IBM 2007 IMPACT

IBM

Implementing Web Services in CICS TS V3

Darren Beard

CICS Developer

darren_beard@uk.ibm.com

4116A

impact·venture*

IBM 2007 IMPACT

IBM

This presentation attempts to explain the new support provided for Web Services in CICS TS V3.1 and CICS TS V3.2. The main emphasis is on V3.1 with some pointers to what is new in V3.2.

Web Services Introduction

- What is a Web Service?
  - A Web Service
    - Describes a collection of operations
    - Is network accessible
    - Uses standardized XML messaging
  - A Web Service is described:
    - Using standard, formal XML notation (service description)
    - Covers all the details necessary to interact with the service
      - Message formats
      - Transport protocols and location
      - Independent of hardware or software platform
      - Independent of programming language

A common program to program communication model built on existing and emerging standards, such as, HTTP, XML, SOAP, WSDL and UDDI.

A Web service is a collection of operations that are network accessible through standardized XML messaging.

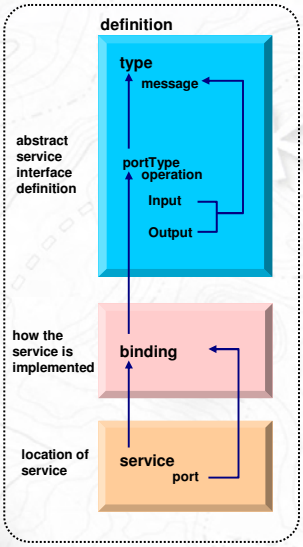
A Web service is described using a standard, formal XML notation, called its service description.
It covers all the details necessary to interact with the service, including message formats (that detail the operations), transport protocols and location.

The interface hides the implementation details of the service, allowing it to be used independently of the hardware or software platform on which it is implemented and also independently of the programming language in which it is written. This allows and encourages Web Services-based applications to be loosely coupled, component-oriented, cross-technology implementations.

Web Services fulfill a specific task or a set of tasks. They can be used alone or with other Web Services to carry out a complex aggregation or a business transaction.

WSDL

- **Web Services Description Language**
  - XML based language to describe an interface of a service
  - WSDL comprises of
    - type
    - portType (WSDL 2 calls this an interface)
    - message
    - operation
    - binding
    - service
    - port (WSDL 2 calls this an endpoint)

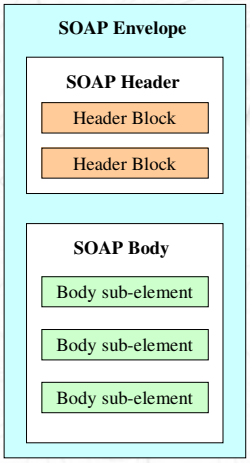


type	the data types in the form of XML schemas (usually)
portType	an abstract set of operations mapped to one or more end points
message	an abstract definition of the data in the form of a message
operation	the abstract definition of the operation for the message
binding	the concrete protocol and data formats for the operations
service	a collection of related end points
port	a combination of binding and a network address

Note that the above terms are as used in WSDL 1.1. In WSDL 2.0, some of the terms are different, as indicated on the slide.

SOAP Messages

- What does the “standardized XML message” look like?
  - SOAP 1.1 or 1.2 message
    - Soap envelope <Envelope> consisting of:
      - Optional header element, <Header>
      - Body element, <Body>
      - Optional fault element <Fault>, may be added by service provider



SOAP is a protocol for the exchange of information in a distributed environment. SOAP messages are encoded as XML documents, and can be exchanged using a variety of underlying protocols.

A SOAP message is encoded as an XML document, consisting of an <Envelope> element, which contains an optional <Header> element, and a mandatory <Body> element. The <fault> element, contained within the <Body> is used for reporting errors.

The SOAP <Envelope> is the outermost element in every SOAP message, and contains two child elements, an optional <Header> and a mandatory <Body>.

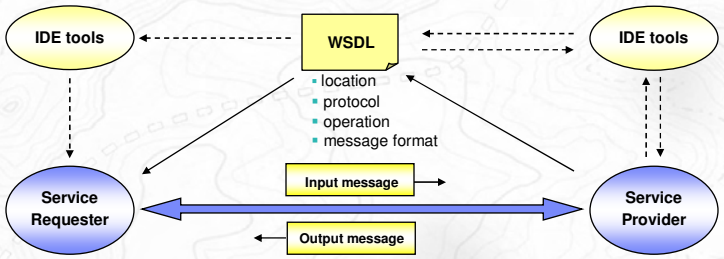
The SOAP <Header> is an optional element within the SOAP message, and is used to pass information in SOAP messages that is not application payload. The SOAP header allows features to be added to a SOAP message in a decentralized manner without prior agreement between the communicating parties. SOAP defines a few attributes that can be used to indicate who should deal with a feature and whether it is optional or mandatory.

The SOAP <body>, a mandatory element, containing information intended for the ultimate recipient of the message. In CICS terms, this will become the COMMAREA for the application program.

The SOAP <fault>, an element contained within the <body>, used for reporting errors.

Production and usage of WSDL

- Requests from Service Requesters will be generated based on the information contained in WSDL
- IDE tools help generation of WSDL or application



A web service provider needs to produce a WSDL document to describe the service which is being provided. This document is then published, using UDDI for example, or otherwise communicated to potential requesters of the service. Various tools may be used to assist with the generation of the WSDL.

A web service requester needs to obtain the WSDL description of the service which it wants to use. Based on the WSDL, an application can be produced which will be able to request the services described in the WSDL.

IDE – Integrated Development Environment

IMPACT IBM

Usage Scenarios

- CICS as a Service Provider using existing program (bottom up)
  - Existing application not changed
    - Existing language structure
- CICS as a Service Provider using new program (top down)
  - New application
    - Existing WSDL
- CICS as a Service Requester using a new program (top down)
  - New application
    - Existing WSDL

IMPACT IBM

There are three usage scenarios where the CICS tooling is thought to be likely to be applied. These situations are indicated in the foil.

Batch Tooling

(CICS Web Services Assistant)

The next section of the presentation explains the batch time tooling provided to assist with web service deployment in CICS.

IMPACT IBM

"Bottom up" approach to web services

- An existing CICS application is to be exposed as a web service
  - Language structure(s) need to be extracted from the source code
  - If the COMMAREA is very complex, it may be necessary to write a 'wrapper program' to map the COMMAREA into a form which can be handled by the CICS tooling.
  - Use a CICS supplied batch procedure (DFHLS2WS) to convert language structure(s) to WSDL.
    - The language structures can be COBOL, PL/I, C or C++
    - Publish the generated WSDL, on UDDI for example.
    - A file called the WSBind file is also produced.

IMPACT IBM

For an existing CICS application, the COMMAREA will already be mapped by a language structure. The language structure is used as input to a CICS supplied utility which runs as a batch procedure. The procedure is called DFHLS2WS. This converts the supplied language structure into a WSDL document. A special file, the WSBind file, is also generated. This needs to be placed in an HFS directory where CICS will subsequently find it and install it.

The WSDL so produced may then be published to the potential clients through the UDDI for example.

