Understanding Urimaps, Pipelines and Webservices for CICS

Steve Webb
CICS Level 2 Support

WebSphere® Support Technical Exchange
Agenda:

- Web service overview
  - Requester / Provider, SOAP, WSDL, WSBIND
- Pipeline structure
- Application considerations
- CICS Web services resources
  - TCPIPSERVICE
  - URIMAP
  - PIPELINE
  - WEBSERVICE
- CICS catalog manager sample application
- Debugging
What is a Web service?

- Software system that allows program-to-program interactions over a network
- Clients and Servers communicate using Extensible Markup Language (XML)
- XML messages are exchanged using the Service Oriented Architecture Protocol (SOAP) – also known as Simple Object Access Protocol
What is a Web service? (continued)

- Interface defined by Web Service Definition Language (WSDL) which provides details necessary to interact with the service

- Independent of hardware or software platform
Type of Web service application

- **Requester**
  - CICS outbound request
  - “Requesting” information
  - CICS acting as client
- **Provider**
  - CICS receives inbound SOAP message and provides a response
  - “Providing” information
  - CICS acting as server
Type of Web service application - notes

- There are two types of Web service applications: requester and provider.
- A requester makes a request of a Web service.
- A provider supplies (provides) a response to a Web service.
- If CICS is a requester, CICS is referred to as the client and CICS sends an outbound request to a Web service. Usually the Web service will return a response to CICS.
- If CICS is a provider, CICS is referred to as the server and CICS receives an inbound SOAP request from a Web service. Usually CICS will provide a response to the Web service.
SOAP example

<SOAP-ENV:Body>
  <dispatchOrderRequest xmlns="http://www.exampleApp.dispatchOrder.Request.com">
    <itemReferenceNumber>10</itemReferenceNumber>
    <quantityRequired>5</quantityRequired>
    <customerId>STEVEW</customerId>
    <chargeDepartment>PLANNING</chargeDepartment>
  </dispatchOrderRequest>
</SOAP-ENV:Body>
SOAP example - notes

- The above SOAP message is from the CICS catalog manager sample application.
- SOAP messages are made up of name value pairs. In this case:
  - itemReferenceNumber has a value of 10
  - quantityRequired has a value of 5
  - customerId has a value of STEVEW
  - chargeDepartment has a value of PLANNING
WSDL

- XML document detailing specifications needed to invoke Web service
- Platform and programming language independent
WSDL - notes

- WSDL can be exchanged between application programmers within an organization to implement a requester and provider application.

- WSDL can also be exchanged between companies that wish to communicate through Web services.

- WSDL is not tied to a particular programming language or operating system platform so it enables different applications running on different platforms to communicate according to the specifications of the WSDL document.
WSDL example

dispatchOrder – itemReferenceNumber:

<xsd:element name="itemReferenceNumber" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:short">
      <xsd:maxInclusive value="9999" />
      <xsd:minInclusive value="0" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
WSDL example - notes

- The WSDL above is from the CICS catalog manager sample application.

- It defines the specifications for the element (variable) named itemReferenceNumber.

- It can have a value between 0 and 9999.
Preparing for a Web service

- CICS provides Web services assistant tooling
  - Bottom Up
    - Data structure to WSDL
    - DFHLS2WS
  - Top Down
    - WSDL to data structure
    - DFHWS2LS
- WebSphere Developer for System z (WD/z)
Preparing for a Web service - notes

- You can also use WebSphere Developer for System z (WD/z) to create your Web service.

- To do this, you create a Web service binding file, and the Web service description or language structures.

