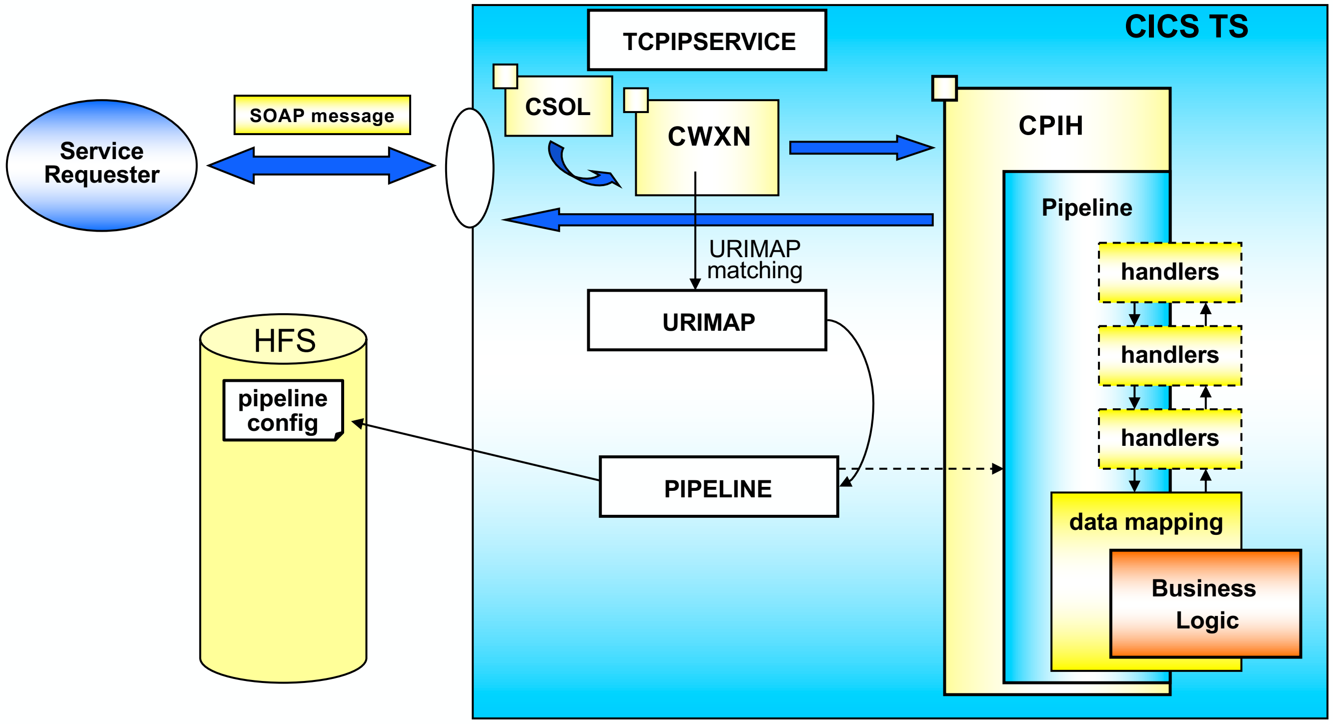
Pipeline configuration file (3 of 7)



- PIPELINE resource definition points to a Pipeline Configuration File

IBM
Washington
Systems
Center

Pipeline configuration file (4 of 7)

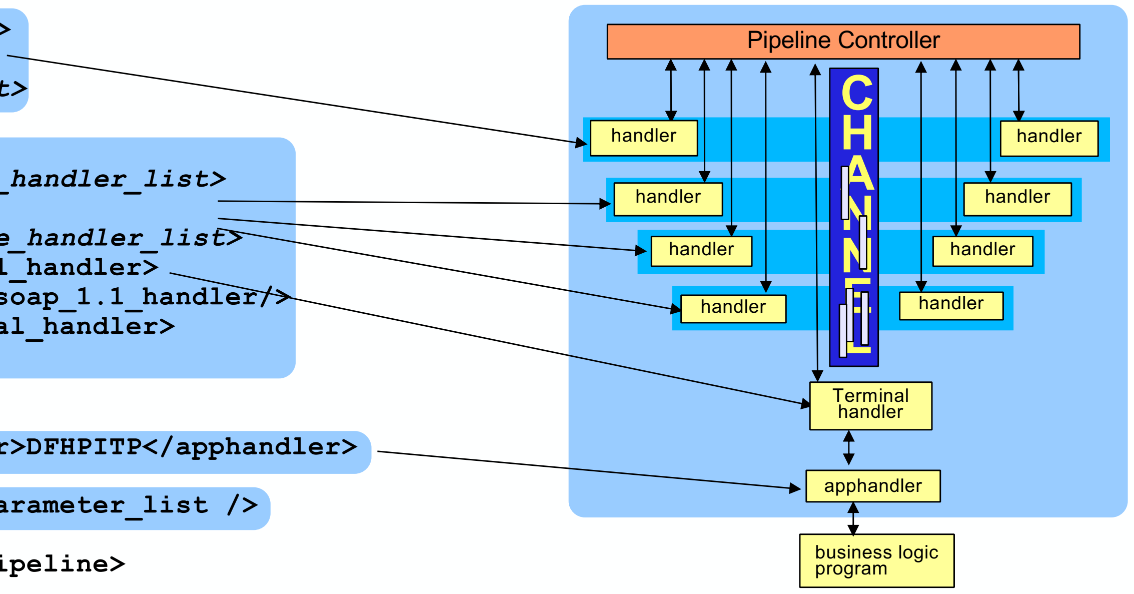


- The Pipeline Configuration File specifies a list of optional programs (called Handlers) that can look at or massage the message before it reaches the target