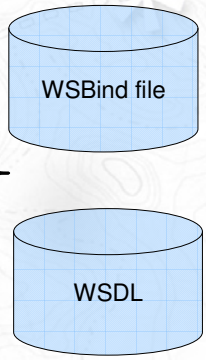
Batch Processing (DFHLS2WS)

Language structure(s)

```
05 WSTEST2.
 10 Wrapper.
  15 SOME-DATA          PIC X(79).
  15 USER-DETAILS.
   20 FIRST-NAME        PIC X(10).
   20 LAST-NAME         PIC X(10).
   20 AGE               PIC 9(3) DISPLAY.
```

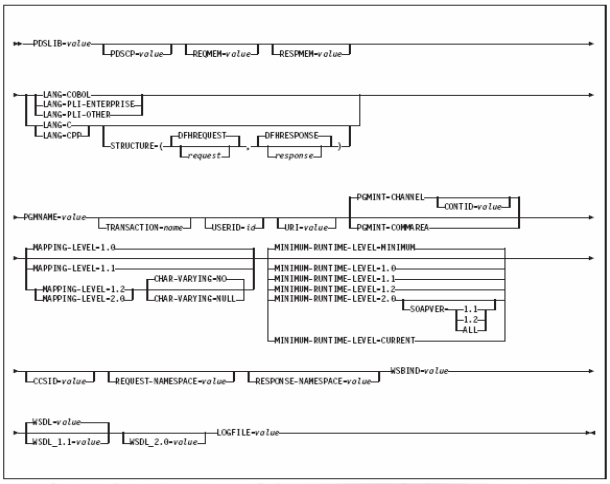
JCL to specify parameters

```
//JAVAPROG EXEC DFHLS2WS,
LOGFILE=/u/myuserid/wsbind/ls2ws.log
WSDL=/u/myuserid/wsdl/temp.wsdl
PGMNAME=PETS
URI=/reqpetsURI
PGMINT=CHANNEL
CONTID=mycontname
LANG=COBOL
WSBIND=/u/myuserid/wsbind/reqpet.wsbind
PDSLIB=//MYUSERID.COPYBOOK
REQMEM=INPUT01
```



The foil shows an overview of the batch process when starting from a language structure (or several language structures) and converting them into a WSDL document. The WSDL generated by the CICS tooling will always be Document literal.

The WSBind file is also produced. The syntax diagram below is for CICS TS V3.2.



IMPACT IBM

"Top down" approach to web services

- A supplied WSDL definition of a web service is to be implemented as a CICS application
  - Use CICS supplied batch procedure (DFHWS2LS) to convert the WSDL into a language structure.
    - Use the generated language structure in a CICS application program.
    - A file called the WSBind file is also produced.

19

IMPACT IBM

The WSDL may be obtained from UDDI or by other means. The CICS tooling can handle DOC literal, RPC literal or wrapped Doc literal forms of WSDL as input. The outputs from the batch procedure are a WSBind file, as before, and a language structure which maps the data definitions from the WSDL into a structure in the specified high level programming language (COBOL, PL/I, C or C++).

At CICS TS V3.1, the supplied WSDL must be WSDL 1.1. At CICS TS V3.2, the supplied WSDL can be either WSDL 1.1 or WSDL 2.0.

20

IMPACT IBM

Batch Processing (DFHWS2LS)

WSDL

```
<xsd:element name="SOME-MESSAGE" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="79"/>
      <xsd:whiteSpace value="preserve"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="NAME" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="20"/>
      <xsd:whiteSpace value="preserve"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

JCL to specify parameters