- For more information about this tool see http://www-306.ibm.com/software/awdtools/devzseries/
Bottom Up Approach example

DFH0XCP2 copybook

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).
```
Bottom Up Approach example - notes

- A bottom up approach takes a high level data structure as input and produces equivalent WSDL to be used by a Web service.
WSDL Equivalent

```xml
<xsd:element name="itemReferenceNumber"
nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:short">
      <xsd:maxInclusive value="9999" />
      <xsd:minInclusive value="0" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```
WSDL Equivalent - notes

- This WSDL above is the equivalent to the previous data structure:
  - 05 CA-ORD-ITEM-REF-NUMBER PIC 9(4).
Top Down Approach

- WSDL to data structure (language structure or copybook)
Top Down Approach - notes

- The top down approach takes the WSDL as input and produces a high level data structure (language structure or copybook).
Binding – WSBIND

- Connects WSDL and data structure
- Enables inbound SOAP message mapping to data structure
- Enables data structure mapping to outbound SOAP message
- WSBIND must match WSDL and data structure being used
Binding – WSBIND - notes

- The CICS Web services assistant tooling (or WebSphere Developer for System z tooling) produces a binding file, referred to as a WSBIND file.

- It is the WSBIND file that enables the transformation of a SOAP message into a data structure and the data structure into a SOAP message.

- It is the glue (binding) that connects the WSDL specifications to a high level language data structure.
Pipeline Structure

- Message handler is a program that can process Web service requests and responses
- Message handler can modify request / response
- Pipeline consists of series of message handlers executed in a sequence
- May have different message handlers for HTTP vs MQ transports
Pipeline Structure - notes

- A pipeline can be for either a request (requester) or a response (provider), but a single pipeline cannot be used for both.

- A requester pipeline would be used for an outbound request (where CICS is the client) to some other Web service.

- A provider pipeline would be used for an inbound SOAP message (where CICS is the server) from some other Web service and CICS will provide the response.
Service Provider Pipeline

- Service Requester
- Message Handler 1
- Message Handler 2
- Message Handler 3
- CICS Transaction Server
- CICS Web Services
- CICS Application Program

Flow:
- Request
- Response

Diagram illustrates the flow of data between the service requester and the CICS application program through message handlers.
Service Provider Pipeline - notes

- In a service provider pipeline, CICS receives an inbound SOAP message from a service requester.

- This SOAP message then passes through the message handlers in the pipeline.

- The final message handler is referred to as the terminal handler since it is responsible for turning the request around and producing the response back through the pipeline for the requester. Usually, a terminal handler will take the request as input and pass this to a CICS application program. The application program’s response will be used by the terminal handler to construct the response and send it back out through the pipeline.
SOAP Body to Data Structure

- CICS will transform contents of SOAP body to application data structure
- WSBIND and WSDL files are used for the mapping
  - DFHLS2WS, DFHWS2LS, WD/z
- CICS-supplied SOAP message handler in pipeline maps the data
The process of transforming the contents of a SOAP body to an application data structure and back is referred to as mapping.

If you use the CICS Web services assistant, CICS supplies a SOAP message handler to map the data. See the discussion for the pipeline configuration file later in the presentation.
Data Mapper

Service Requester

Request

Response

Message Handler 1

CICS Web Services

Message Handler 2

Message Handler 3

WSBIND

WSDL

Data Mapper

CICS Transaction Server

CICS Application Program
Data Mapper - notes

- The data mapper is like the brain to transform the contents of the SOAP message body to the high level language interface or data structure.

- It uses the WSBIND file (and the WSDL file if full validation is turned on) to do this.

- It also transforms the contents of the data structure back to a SOAP message body.

- The picture above is a service provider, but the same process in reverse applies for a service requester.
Business Logic

Client -> Communications Logic -> EXEC CICS LINK -> Business Logic

CICS Transaction Server

Client -> CICS Web services support -> Business Logic

CICS Transaction Server
Business Logic - notes

- If your business logic is separated from your communication logic, and the data structure is supported by the CICS Web services assistant, then you can quite easily use CICS Web services to access your application.

- If, however, either your application data structure is not supported by the CICS tooling, or your business logic is not well separated from your communication logic, you will need to use a wrapper program which we will discuss next.

- Note: If you do not use the CICS Web services assistant, message handlers in your pipeline must interact directly with your business logic.
Accessing Application via Web service

- Rewrite part of the application to separate business logic

- Create a wrapper program
  - **Provider**
    - Wrapper receives request and links to business logic program
  - **Requester**
    - Business logic links to wrapper program which calls Web service
Accessing Application via Web service - notes

- Depending on your application design, there are several steps you may need to take to expose your application to a Web service.