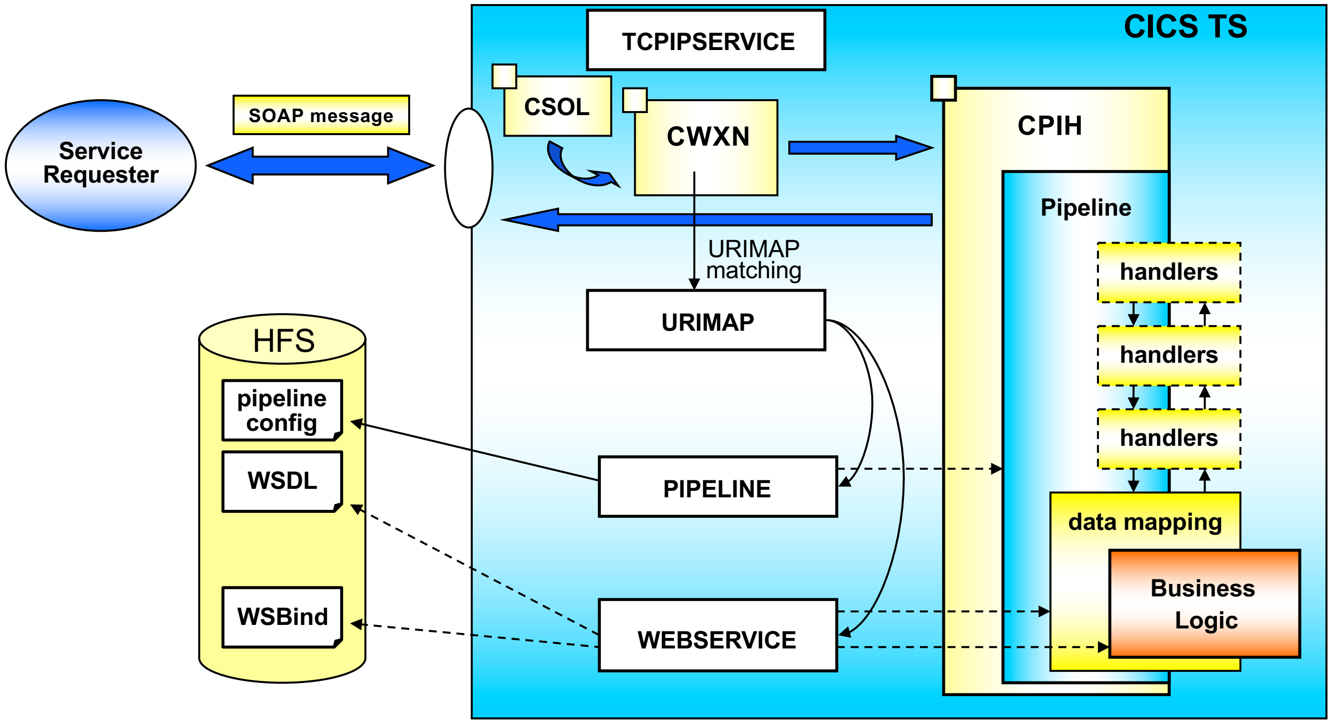
IBM
Washington
Systems
Center

Provider Pipeline Configuration (5 of 7)

```
<?xml version="1.0" encoding="UTF-8"?>
<provider_pipeline

  xmlns="http://www.ibm.com/software/htp/cics/pipeline"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ibm.com/software/htp/cics/provider.xsd ">
   <transport>
:
   </transport>

   <service>
:    <service_handler_list>

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

   <apphandler>DFHPITP</apphandler>

   <service_parameter_list />

</provider_pipeline>
```



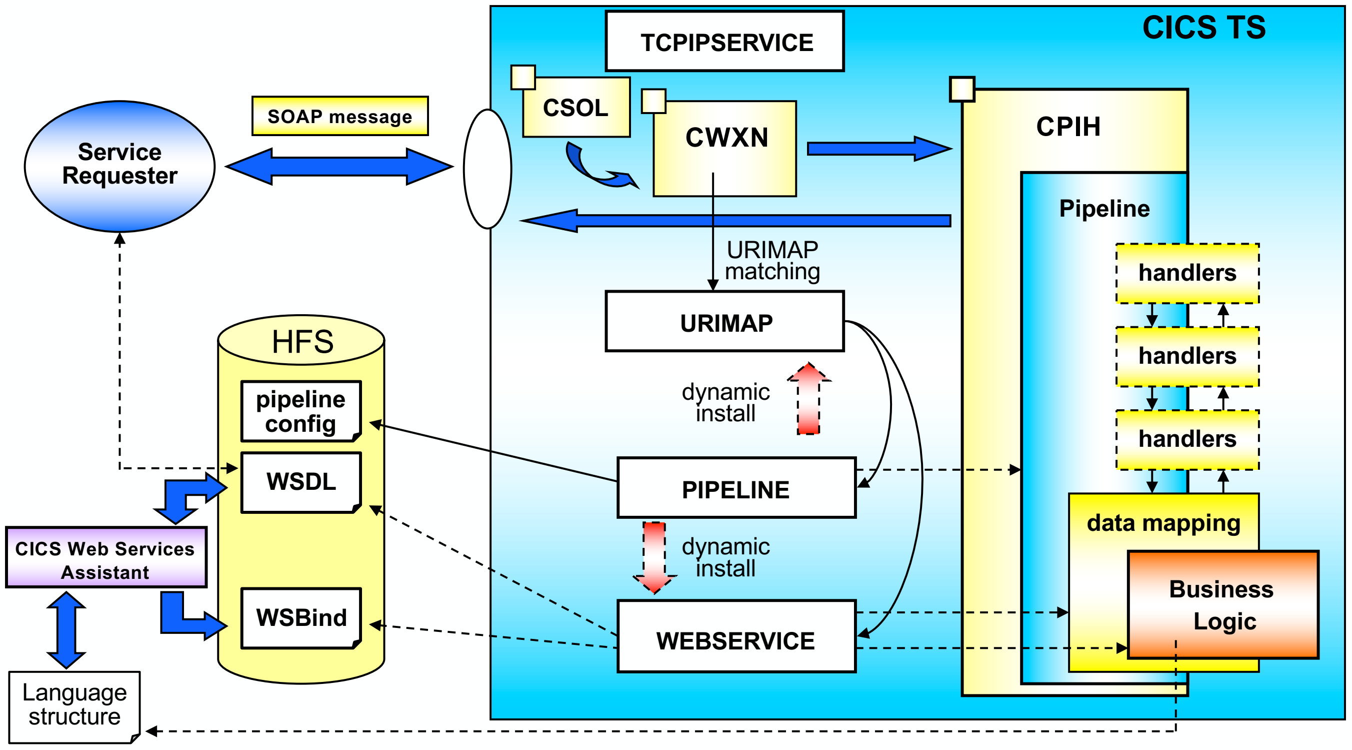
Data mapping uses WEBSERVICE definition (6 of 7)