```
//JAVAPROG EXEC DFHWS2LS,
LOGFILE=/u/myuserid/wsbind/wstest1.log
WSDL=/u/myuserid/wsdl/wstest1.wsdl
BINDING=WSTEST1HttpSoapBinding
PGMNAME=WSTEST1
URI=/wstest1
PGMINT=COMMAREA
LANG=COBOL
WSBIND=/u/myuserid/wsbind/wstest1.wsbind
PDSLIB=//MYUSERID.COPYBOOK
REQMEM=WS1IN
```

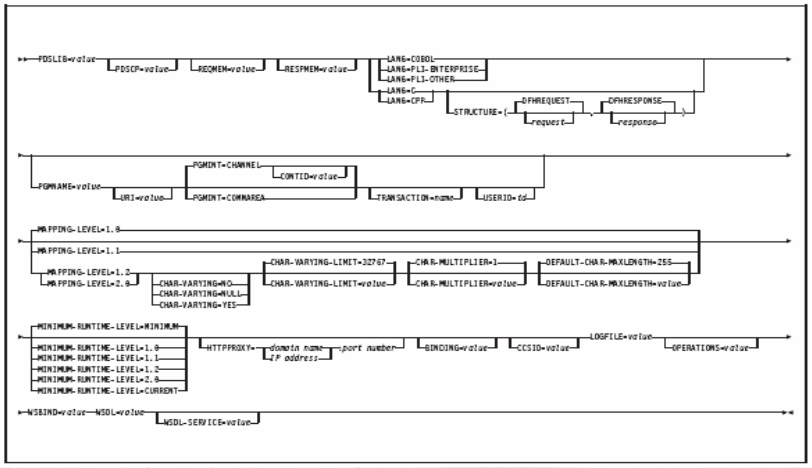
WSBind file

Language structure(s)

IMPACT IBM

The foil shows an overview of the batch process when starting from WSDL and converting into a language structure. The language structure can be COBOL, PL/I, C or C++. The CICS tooling can accept RPC literal, DOC literal or wrapped DOC literal WSDL as input.

The WSBind file is also produced. The syntax diagram below is for CICS TS V3.2.



IMPACT IBM

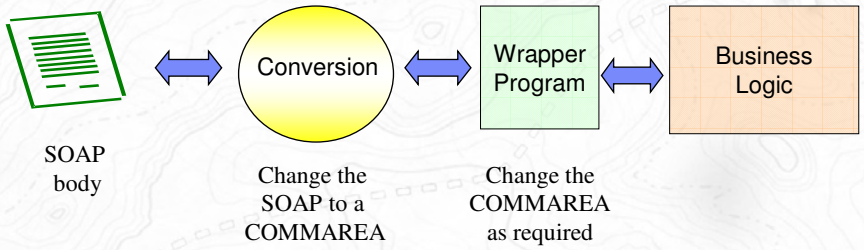
Meet in the middle

- Pure 'top down' or 'bottom up' will not be suitable in all situations.
- In such situations, a wrapper program may provide a solution
  - if the language structure uses data types not supported by the utility tools
    - a wrapper program may be used to map commarea to a supported data type
  - when there are unnecessary fields in the language structure which you do not want to expose externally
    - a wrapper program can be used to hide unnecessary fields
  - when an existing piece of WSDL is to be used with an existing program
    - the WSDL and the program may not match exactly. A wrapper program could perform some intermediate mappings.

IMPACT IBM

Sometimes a pure 'top down' or 'bottom up' approach will not be satisfactory. The foil lists some circumstances when is might be desirable to have a wrapper program between the data mapping and the business logic.

Where a wrapper program fits in

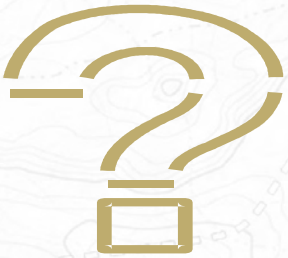


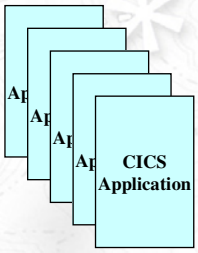
The foil shows where a wrapper program fits into the scheme of things. It sits between the conversion operation and the business logic. It is then able to perform data manipulation either on the way in, on the way out or both ways.

IMPACT IBM

CICS Resource Definitions

Soap
Message





How does a SOAP message get to the required CICS application?

IMPACT IBM

The next section of the presentation describes and explains the CICS resource definitions which need to be installed so that the runtime can operate. How does a SOAP message get to the required CICS application?

IMPACT IBM

Decide on the Transport

- SOAP messages to and from CICS may use
  - HTTP
  - WMQ
- Once the decision has been taken, define either a TCPIPSERVICE or a WMQ

29

IMPACT IBM

The first thing to decide is what transport is to be used to carry the SOAP message. If it is to be HTTP, the a TCPIPSERVICE definition is required. If the transport is to be WMQ, then an appropriate definition is required for that.

30

TCPIPService

- TCPIPSERVICE definition
  - Required when CICS is a service provider
  - URIMAP matching will occur when protocol (HTTP) is specified
    - Some parameters ignored
    - URM
    - Transaction

A TCPIPSERVICE definition is required when CICS is the service provider and the chosen transport is HTTP. That is, the TCPIPSERVICE definition is only required for inbound requests.

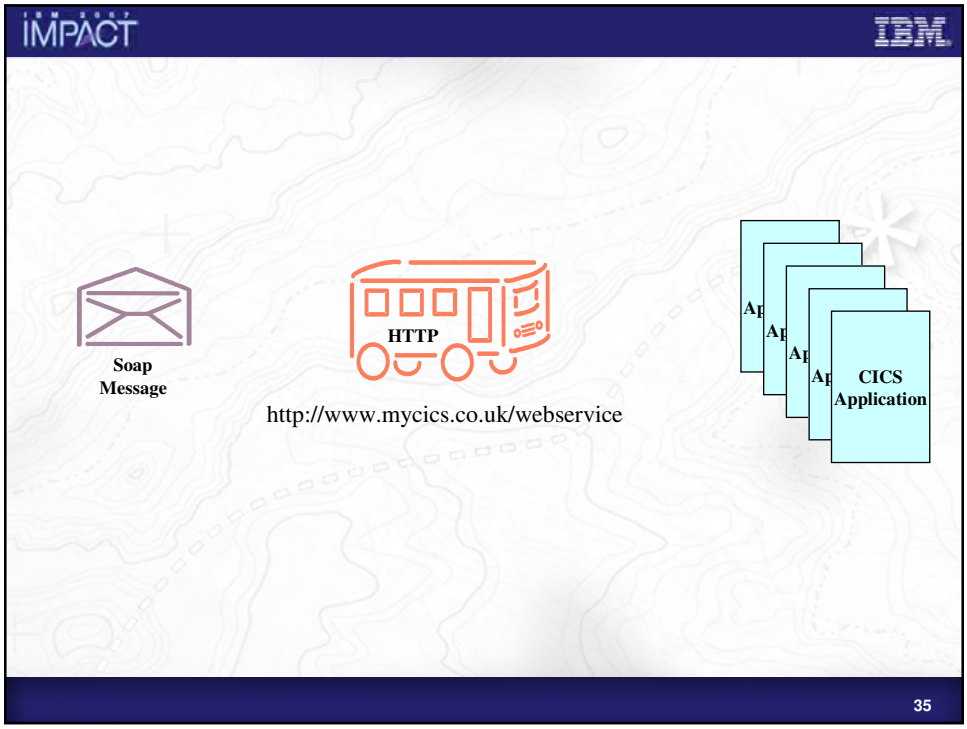
When a TCPIPSERVICE definition is used with protocol HTTP, the URIMAP definitions will be matched against the URI. If a match is found (it better be for Web Services) then some parameters will be taken from the URIMAP definition instead of the TCPIPSERVICE definition.

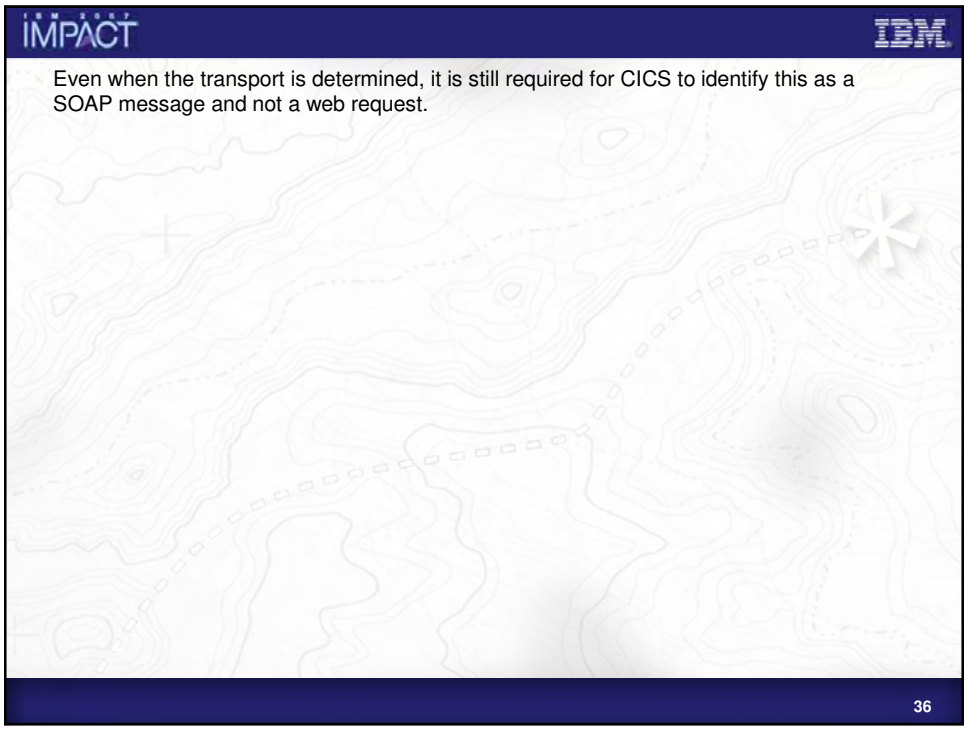
WMQ

- WMQ definition
  - Required when CICS is a service provider
  - Pick the target up from the RFH2 header if present, otherwise default to trigger data

DEFINE
QLOCAL('*queuename*')
DESCR('*description*')
PROCESS('*processname*')
INITQ('*initqueue*')
TRIGGER
TRIGTYPE(FIRST)
TRIGDATA('*path part of URI*')
BOTHRESH(*nnn*)
BOQNAME('*requeuename*')

A local WebSphere MQ queue definition is required when CICS is the service provider and the chosen transport is WMQ.


IMPACT
IBM
Soap
Message
HTTP
http://www.mycics.co.uk/webservice
CICS
Application
35


IMPACT
IBM
Even when the transport is determined, it is still required for CICS to identify this as a SOAP message and not a web request.
36

Choose class of service

- CICS needs to select a number of things based on the URI
  - Class of service to be provided
  - Which application program (Web service) is being invoked
- This is achieved via a CICS resource called the URIMAP

CICS needs to select a number of things based only upon the URI received. The class of service to be used is implemented via a new CICS RDO object known as a PIPELINE. The Web service being invoked is actually an application program. Both of these items of information are derived from the URI by using a definition new to CICS TS V3.1 known as a URIMAP definition.

IMPACT IBM

URIMAP

- URIMAP definition
  - Locates the Web Service and the Pipeline resources required to process the request
  - USAGE (PIPELINE)
    - Specifies a Web Service request
  - Matching the request URI
    - HOST (www.mycics.co.uk)
    - PATH (web_service_identifier)
  - TCPIPSERVICE
    - Restricts matching to a single port
  - PIPELINE
    - Names the Pipeline to process this request
  - WEBSERVICE
    - Names the associated Web Service for this request
  - TRANSACTION
    - Alias transaction for the Web Service