- You may need to rewrite part of the application to separate the communication logic from the business logic, making it easier to directly replace the communication logic with CICS Web services.

- You may choose to write a wrapper program which will interface with CICS Web services and the application’s data structure.
Wrapper Program

- If tooling cannot generate code to interact directly with business logic
- Create data structure for wrapper
- Manipulate data for business logic program
- Moves data between wrapper’s data structure and business logic’s data structure
- Invokes business logic program
Wrapper Program - notes

- The wrapper would be responsible for taking the wrapper’s data structure (that is supported by CICS Web services) and manipulating the data as required so it can be used by the existing business logic application.

- The wrapper would move the data from the wrapper’s data structure to a data structure the business logic can understand and it would invoke the business logic application using the existing interface.

- Once the business logic application produced a response, the wrapper would be responsible for moving the data back to the wrapper’s data structure so that a CICS Web services response can be built.
Wrapper Program (continued)

Client ➔ CICS Web services support ➔ Wrapper Program ➔ Business Logic

EXEC CICS LINK

CICS Transaction Server
Wrapper Program (continued) - notes

- In this scenario, the client sends in a SOAP message which is processed by CICS Web services support (pipeline processing) and the wrapper program is invoked.

- The wrapper program has a data structure that the CICS Web services support can work with.

- The wrapper then processes the wrapper’s data structure into a form that the business logic’s data structure can understand and it invokes the business logic application.

- The wrapper manipulates the data on the way back out as well.
CICS Web services Resources

- TCPIPSERVICE
- URIMAP
- PIPELINE
- WEBSERVICE
TCPIPSERVICE

- Required for service provider using HTTP
- Provides access to CICS from TCP/IP network via port number
TCPIPSERVICE - notes

- There are other parameters on the TCPIPSERVICE definition, but for this presentation, PORT is the important one.

- The CICS Catalog manager sample application uses a TCPIPSERVICE name of EXMPPORT and I used Port 08021.

- Tcpipservice(EXMPPORT)

- Port(08021)
URIMAP

- Required for service provider

- Passes incoming request to appropriate pipeline based on URI (URL) match on the PATH attribute

- Usually created automatically from WSBIND when pipeline is installed
URIMAP - notes

- For example, the following attributes of interest are for the URIMAP automatically created for the CICS catalog sample application after installing Pipeline(EXPIPE01).

- Urimap($918180)

- Usage(Pipe)
  - Pipe indicates this for an inbound request, where CICS will be the “provider” of the information.
URIMAP – notes continued

- Path(/exampleApp/dispatchOrder)
  - Path is mixed case and is used to match (or pattern match) the incoming URI to an installed URIMAP definition.
  - So an incoming URI of /exampleApp/dispatchOrder will match this URIMAP and therefore this is the URIMAP that will be used. This information is obtained from the WSBIND file when the pipeline is installed, or a Pipeline Scan is done; it can also be specified on a URIMAP resource definition.
URIMAP – notes continued

- **Transaction(CPIH)**
  - Default alias transaction for Usage Pipeline. This is the transaction id that will start the pipeline.

- **Program()**
  - Only used for Usage Server; don’t confuse this with the Program attribute on the WEBSERVICE definition.
URIMAP – notes continued

- **Pipeline(EXPIPE01)**
  - Specifies the PIPELINE that will be used to process this incoming request. The referenced PIPELINE resource definition details the message handlers in the pipeline configuration file.

- **Webservice(dispatchOrderEndpoint)**
  - dispatchOrderEndpoint is the name of the Web service generated by the CICS Web services assistant. You will find a dispatchOrderEndpoint.wsbind has been created from the tooling. Webservice can also be the name of a WEBSERVICE resource definition, but usually it will be the mixed case name generated from the CICS Web services assistant.
PIPELINE

- Required in both provider and requester
- Pipeline configuration file
  - z/OS UNIX XML file details of message handler programs
- Location of WSDL and WSBIND files
- URIMAP name
- WEBSERVICE name
PIPELINE - notes

- CICS provides a standard set of message handlers that you can use to enable different options in your pipeline. A basic pipeline sample, basicsoap11provider.xml, is provided in library /usr/lpp/cicsts/samples/pipelines.