- The CICS-supplied data mapping program is DFHPITP, which uses the information in the WEBSERVICE definition

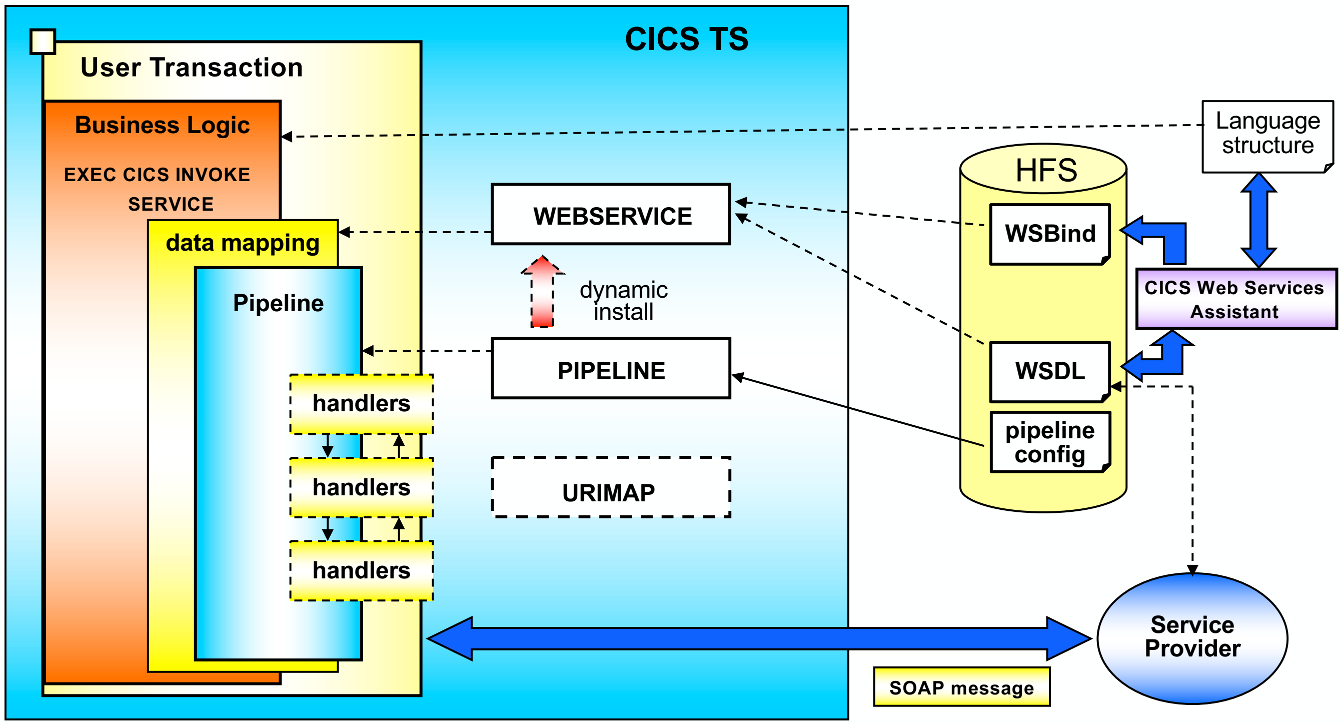
IBM Washington Systems Center

CICS Web Services: CICS as a Provider (7 of 7)