```
    Urimap        =>
    Group       =>
    DEscription   =>
    STatus        => Enabled    Enabled  | Disabled
    USAge         => Server     Server  | Client | Pipeline
UNIVERSAL RESOURCE IDENTIFIER
    SCheme        => HTTP       HTTP  | HTTPS
    HOST        =>
    (Mixed Case) =>
    PAth          =>
    (Mixed Case) =>
                  =>
                  =>
                  =>
ASSOCIATED CICS RESOURCES
    TCpipservice  =>
    Analyzer    => No           No  | Yes
    COnverter     =>
    TRansaction   =>
    PRogram       =>
    PIpeline      =>
    Webservice    =>                                (Mixed Case)
```


IMPACT IBM

The URIMAP definition is used to match a URI to a WEBSERVICE definition and a PIPELINE definition. You should have a unique URI for each Web Service that you want to use in CICS.

The parameters that apply to a Web Service form of the URIMAP are:

USAGE (PIPELINE): This indicates that the URIMAP definition is applicable to a web service and that the PIPELINE and WEBSERVICE parameters must be specified.

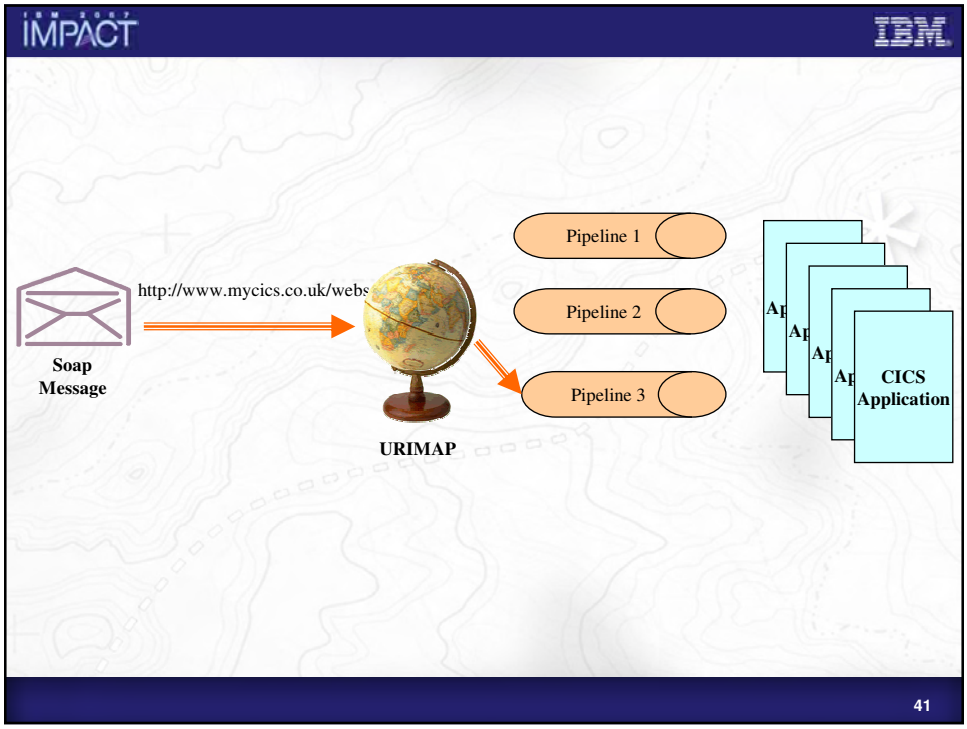
The SCHEME, HOST and PATH values must be specified to allow matching of the URI. A URI, such as,http://www.mycics.co.uk/webservice would be decomposed to SCHEME (http), HOST (www.mycics.co.uk) and PATH (webservice)

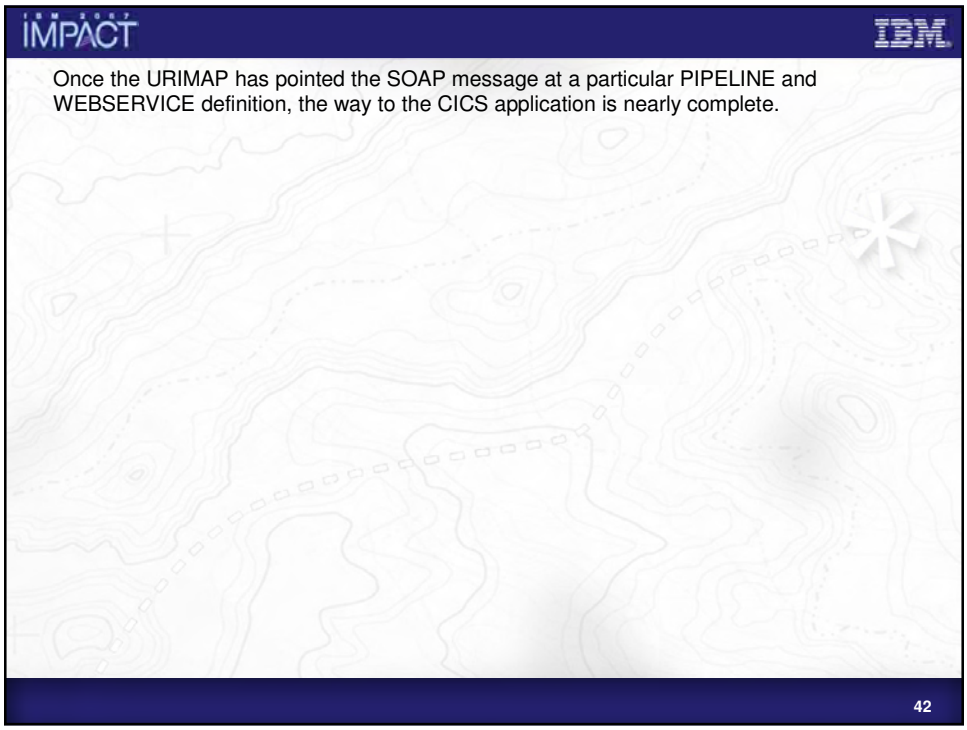
TCPIPSERVICE is optional on the URIMAP definition for USAGE (PIPELINE). If a named TCPIPSERVICE is specified then only requests from that specific port will be matched against this URIMAP definition.

The PIPELINE parameter names an installed PIPELINE resource which will be used to determine the processing nodes or message handlers that will be invoked for this Web Service request.

The WEBSERVICE parameter names an installed WEBSERVICE requests that defines the execution environment that lets a CICS application program operate as a Web service provider or requester.

The TRANSACTION parameter specifies the 1-4 character name of an alias transaction that is to be used to run the user application that composes a response to the web service request.


IMPACT
IBM
Soap
Message
http://www.mycics.co.uk/webs
URIMAP
Pipeline 1
Pipeline 2
Pipeline 3
CICS
Application
41


IMPACT
IBM
Once the URIMAP has pointed the SOAP message at a particular PIPELINE and WEBSERVICE definition, the way to the CICS application is nearly complete.
42

IMPACT IBM

PIPELINE

- What is a CICS pipeline?
  - Responsible for dealing with SOAP headers
  - Implemented as a series of programs
  - Can be configured by end user by using message handlers

Pipeline 3

IMPACT IBM

A PIPELINE resource definition is used when a CICS application is in the role of a Web service provider or requester. It is the responsibility of the PIPELINE to handle the SOAP headers. In general, it will not need to change the SOAP body element at all. It is in the PIPELINE where it is possible to configure special processing which needs to be performed for the Web services deployed in the PIPELINE. For example, logging or security can be configured into the PIPELINE.

IMPACT IBM

PIPELINE resource

- PIPELINE definition
  - Defines the processing nodes for a web service request
    - Different pipelines for:
      - Requester and provider
  - CONFIGFILE
    - HFS file that contains information about the processing nodes that will act on a service request and on the response
  - SHELF
    - HFS directory for CICS use
  - WSDIR
    - Name of the Web service binding directory
      - HFS pickup directory