- The PIPELINE definition EXPIPE01, for the CICS catalog manager sample application, would look like the following from a CEMT INQUIRE PIPELINE - depending on the location of your configfile, pickup directory etc:

  - Pipeline(EXPIPE01)
  - Mode(Provider)
  - Soaplevel(1.1)
PIPELINE – notes continued

- **Configfile(/u/webbste/dir1/spud20/dir2/samples/pipelines/basicsoap11provider.xml)**
  - Specifies path and file for the pipeline configuration file which details the message handlers

- **Shelf(/u/webbste/var/cicsts/)**
  - Shelf directory where files are written, primarily WSBIND files. Directories specific to the region’s applid and pipeline name are also created to hold the WSBIND files.
PIPEDLINE – notes continued

- Wsdir(/u/webbste/dir1/spud20/dir2/samples/webservices/wsbind/provider/)
  - Web service binding directory, or pickup directory. WSBIND files found here are used to install the WEBSERVICE(s) and URIMAP(s) when the PIPELINE is installed, or a Pipeline Scan is done. You manually copy the WSBIND files needed for this pipeline into the Wsdir pickup directory.
WEBSERVICE

- Required if CICS Web services assistant was used
- Usually created automatically from WSBIND when pipeline is installed
- PROGRAM attribute required for service provider
- PROGRAM can not be specified for service requester
The CICS catalog manager sample application WEBSERVICE definition dispatchOrderEndpoint (installed automatically from the Wsdir pickup directory when the PIPELINE is installed) would look like the following from a CEMT INQUIRE WEBSERVICE. I have only included attributes of interest to this presentation.

- Webservice(dispatchOrderEndpoint)
- Pipeline(EXPIPE01)
  - Pipeline associated with this Webservice
WEBSERVICE - notes

- Urimap($918180)
  - Urimap associated with this Webservice. It was installed automatically when the pipeline was installed.

- Program(DFH0XODE)
  - Name of program that implements the Webservice. DFH0XODE is a CICS supplied COBOL program that performs the “Order Dispatch” function.

- Pgminterface(Commarea)
  - The program expects input in the form of a commarea (this could also be a channel)
WEBSERVICE - notes

- Wsbind(/u/webbste/dir1/spud20/dir2/samples/webservices/wsbind/provider/dispatchOrderEndpoint.wsbind)
  - Webservice binding file used to convert data structure to WSDL and WSDL to data structure.

- Endpoint(http://my-server:9080/exampleApp/dispatchOrder)
  - Endpoint URI of a remote Webservice, specified in the WSDL, and it is picked up from the WSBIND file in the WSDIR pickup directory.

- Binding(dispatchOrderSoapBinding)
  - From WSDL file - <binding name="dispatchOrderSoapBinding" type="tns:dispatchOrderPort"> It is picked up from the WSBIND file in the WSDIR pickup directory.
WEBSERVICE - notes