IBM
Washington
Systems
Center

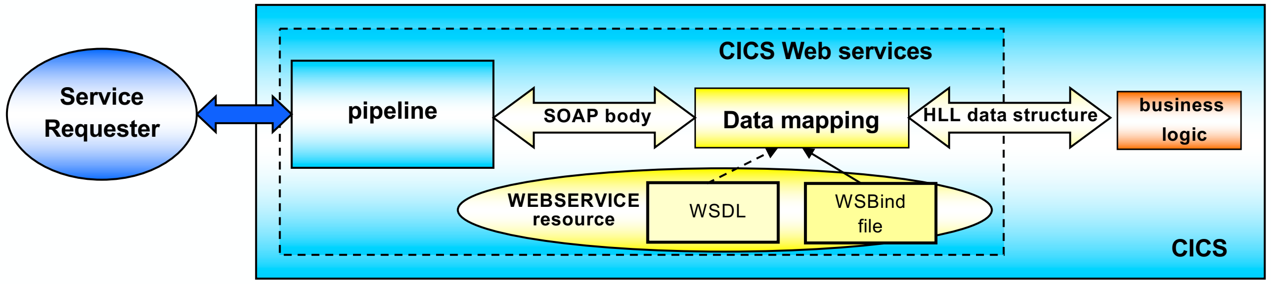
CICS Web Services: CICS as a Requester



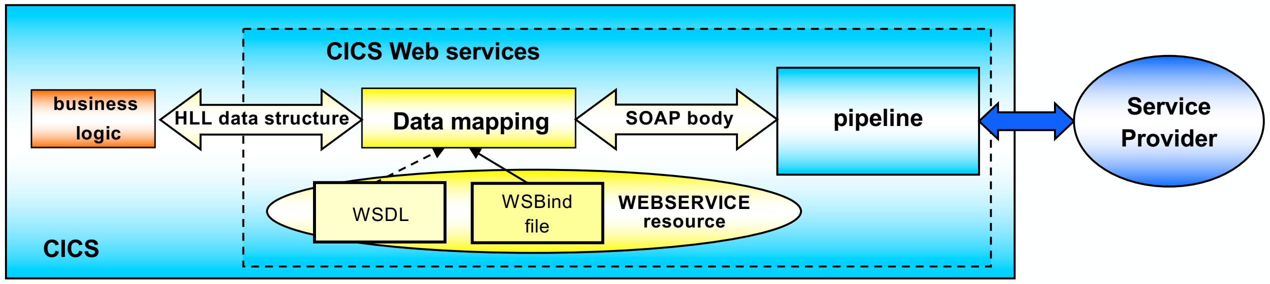
IBM
Washington
Systems
Center

CICS Data Mapping: usage of the WSBind file

- CICS as a service provider



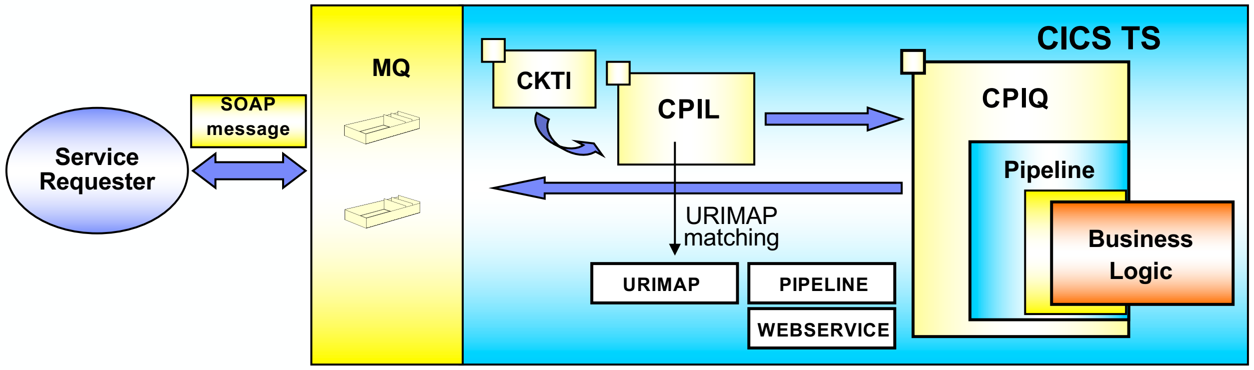
- CICS as a service requester



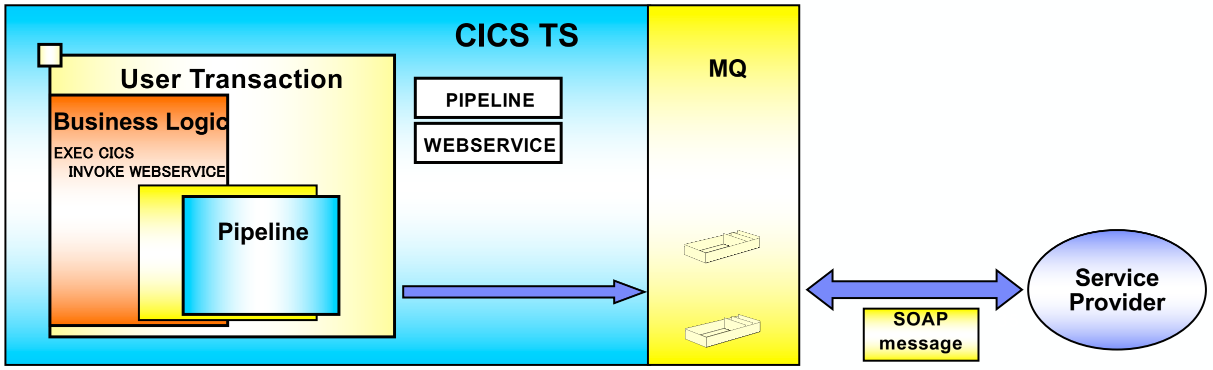
IBM
Washington
Systems
Center

WebSphere MQ Transports

- CICS as a service provider



- CICS as a service requester



IBM
Washington
Systems
Center

CICS Web Services Artifacts

- Systems Programmer
  - CICS Definitions
    - TCPIPService
    - URIMAP
    - WEBSERVICE
    - PIPELINE
  - Handler Programs
- Application Programmer
  - Business Logic Program
    - Associated Language Structures (copybooks)
  - WSDL file (to describe the service interface)
  - WSBIND file (for CICS’s conversion to/from XML/Language structure)

CICS Web Services Assistant available to help develop Application Programmer artifacts