```
PIPELINE      ==>
   Group          ==>
   Description    ==>
   Status         ==>
   Configfile     ==>
   (Mixed Case)   ==>
                  ==>
                  ==>
                  ==>
   Shelf          ==>
   (Mixed Case)   ==>
                  ==>
                  ==>
                  ==>
   Wsdir          ==>
   (Mixed Case)   ==>
                  ==>
                  ==>
                  ==>
```


IMPACT IBM

A PIPELINE resource definition is used when a CICS application is in the role of a Web service provider or requester. It provides information about the processing nodes which will act on a service request and on the response. Typically, a single PIPELINE definition defines an infrastructure that can be used by many applications. There will be separate configuration files for CICS applications acting as a service provider and service requester.

The information about the processing nodes is supplied indirectly: the PIPELINE specifies the name of an HFS configuration file (CONFIGFILE) which contains an XML description of the nodes and their configuration.

The SHELF is an HFS directory where CICS will copy information about installed Web Services. CICS regions into which the PIPELINE definition is installed must have full permissions to the shelf directory--read, write, and the ability to create subdirectories. A single shelf directory may be shared by multiple CICS regions and by multiple PIPELINE definitions. Within a shelf directory, each CICS region uses a separate subdirectory to keep its files separate from those of other CICS regions. Within each region's directory, each PIPELINE uses a separate subdirectory. After a CICS region performs a cold or initial start, it deletes its subdirectories from the shelf before trying to use the shelf. You should not attempt to modify the contents of a shelf that is referred to by an installed PIPELINE definition. If you do, the effects are unpredictable

The Web service binding directory (WSDIR) contains Web service binding files that are associated with a PIPELINE, and that are to be installed automatically by the CICS scanning mechanism. When the PIPELINE definition is installed, CICS scans the directory and automatically installs any Web service binding files it finds there. Note that this happens regardless of whether the PIPELINE is installed in enabled or disabled state. A CEMT PERFORM PIPELINE SCAN command can be used to force CICS to scan the Web Service binding directory.

An inbound Web service request (that is, a request by which a client invokes a Web service in CICS) is associated with a PIPELINE resource by the URIMAP resource. The URIMAP identifies the PIPELINE resource that applies to the URI associated with the request; the PIPELINE specifies the processing that is to be performed on the message.

IMPACT IBM

Pipeline Configuration

- Pipeline configuration file
  - XML file that describes:
    - The mandatory <service> and optional <transport> elements
    - The sequence of message handlers to be invoked
  - Different applications will require different configuration files
    - Service provider
    - Service requester
    - SOAP 1.1
    - SOAP 1.2
    - User message handlers
      - e.g. Extract USERID from the message

IMPACT IBM

The Pipeline configuration file, named in a PIPELINE resource definition, is used to describe the series of message handlers (i.e. the pipeline) to process the request. The configuration file is an XML document, stored in HFS and can be edited with any XML editor.

The configuration file will contain mandatory <service> and optional <transport> elements along with application handler <apphandler> and a service parameter list <service_parameter_list>.

Different applications will require different configuration files. There are different pipeline configurations necessary for a service provider and service requester as well as different configurations for processing SOAP 1.1 and 1.2 messages. CICS provides the configuration files necessary for CICS to function as both a service requester and a service provider handling both SOAP 1.1 and 1.2 messages.

The configuration file can also be used to add your own user message handlers. An example would be a user message handler to extract user identification from the message to determine which USERID and transaction id should be used to process the message.

IMPACT IBM

WEBSERVICE resource

- WEBSERVICE definition
  - Defines the application specific details for a web service request
    - Defines the execution environment for Web Service application
  - PIPELINE
    - Name of the pipeline where this WEBSERVICE is to be installed
  - WSBIND
    - HFS name of the WS Binding file
  - WSDLFILE
    - HFS name of the WSDL file
  - VALIDATION
    - Run time SOAP message validation against WSDL schema