- A WEBSERVICE is required if you use the CICS tooling since it specifies the location of the resources necessary for CICS to convert the SOAP message to a data structure (and data structure to SOAP message). If the CICS tooling is not used, the message handlers in your pipeline would need to communicate with the business logic directly and do the translation from SOAP message to data structure and from a data structure to a SOAP message.
PIPELINE Configuration

```xml
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<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/pipeline provider.xsd">
  <service>
    <terminal_handler>
      <cics_soap_1.1_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```
PIPELINE Configuration - notes

- Provider pipeline configuration file, basicsoap11provider.xml in /usr/lpp/cicsts/samples/pipelines, is shown above.

- `<provider_pipeline>` element indicates it is a provider pipeline configuration file.

- `<service>` element specifies the message handlers that are invoked for every request.
PIPELINE Configuration - notes

- Since this is a provider pipeline configuration, it has a `<terminal_handler>` element. The `<terminal_handler>` contains the terminal message handler, in this case `<cics_soap_1.1_handler>` since it will process SOAP 1.1 messages. The `cics_soap_1.1_handler` is the CICS supplied program DFHPISN1. If this was `cics_soap_1.2_handler`, it would use DFHPISN2.

- The `<apphandler>` element specifies the target application (or wrapper program). In this case, since the application is deployed with the CICS Web services assistant, the program is DFHPITP. It will be linked to by the terminal handler (DFHPISN1 for `cics_soap_1.1_handler`).
Install Pipeline

- Creates WEBSERVICE and URIMAP definitions based on WSBIND information from WSDIR pickup directory

- Install done in two stages
  - Resource Definition Online (RDO) phase
  - Pipeline resolution phase
Install Pipeline - notes

- The install of a PIPELINE resource is actually done in two stages. The pipeline may install ok, but there could be problems with the resolution stage of the pipeline, which could lead to the pipeline being disabled, or the WEBSERVICE or URIMAP resources not being installed completely. If this happens, you may need to trace the installation of the PIPELINE resource to see where it is failing.
Putting it all together - Install Pipeline

TCPIPSERVICE(EXMPPORT)
PORT(08021)

PIPELINE(EXPIPE01)
CONFIGFILE(…basicsoap11provider.xml)
SHELF(…cicsts)
WSDIR(…webservices/wsbind/provider)

URIMAP($918180)
PATH(/exampleApp/dispatchOrder)
PIPELINE(EXPIPE01)
WEBSERVICE(dispatchOrderEndpoint)

WEB SERVICE(dispatchOrderEndpoint)
PIPELINE(EXPIPE01)
URIMAP($918180)
PROGRAM(DFH0XODE)
WSBIND(…dispatchOrderEndpoint.wsbind)
ENDPOINT(http://ip:port/exampleApp/dispatchOrder)
BINDING(dispatchOrderSoapBinding)

WSDIR pickup directory
dispatchOrderEndpoint.wsbind

Configfile – msg handlers
Putting it all together - Install Pipeline - notes

- When a PIPELINE is installed (or a Pipeline Scan is done), a URIMAP and WEBSERVICE are installed. The pipeline install process uses the WSDIR pickup directory and a WEBSERVICE and URIMAP are installed for every WSBIND file found in the pickup directory.

- The picture above shows the relationships between the resources after the PIPELINE is installed. This is from the CICS catalog manager sample application.

- The pipeline configfile parameter gives the path and name of the pipeline configuration file.
Putting it all together – Inbound Flow

TCPIPSERVICE (EXMPPORT)
PORT (08021)

PIPELINE (EXPIPE01)
CONFIGFILE (…basicsoap11provider.xml)
SHELF (…cicsts)
WSDIR (…webservices/wsbind/provider)

URIMAP ($918180)
PATH (/exampleApp/dispatchOrder)
PIPELINE (EXPIPE01)
WEBSERVICE (dispatchOrderEndpoint)

WEBSERVICE (dispatchOrderEndpoint)
PIPELINE (EXPIPE01)
URIMAP ($918180)
PROGRAM (DFH0XODE)
WSBIND (…dispatchOrderEndpoint.wsbind)
ENDPOINT (http://ip:port/exampleApp/dispatchOrder)
BINDING (dispatchOrderSoapBinding)

WSDIR pickup directory
dispatchOrderEndpoint.wsbind

Configfile – msg handlers
Putting it all together – Inbound Flow - notes

- The flow of an incoming SOAP request is shown above.
- The data arrives at the port specified on the TCPIPSERVICE definition.
- Based on the incoming uri, CICS finds a URIMAP definition that best fits the incoming uri. Once the URIMAP is identified, CICS knows the PIPELINE to process the request and the SOAP message moves to the pipeline stage.
- The PIPELINE definition tells CICS which pipeline configuration file to use, which lists the message handlers that will be invoked.