CICS TS support of Web Service standards

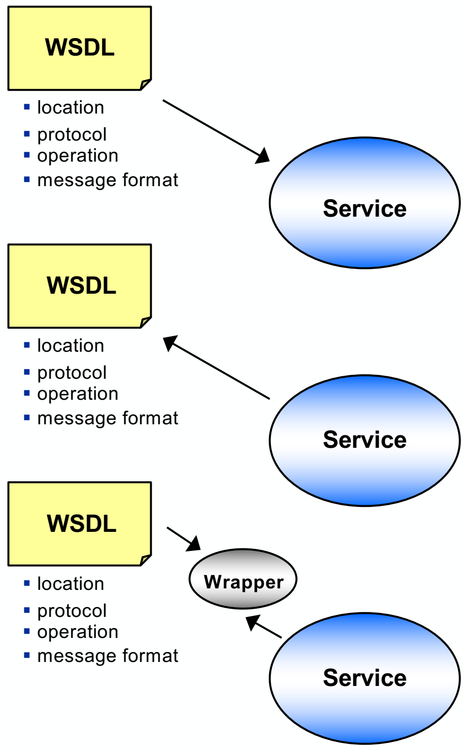
Both HTTP and MQ	both HTTP 1.0 and 1.1
XML Encryption Syntax and Processing	interoperability with entities using XML
XML Signature Syntax and Processing	interoperability with entities using XML
SOAP 1.1 and 1.2	to send and receive Web services messages
WSDL 1.1 and 2.0	to describe Web service interfaces
WSDL 1.1 Binding Extension for SOAP 1.2	for Interoperability with interfaces
WS-I Basic Profile 1.1	for interoperability between providers and requesters using SOAP
WS-I Simple SOAP Binding Profile 1.0	for interoperability using SOAP
WS-Atomic Transaction	for propagating transactional context
WS-Coordination	for coordinating transaction outcome
WS-Security	for authentication and encryption of all or part of a message (PK22736); username token; X.509 certificate token; SAML token; Kerberos
WS-Trust	for establishing trust relationships
MTOM / XOP	for efficient handling of large messages
SOAP 1.1 Binding for MTOM 1.0	to describe the use of MTOM
WS-Addressing	to indicate request and response routing

- CICS applications acting as providers or requesters are agnostic to transport mechanism used

IBM Washington Systems Center

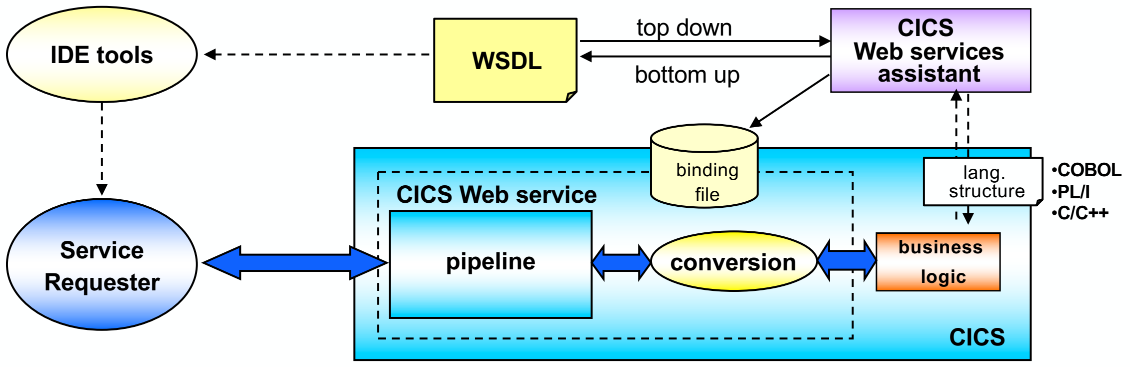
Developing Web Services Artifacts

- "Top down" approach
  - Create a service from an existing WSDL
    - Create a new Web service application
      - Better interfaces for the requester
      - New CICS code using the new languages structure
- "Bottom up" approach
  - Create a WSDL from an exiting application
    - Expose the application as a Web service
      - Quicker implementation of the service
      - Potentially more complex interface for the requester
- "Meet in the middle" approach
  - Create WSDL from an existing application, modify the WSDL and create a wrapper from the modified WSDL
    - Indirectly expose the application as a Web service
      - More suitable interface for the requester
      - Minimal, if any, CICS development



CICS Web Service Development

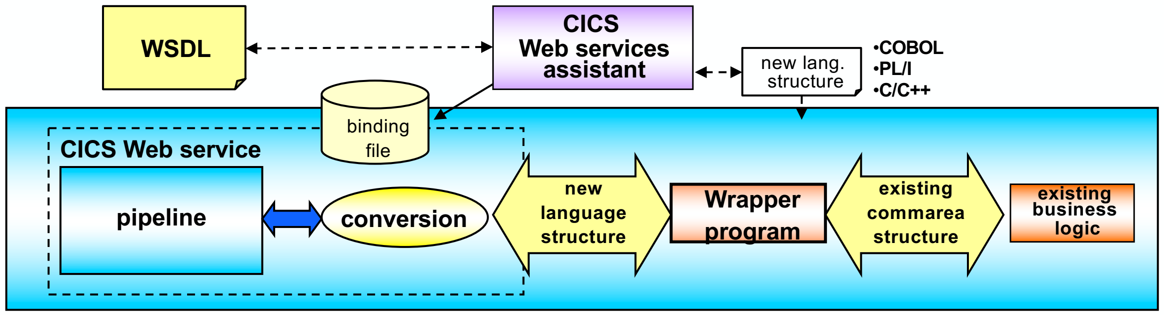
- CICS provides the necessary tools and runtime
  - WSDL can be generated from a utility
    - a bottom up approach from an existing application
  - Utility can generate language structures from WSDL
    - a top down approach to a new CICS service provider programs
    - for CICS service requester programs
  - CICS provides XML-language structure (Commarea or Container) conversion



IBM
Washington
Systems
Center

Meet In The Middle

- If you have an existing application and…
  - an existing WSDL is to be used as interface to the client
    - e.g. WSDL defined from a requester's perspective
  - only want to expose fields that are necessary to the requester
    - existing language structure may be complex, contain unnecessary fields for the requester
    - use an interface more suitable for the requester
  - the existing language structure uses data types not supported by the utility
    - wrapper program converts the data type to a supported data type
  - the existing application is written in a language not supported by the utility
    - Assembler or Java programs
  - etc.