```
WEBSERVICE       ==>
   Group         ==>
   Description   ==>
   Pipeline      ==>
   (Mixed Case)  ==>
                 ==>
                 ==>
                 ==>
   WSBIND        ==>
   (Mixed Case)  ==>
                 ==>
                 ==>
                 ==>
   WSDLFILE      ==>
   (Mixed Case)  ==>
                 ==>
                 ==>
   VALIDATION    ==>  NO NO|YES
```


IMPACT IBM

A WEBSERVICE resource defines the execution environment that lets a CICS application program operate as a Web service provider or requester. The Web service interaction in which the CICS application participates uses SOAP messaging, and is formally described with Web service description language (WSDL).

The execution environment contains three components that are specified in the WEBSERVICE attributes:

A pipeline
A Web service binding file
A Web service description

Although CICS provides the usual resource definition mechanisms for creating WEBSERVICE resources, and installing them in your CICS region, there is an alternative strategy which you can use. You can use the scanning mechanism to install WEBSERVICE resources in your running CICS region.

Validation: Specifies whether full validation of SOAP messages against the corresponding schema in the Web service description should be performed at run-time. Validating a SOAP message against its schema incurs considerable processing overhead, and you should normally specify VALIDATION(NO). Full validation ensures that all SOAP messages which are sent and received are valid XML with respect to the XML schema. If VALIDATION(NO) is specified, sufficient validation is performed to ensure that the message contains well-formed XML.

IMPACT IBM

CICS Web Services Definitions

- Resource Definitions
  - Define the transport
    - http: TCPIPSERVICE for inbound requests
    - wmq: QLOCAL definition
  - Find the Web Service
    - URIMAP definition
  - Define the qualities of service
    - PIPELINE definition
  - Define the Web Service execution environment
    - WEBSERVICE definition

IMPACT IBM

There are a number of interrelated resource definitions required to process a Web Service in CICS TSV3.1.

A resource definition is required to define the transport. Both http and WebSphere MQSeries can be used as transports. For http, a CICS TCPIPSERVICE definition is required. For WMQ, a request queue must be defined with a QLOCAL definition.

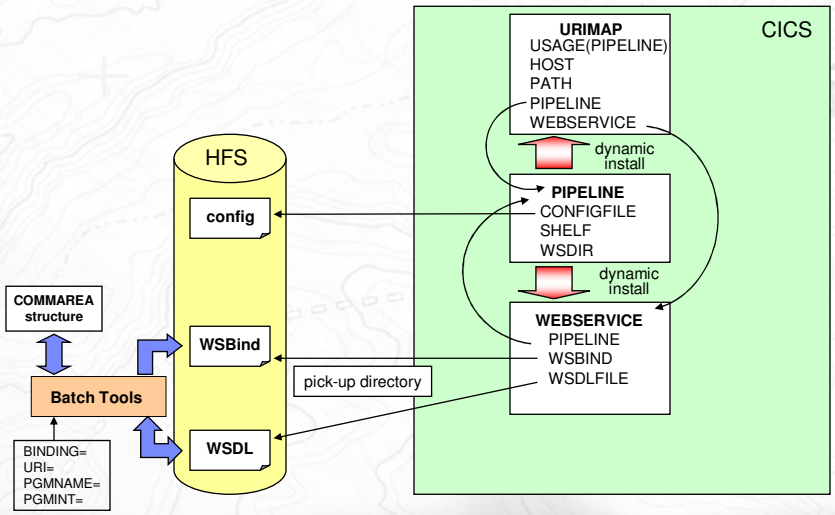
Next CICS must determine which Web Service is required. CICS TS V3.1 uses a URIMAP definition to map the incoming Universal Resource Identifier (URI) to a specific WEBSERVICE definition. The associated PIPELINE definition is determined from the matching URIMAP definition.

The PIPELINE definition is used to specify which processing nodes or message handlers are to operate on a Web Service request.

The WEBSERVICE definition is used to specify how CICS is to execute the application. The WSBIND file, specified in the WEBSERVICE definition, is used to tell CICS which application program to execute, whether a COMMAREA or CHANNEL is used and how CICS is to transform the message between an XML format and COMMAREA format.

IMPACT IBM

Web Service Resource Interrelationships



IMPACT IBM

This chart shows the interrelationships between the CICS resource definitions necessary to support Web Services.

The CICS WSDL utility will produce a WSDL file from a language structure (copybook) or a language structure from WSDL. As part of the generation process a Web Services Binding file (WSBIND) will be produced. The WSBIND file contains information about the CICS program to be invoked, the name of the WSDL file, the local URI and information necessary to populate a COMMAREA from XML and vice versa. Both the WSBIND and the WSDL file will be used by the executing CICS region.

The URIMAP definition will name both the PIPELINE definition and the WEBSERVICE definition. Optionally, the URIMAP can specify an installed TCPIPSERVICE name to restrict the matching to information for the specific port named in that resource definition.

The PIPELINE resource definition will copy installed WEBSERVICE definitions to its SHELF. The WEBSERVICE definitions can be dynamically created through the use of the pick-up directory (WSDIR).

The WEBSERVICE definition will name the PIPELINE definition that contains the configuration information (CONFIGFILE) on which message handlers are invoked when processing this Web Service.

IMPACT IBM

Setting up the Resources

- Define a TCPIPSERVICE (or WMQ) and a PIPELINE
- Then either
  - install the PIPELINE definition or
  - issue CEMT PERFORM PIPELINE SCAN
- CICS uses the PIPELINE definition to
  - locate the WSBind file
  - from the WSBind file, CICS will dynamically create a WEBSERVICE resource
  - CICS will also dynamically create a URIMAP definition
- Can define everything individually if preferred

55

IMPACT IBM

Setting up the CICS resources is not as difficult as it might seem. It is only necessary to define the TCPIPSERVICE (or WMQ) and the PIPELINE. Place the WSBind file generated from the batch tooling into the HFS directory specified in the PIPELINE definition. Then either install the PIPELINE definition or issue a CEMT PERFORM PIPELINE SCAN command.

The PIPELINE definition contains the directory name where the WSBind file can be found. From the WSBind file, CICS will dynamically create the Web Service resource definition. This provides CICS with enough information to be able dynamically to create a URIMAP definition as well. So, as long as you create a valid PIPELINE definition and put the WSBind file in the correct location, CICS will do the rest.

The necessary definitions can all be input and installed manually if preferred. The definitions can be put into a group and the group installed as for any CICS resource.

56

Runtime Support

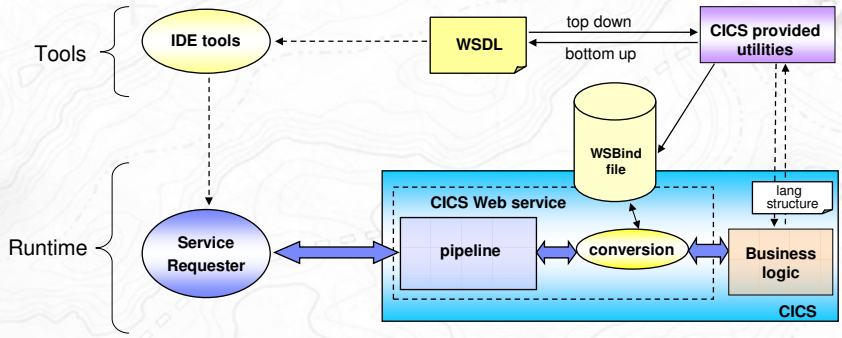
This section of the presentation describes the CICS runtime support for web services.

Tooling to Runtime (the connection)

- CICS provides the necessary tools and runtime for web services
  - A utility can generate WSDL from language structures
    - *a bottom up approach from an existing application*
  - A utility can generate language structures from WSDL
    - *a top down approach to new CICS service provider or requester programs*
  - XML to language structure (e.g. COMMAREA) conversion and vice versa at runtime