- The WEBSERVICE definition (which was also identified from the URIMAP match) then becomes involved. The WEBSERVICE details what program will be used to process the request (note this is for a service provider example). The WEBSERVICE also contains the location of the WSBIND file which will be used by CICS Web services to transform the SOAP message to an application data structure and back.
WSDL graphical representation
WSDL graphical representation - notes

- There are tools available to process and validate WSDL files.
- The example above shows a graphical representation of the CICS catalog manager sample application’s dispatchOrderService.
- Validation tools can help find errors with WSDL files before you try to run them through the CICS Web services assistant (DFHWS2LS).
- The small box in the middle at the top is the binding for the MQ transport, which I did not configure.
- The other small box in the middle at the bottom is the binding for the HTTP transport.
Catalog Manager sample application

- CICS-supplied

- Configurations
  - CICS requester to CICS provider
  - Same CICS acting as requester and provider
  - WebSphere Application Server (WAS) requester to CICS provider

- See Chapter 14 Web Services Guide SC34-6838
Catalog Manager sample application - notes

- HIGHLY recommend starting with the CICS catalog manager sample application. Make sure it works before starting development for other applications.

- The sample can be configured on two different CICS regions, where one acts as the requester and the other acts as the provider. It can also be configured in a single CICS region acting as both the requester and the provider. Sample EAR files are also provided to configure a WebSphere Application Server as the requester and CICS as the provider.
Debugging

- CEDF / CEDX
- AUXTRACE
- Dump with trace components
- Validation
- Editor for validating WSDL document
- Log file and job output from tooling
- Reporting problems to IBM
  - MustGather: Documentation for Web services problems in CICS on z/OS
Debugging - notes

- CEDX can be used to step through Web services processing.

- Trace is a very valuable tool for debugging runtime problems with a URIMAP, PIPELINE or WEBSERVICE.

- Typically I would recommend all standard trace components set to level 1, and the following set to standard level 1-2: AP, EI, PI, PG, SO and WB.

- If you are receiving a CICS message, you can set up the trace components as above, with an appropriate internal trace table size (10-20 Meg is usually good), and obtain a dump on the CICS message by putting it into the system dump table. For example, if you were getting a DFHPI1234, you can put this in the system dump table with:
  - CEMT SET SYD(PI1234) SYS MAX(1) ADD
Debugging - notes

- Turning on validation for the WEBSERVICE will help find problems in the message exchange process between a requester and a provider. CEMT SET WEBSERVICE(name) VALIDATION can be used, or specify it on the RDO definition of the WEBSERVICE if it is RDO defined. The validation status can be changed with CEMT.

- Using a WSDL editor that can perform validation can help find problems with a WSDL file before you try to run it through the CICS Web services assistant. If the WSDL isn’t valid, DFHWS2LS will not be able to process it. If it passes a WSDL editor’s validation tests, it doesn’t necessarily mean DFHWS2LS will be able to process it. But if it fails the WSDL editor’s validation, then the WSDL needs to be fixed before trying to process it with DFHWS2LS.

- Examine the log file produced by the CICS Web services assistant as well as the job output from DFHWS2LS and DFHLS2WS.
Summary

- Web service components
- CICS Web services assistant tooling (and WD/z)
- Application considerations
- CICS Web services resources
- CICS Catalog manager sample application
- Debugging
References

- CICS Transaction Server for z/OS Web Services Guide SC34-6838
- CICS Transaction Server for z/OS Resource Definition Guide SC34-6815
- IBM Redbooks
  - Search:
    - CICS Web services
    - CICS SOA
Additional WebSphere Product Resources

- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at: http://www.ibm.com/developerworks/websphere/community/

- Learn about other upcoming webcasts, conferences and events: http://www.ibm.com/software/websphere/events_1.html

- Join the Global WebSphere User Group Community: http://www.websphere.org

- Access key product show-me demos and tutorials by visiting IBM Education Assistant: http://www.ibm.com/software/info/education/assistant


- Sign up to receive weekly technical My support emails: http://www.ibm.com/software/support/einfo.html
Questions and Answers