IBM
Washington
Systems
Center

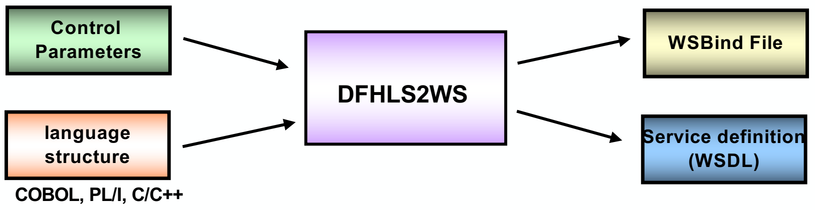
CICS Web Services Assistant

- Create CICS Web services artifacts
  - WSDL or language structure
  - WSBind file
- Two utility programs provided
  - DFHLS2WS
    - creates WSDL from a language structure
  - DFHWS2LS
    - creates language structure from WSDL
  - Supported languages are COBOL, PL/I, C, and C++
- Supplied JCL procedures will invoke either utility (a Java program)
- Generates the WSDL or language structure
  - Generates a Web service binding file (WSBind file)
    - for use in CICS Web service runtime

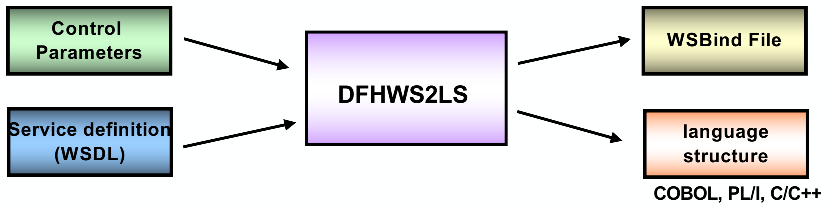
IBM Washington Systems Center

CICS Web Services Assistant

- DFHLS2WS (Language Structure to Web Service)



- DFHWS2LS (Web Service to Language Structure)



- Batch Jobs
- Samples supplied
- Limitations fully documented in the CICS manuals

Some of the CICS TS DFHLS2WS Input parms

Parameter	Description
PDSLIB	**Name of PDS containing the high level language structures to be processed**
REQMEM	**Member in the PDSLIB for the Web service request**
PDSCP	**PDS member code page**
RESPMEM	**Member in the PDSLIB for the Web service response**
STRUCTURE	**C and C++ only – structure name in REQMEM and RESPMEM**
LANG	**Programming language of input language structure. COBOL, PL/I, C, or CPP**
PGMNAME	**If CICS implements the service, name of the CICS application program**
TRANSACTION	**TRANSID to be placed in the generated URIMAP definition**
USERID	**USERID to be placed in the generated URIMAP definition**
URI	**If CICS implements the service, local URI to use as reference to the program**
PGMINT	**Target program interface. Either COMMAREA or CHANNEL**
CONTID	**Name of container that will hold the highest level of the language structure**

Parameter	Description
MAPPING-LEVEL	**Capabilities (1.0,1.1,1.2,2.0,2.1,2.2,3.0)**
CHAR-VARYING	**(NO or NULL) If MAPPING-LEVEL=1.2 or higher, if NULL, characters are null delimited**
MINIMUM-RUNTIME-LEVEL	**Lowest possible runtime level**
SOAPVER	**Protocol to be used (can be 1.1, 1.2, or ALL), only allowed when MINIMUM-RUNTIME-LEVEL=2.0**
CCSID	**Used to encode application structure character data (default is LOCALCCSID specified in SIT)**
REQUEST-NAMESPACE	**targetNamespace of the XML schemas (default: CICS generated)**
RESPONSE-NAMESPACE	**targetNamespace of the XML schemas (default: CICS generated)**
WSBIND	**Fully qualified HFS name of the Web service binding file**
WSDL, WSDL_1.1 or WSDL_2.0	**Fully qualified HFS name of the Web service description file**
XML-ONLY	**Set to YES if the program will be responsible for transforming XML to copybook**
LOGFILE	**Fully qualified HFS name for the log and trace information of the utility**
Many More….	**Documented in the CICS InfoCenter**

IBM
Washington
Systems
Center

DFHLS2WS sample job

```
//LS2WS JOB (accounting information),CLASS=A,REGION=0M
// SET QT=''''
//JAVAPROG EXEC DFHLS2WS,
// JAVADIR='java7'
//INPUT.SYSUT1 DD *
LOGFILE=/some/user/directory/ls2ws.log
WSDL=/some/user/directory/wsdl/RegisterPet.wsdl
PGMNAME=REGPETS
MAPPING-LEVEL=3.0
URI=/registerPet
PGMINT=CHANNEL
CONTID=PETCONTAINER
LANG=COBOL
WSBIND=/some/user/directory/RegisterPet.wsbind
PDSLIB=//USER1.COPYBOOK
REQMEM=TESTFILE
*/
```

< Input COBOL structure >

```
03 NAME                 PIC X(16).
03 AGE                  PIC 99.
03 PETS OCCURS 3.
    05 TYPE             PIC X(8).
    05 COLOUR           PIC X(8).
```

Fault processing

- SOAP <Fault> element is used to carry error information
  - Contained in the body of a message
- SOAP 1.1 Fault sub elements
  - <faultcode> Error information for software processing
  - <faultstring> Error information for a human reader
  - <faultactor> URI of the SOAP node that caused the fault
  - <details> Application specific error information related to <Body>
- SOAP 1.2 Fault sub elements
  - <Code> Error information for software processing
  - <Reason> Error information for a human reader
  - <Node> URI of the SOAP node that caused the error
  - <Role> Role the node was in when the error occurred
  - <Details> Application specific error information