  - The link between the utilities and the runtime is via the **WSBind** file

CICS TS V3.1 provides both the tools to prepare applications for becoming web services and the runtime support for them. The 'connection' between the tooling and the runtime, apart from the language structures which the application programs use, is via the WSBind file.

Web Services (the complete picture)



The foil attempts to show the separate components of the batch/tooling environment and the CICS runtime environment. The entity which connects the two is the WSBind file. This is generated by the CICS tooling and utilised by the runtime.

IMPACT IBM

Runtime Scenarios

- CICS can be the service provider
  - traditional situation. CICS is the server in a client/server scenario. A client sends a request in to CICS.
- CICS can be the service requester
  - this is where CICS is the client in the client/server scenario. CICS is sending a request to execute a webservice to an external service provider.

63

IMPACT IBM

There are two fundamental roles which CICS can take in implementing web service support. CICS can be the service provider. This is the traditional CICS role where it is the server running transactions and an external client is requesting work to be performed. CICS can also be the service requester. This is where CICS is requesting an external provider of service to perform work on behalf of a CICS application.

64

IMPACT IBM

What happens? (Service Provider)

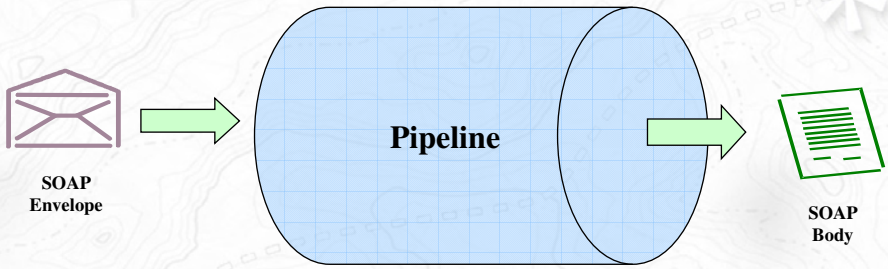
- The URIMAP is used to locate the webservice which needs to be invoked based on the URL.
- The pipeline passes a number of containers in the pipeline channel .
  - One container contains the SOAP body to be processed

IMPACT IBM

The pipeline code passes a number of containers to the runtime. These provide the information required so that the runtime can correctly invoke a CICS application which will service the request.

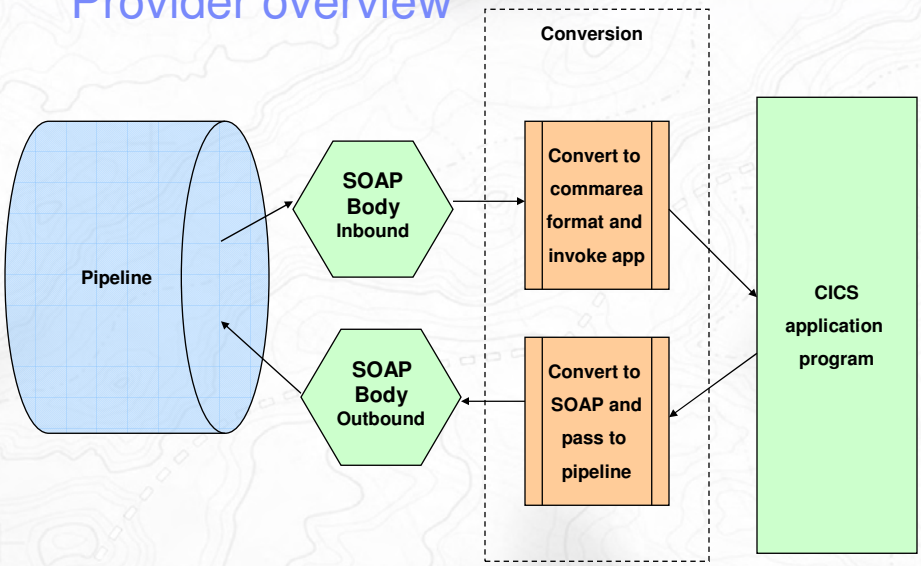
Containers are named blocks of data designed for passing information between programs. You can think of them as "named communication areas (COMMAREAs)". Programs can pass any number of containers between each other. Containers are grouped together in sets called **channels**. A channel is analogous to a parameter list.

Pipeline Processing



The pipeline receives the SOAP message from the transport. This will be the complete envelope with headers and the body. The pipeline will process any header handling routines and remove the envelope. Assuming that there are no errors, it will pass the SOAP body on for further processing.

Provider overview



The foil shows, at a high level, the general flow of data through the system when CICS is acting as the service provider. It shows the main actions performed and how the application program fits into the picture.

IMPACT IBM

What happens? (Service Requester)

- The CICS application which invokes a web service must
  - place the outbound data into container DFHWS-DATA
  - call the operation required and specify the webservice which implements the operation
  - handle the response which comes back in container DFHWS-DATA
- The process is basically the reverse of the provider situation in terms of the conversions are performed.

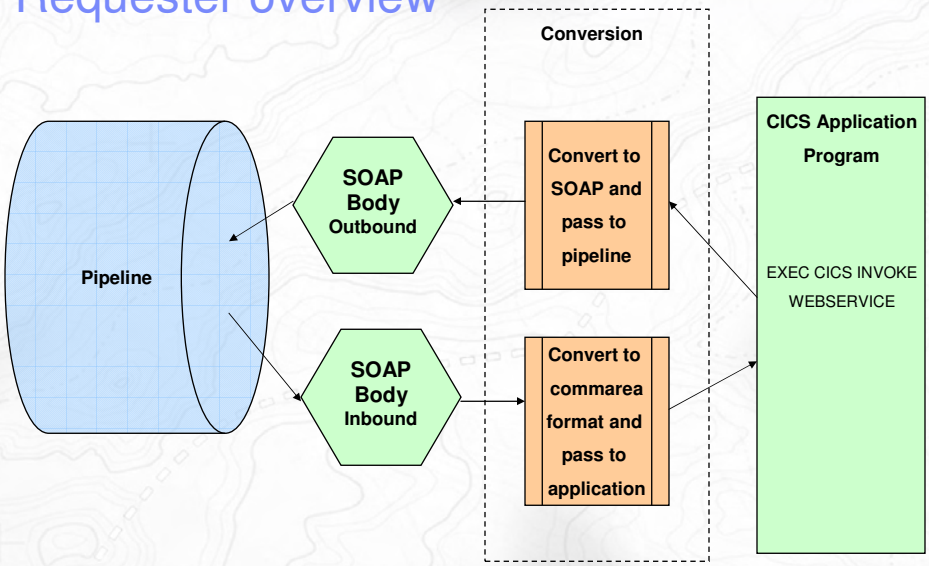
71

IMPACT IBM

CICS can act as a service requester by an application program issuing an EXEC CICS INVOKE WEBSERVICE call. The outbound data are placed into container DFHWS-DATA. Any response is placed into the same container.

72

Requester overview



The foil summarises the runtime operations of the CICS code, when CICS is a service requester, at a very high level. This foil also assumes that the service is being provided remotely.

IMPACT IBM

Overview of New things in CICS TS V3.2

IMPACT IBM

Now we look at some of the new support added in CICS TS V3.2. Everything which was supported in CICS TS V3.1 continues to be supported, but extra capability has been added in 3.2.

IMPACT IBM

MTOM/XOP Support

- In standard SOAP messages:
  - Binary objects are base64 encoded
    - Significantly increases their size
  - Included in the message body
    - Can impact transmission time
    - Also impacts message parse time (CPU)
- MTOM/XOP provides a solution to this problem
  - The MTOM specification
    - Defines a method for optimizing SOAP messages
      - Separates out binary data
      - Sends it in separate binary attachments using a MIME Multipart/Related message
  - The XOP specification
    - Defines an implementation for optimizing XML messages
      - Uses binary attachments in a packaging format
        - Includes but is not limited to MIME messages

IMPACT IBM

In standard SOAP messages, binary objects are base64 encoded and included in the message body. This significantly increases their size, and for very large binary objects, this can impact transmission time. Implementing MTOM/XOP provides a solution to this problem.

The SOAP Message Transmission Optimization Mechanism (MTOM) and XML-binary Optimized Packaging (XOP) specifications, often referred to as MTOM/XOP, define a method for optimizing the transmission of large base64binary data objects within SOAP messages.

The MTOM specification conceptually defines a method for optimizing SOAP messages by separating out binary data, that would otherwise be base64 encoded, and sending it in separate binary attachments using a MIME Multipart/Related message. This type of MIME message is called an *MTOM message*. Sending the data in binary format significantly reduces its size, thus optimizing the transmission of the SOAP message.

The XOP specification defines an implementation for optimizing XML messages using binary attachments in a packaging format that includes but is not limited to MIME messages.

The size of the base64binary data is significantly reduced because the attachments are encoded in binary format. The XML in the SOAP message is then converted to XOP format by replacing the base64binary data with a special <xop:Include> element that references the relevant MIME attachment using a URI.

MTOM/XOP Configuration

- Support for MTOM/XOP is configured through the pipeline configuration file.
- When using MTOM/XOP, new containers are present in the pipeline channel at runtime.
- MTOM/XOP can be used with CICS as either a provider or a requester.

An example pipeline configuration file for MTOM/XOP is shown here:-