IBM
Washington
Systems
Center

CICS API for fault processing

- Create a SOAP fault
  - `EXEC CICS SOAPFAULT CREATE FAULTCODE ( ) FAULTSTRING ( ) FAULTSTRLEN ( ) ROLE ( ) ROLELENGTH ( ) FAULTACTOR ( ) FAULTACTLEN ( ) DETAIL ( ) DETAILLENGTH ( ) NATLANG ( )`
- Add a reason to an existing fault in a specified language
  - `EXEC CICS SOAPFAULT ADD SUBCODEST ( ) SUBCODELEN ( ) FAULTSTRING ( ) FAULTSTLEN ( ) NATLANG ( )`
- Delete a Soap Fault that has been created
  - `EXEC CICS SOAPFAULT DELETE`

IBM Washington Systems Center

Web Service Invocation API

- **`EXEC CICS INVOKE SERVICE( ) CHANNEL( ) OPERATION( ) {URI() | URIMAP()}`**
  - Must have WSBIND file
  - CICS constructs message body
  - WEBSERVICE: name of the Web Service to be invoked
  - CHANNEL: name of the channel containing data to be passed to the Web Service (DFHWS-DATA container)
  - OPERATION: name of the operation to be invoked
  - URI: Universal Resource Identifier of the Web Service (optional)
  - URIMAP: Name of URIMAP resource with URL and other connection parameters (optional)
  - V3.2 – timeout value (RESPWAIT) can be specified on the PIPELINE definition

IBM
Washington
Systems
Center

Setting Userid and Tranid – (inbound web service)

- **Userid**
  - Specified in URIMAP (hard-coded URIMAP definition)
  - Specified in URIMAP (from WSBind file, CICS created URIMAP)
  - A HANDLER puts the userid in the DFHWS-USERID container
  - SSL   (CICS associates userid based on client certificate)
  - HTTP Basic authentication (normally not used for Web services)
  - WS-Security
  - WS-Trust
- **Tranid**
  - Specified in URIMAP (hard-coded URIMAP definition)
  - Specified in URIMAP (from WSBind file, CICS created URIMAP)
  - A HANDLER puts the tranid in the DFHWS-TRANID container

IBM
Washington
Systems
Center

CICS Web Service Support in IDz
(IBM Developer for z System)

IBM
Washington
Systems
Center

IDz: Creating CICS Web Service Artifacts

- "Interpretive" XML Conversion
  - Takes entire COBOL program as input (you select structure to expose)
  - Allows you to expose selected fields as input or output
  - Invokes the CICS Web Services Assistant Java classes 'under the covers'
  - Uses CICS's WSBind file conversion mechanism provided by DFHPITP
  - For top-down, also generates a template service implementation
  - For CICS as a requester, generates template program containing the Web service invoke
  - COBOL and PL/I

- Graphical interface

- Mainframe connection

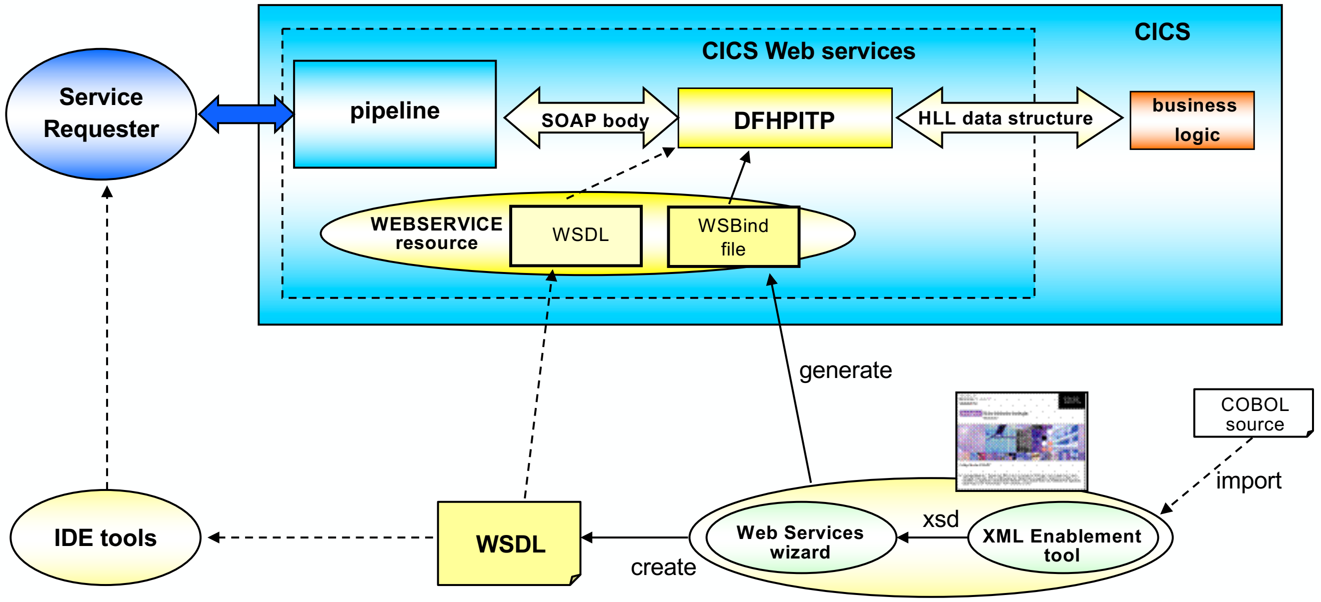
- WSDL Editor

- XML Wizards

- Much more

IDz “Interpretive” XML Conversion

- Invokes the CICS Web Services Assistant
  - Same Java classes as used on mainframe supplied with CICS
  - Graphical user interface
  - WSBind and WSDL file generation is performed on your workstation



IBM
Washington
Systems
Center

IDz: Creating CICS Web Service Artifacts

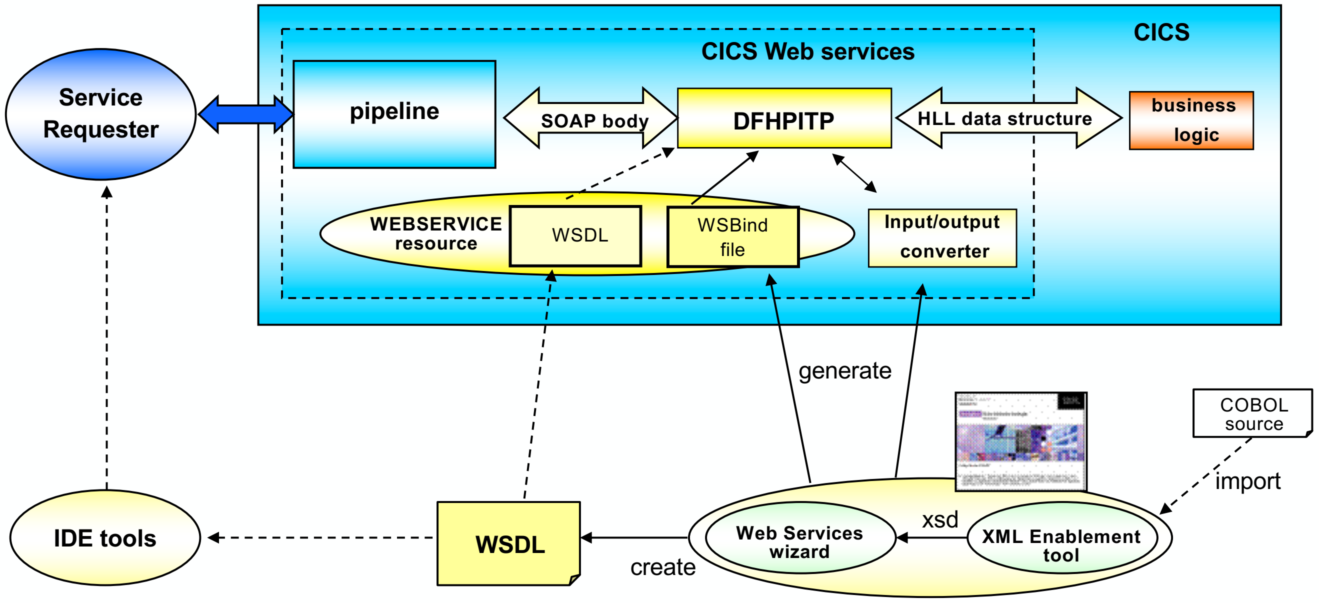
- "Compiled" XML Conversion
  - Takes entire COBOL program as input (you select structure to expose)
  - Allows you to expose selected fields as input or output
  - Generates a COBOL Converter program
  - Generates a WSBind file containing a 'Vendor Segment' which tells DFHPITP to use the generated Converter program
  - Supports more complex COBOL data structures like OCCURS DEPENDING ON and REDEFINES
  - You can modify the generated code for special needs
  - Generated COBOL program will need to go into your source repository
  - Generated COBOL program will need to go through production turnover
  - Generated COBOL program, if modified, will need user acceptance testing

- Graphical interface
- Mainframe connection
- WSDL Editor
- XML Wizards
- Much more

IBM Washington Systems Center

IDz “Compiled” XML Conversion

- Generates COBOL programs that have to be compiled
  - Graphical user interface
  - WSBind and WSDL file generation is performed on your workstation
  - Use of a “Vendor Segment” in the WSBind file tells DFHPITP to use the converter program instead of using its own conversion mechanism.



IBM
Washington
Systems
Center

Web Services in Java

IBM Washington Systems Center

CICS Liberty – Web Service support

- CICS TS V5.2+: JAX-WS and JAXB (in the Liberty profile)
- See an example in the CICSdev community article
  https://www.ibm.com/developerworks/community/blogs/cicsdev/entry/jax_ws_and_jaxb_support_in_cics_ts_v5_2_open_beta_liberty_profile?lang=en (Mark Cocker)
  - Article contains the steps necessary to get a sample web service running in:
    - Liberty on your desktop
    - CICS Liberty profile
  - Gets some CICS APIs running in the page that displays the web service output (CICS APIs only work in CICS)
  - Also tests the web service in the Eclipse Web Service Explorer

IBM
Washington
Systems
Center

CICS-Liberty – Web Services Support (CICS TS V5.2)

- Web Service Hello World (very, very, simple)

```
package com.ddw.verysimple.web.service;

import javax.jws.WebService;
import javax.jws.WebMethod;

@WebService
public class HelloWebService {
    private String message = new String("Hello, ");
    public HelloWebService() { }

    @WebMethod
    public String sayHello(String name) {
        return message + name + ".";
    }
}
```

- Eclipse wizards generate everything else:
  - Implementation
  - WSDL

The Liberty profile also supports JAXB
(JAXB can be used to prepare a string of XML that makes up the body of your web service)

IBM Washington Systems Center

Summary

- High Level Overview
- Development Styles
- Web Services Artifacts
  - WSDL
  - WSBind file
- CICS Web Services Assistant
    - DFHLS2WS
    - DFHWS2LS
- Rational Developer for System z(RDz)
  - Web Services, JSON Service, and XML transforms
- Service Flow Feature
- Web Services in Java