```
An example pipeline
    <service_handler_list>
     <cics_mtom_handler>
       <cics_mtom_handler_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" />
       </cics_mtom_handler_configuration>
     </cics_mtom_handler>
    </service_handler_list>
```

The extra elements are added to the pipeline configuration file prior to installing the pipeline. New containers are also present in the pipeline channel. These are summarized below but for full details, see the CICS infocenter.

DFHWS-CID-DOMAIN
: Contains the domain name that is used to generate content-ID values for referencing binary attachments

DFHWS-MTOM-IN
: Holds information about the MTOM options for the pipeline and information about the message format received

DFHWS-MTOM-OUT
: Holds information about the MTOM options for the pipeline and information about what XOP processing should take place

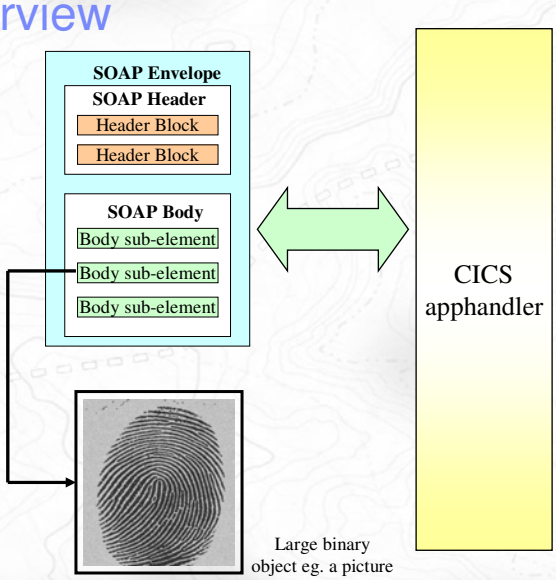
DFHWS-XOP-IN
: Holds information about the binary attachments and their containers

DFHWS-XOP-OUT
: Holds information about the containers and their binary attachments

MTOM/XOP Overview

Large binary objects are separated from the rest of the SOAP message and are referenced from within the message.

- Reduces message size
  - No need for base64Binary
- Reduces transmission time
  - Since message is smaller
- Reduces message parse time
  - No need to parse Mbtyes of binary data



CICS implements support for these specifications in both requester and provider pipelines. As an alternative to including the base64binary data directly in the SOAP message, CICS applications that are deployed as Web service providers or requesters can use this support to send and receive MTOM messages with binary attachments.

You can configure this support by using additional options in the pipeline configuration file.

There are certain scenarios where CICS cannot support the XOP document format in MTOM messages directly. For example, the Web Services security functionality and Web services validation cannot parse the <xop:Include> elements in the XOP document. Therefore, two modes of support are provided in the pipeline to handle XOP documents and any associated binary attachments.

If the application handler program is capable of supporting XOP documents, such as the standard handlers that are provided when you deploy a Web service using the Web services assistant, then CICS performs XOP processing in direct mode. If you are using a different application handler in the pipeline that is not capable of handling XOP documents, all XOP processing is performed in compatibility mode.

If you are using the Web Services Security functionality or are testing with validation switched on, all XOP processing is performed in compatibility mode even if you have specified direct mode in the pipeline configuration file.

IMPACT IBM

WSDL 2.0 Tooling support

- Web Service Assistants changes
  - DFHLS2WS new options
    - WSDL_1.1(<HFS filename location>)
    - WSDL_2.0(<HFS filename location>)
    - SOAPVER(1.1|1.2|ALL)
    - URI parameter may now specify an relative or absolute URI
  - DFHWS2LS new options
    - Automatically determines the WSDL version
    - OPERATION=value
      - Specifies the subset of valid operations that are required for a requestor
      - Used to limit the size of the WSBIND file
    - WSDL-SERVICE=value
      - Specifies the wsdl:Service element to be used when there is more than one Service element for a Binding element

IMPACT IBM

New parameters are available on the DFHLS2WS batch job so that it is possible to specify the version of WSDL to be generated from a supplied language structure.

DFHWS2LS now automatically determines the WSDL version of Web service description that has been supplied as input. The batch job has been enhanced to provide you with more flexibility in how to handle the Web service description.

wsdl:Bindings elements can be associated with multiple wsdl:Service elements in Web service descriptions. A new parameter has been added to enable you to select a specific Service element within the Web service description.

When you are creating a service requester application, you can now specify a subset of wsdl:Operation elements that you want to implement and create a Web service binding file based on that subset. This can be useful when you have a very large WSDL file. By only using a subset of Operation elements, you can save on storage by generating a smaller Web service binding file.

IMPACT IBM

WSDL 2 MEPs

- Message exchange patterns supported
  - In-Only
    - CICS as the provider
      - CICS will receive a message and send no response
    - CICS as the requester
      - CICS application will send a message and expect no response
  - In-out
    - CICS as the provider
      - CICS will receive a message and respond with a normal response or fault
    - CICS as the requester
      - CICS application will send a message and expect a normal response of fault

NB. These are supported in CICS TS V3.1 but they were not called MEPs then.

IMPACT IBM

In-only with CICS as provider This pattern is where CICS receives a message and must not return anything to the requester even if something goes wrong. This pattern is supported already and will continue to be supported in the same way. CICS Web Service support puts the DFHNORESPONSE container into the SOAP handler channel to indicate that the pipeline must not send anything to the requester.

In-only with CICS as requester This pattern is where CICS as a requester of service will send a message to a service provider and receive no response. This situation is supported already. The tasksending the message knows that no response is to expected, so it will not wait for one.

In-out with CICS as provider In this pattern, CICS will receive a message from a requester and will respond with either a normal response or with a fault message. This is a normal flow of messages for a web service and very much in the standard CICS application pattern. It is already supported and will continue to be so.

In-Out with CICS as requester In this pattern, CICS will send a message to a service provider and will receive a response, which may either be a normal response or a fault message. Again, this is a natural set of messages for a CICS application. The pattern is already supported and will continue to be so.

IMPACT IBM

WSDL 2 specific MEPs

- Message exchange patterns supported…
  - Robust in-only
    - CICS as the provider
      - CICS will receive a message and respond only if an error occurs
    - CICS as the requester
      - CICS application will send a message and expect a response only if an error occurs
        - New timeout specification on the PIPELINE definition
  - In-optional-out
    - CICS as the provider
      - CICS will receive a message and may respond with
        - A normal response
        - An error response
        - Nothing (no response)
    - CICS as the requester
      - CICS application will send a message and expect:
        - A normal response
        - An error response
        - Nothing (no response)

These patterns are new for CICS TS V3.2

IMPACT IBM

Robust in-only with CICS as provider CICS as the service provider will receive a message from the requester. CICS only needs to respond if an error occurs. If an error occurs in the pipeline, a SOAP fault will be sent back to the requester.

Robust in-only with CICS as requester If CICS is the service requester in a MEP where a response of some sort may or may not be received, then a timeout needs to be specified to define how long CICS is to wait for any possible response. A new timeout parameter has been added to the PIPELINE resource. The value specified is stored in binary form in a new container called DFHWS-RESPWAIT. The value specifies the timeout value to use in seconds. This is so as to allow the value to be interrogated and perhaps changed by handlers in the PIPELINE if desired.

In-optional-out CICS as provider CICS as a provider with this MEP will receive a message from the requester and then may send a normal response, may send an error response or may send nothing back to the requester. Which option will occur is not known until runtime. The application program will need to indicate to DFHPITL that it does not intend to send a response by deleting the DFHWS-DATA container from the channel.

In-optional-out CICS as requester If CICS is the requester of service, it will send a message to the service provider. The provider may respond with a normal response, an error response or never respond at all. The situation is very similar to that for the robust in-only pattern with CICS as the requester. The question is, how long does CICS wait for the optional response from the provider? The solution is to use the timeout value again for this situation.

WS-Trust

- Submitted to OASIS standardization process
  - Used W3C specification dated 25 February 2005 as input
- Provides a framework for building trust relationships
  - Sender and Receiver in different security domains
  - Security tokens must be vouched for by trusted third party
  - Trusted third party, called Security Token Service (STS)
- WS-Trust defines standard protocols and standard WSDL interfaces to communicate with an STS

The Web Services Trust Language specification enhances Web Services Security further by providing a framework for requesting and issuing security tokens, and managing trust relationships between Web service requesters and providers. This extension to the authentication of SOAP messages enables Web services to validate and exchange security tokens of different types using a trusted third party. This third party is called a Security Token Service (STS).

CICS Support of WS-Trust

- Interoperate with a Security Token Server
  - CICS supplied security handler
    - Inbound messages
      - Validate the security token in the WS-Security header
      - Exchange the security token in the WS-Security header
    - Outbound messages
      - Exchange the security token to be used in the WS-Security header
- Trust Client Interface
  - User supplied custom message handler
    - No requirement for the CICS provided security handler
  - Directly interact with an STS
    - Issue or validate security tokens from the message header
  - Channel and container interface

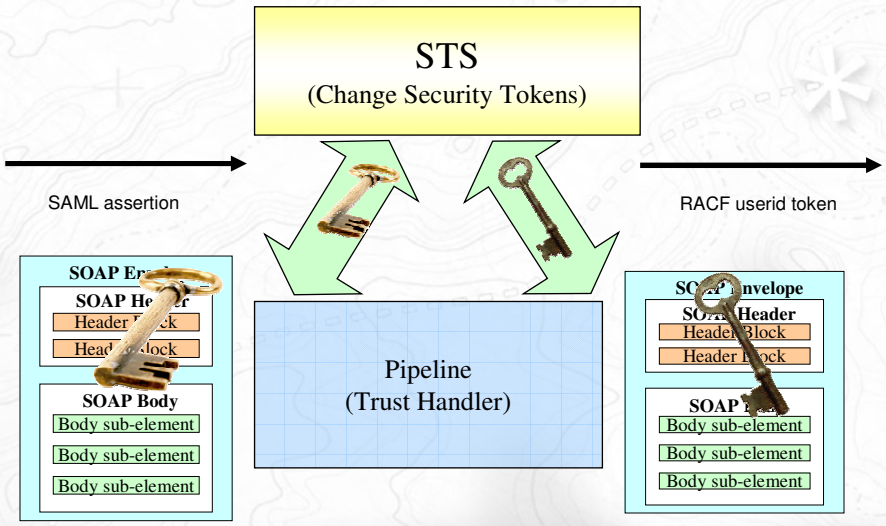
CICS support for securing Web services has been enhanced to include an implementation of the Web Services Trust Language (or WS-Trust) specification.

CICS can now interoperate with a Security Token Service (STS), such as Tivoli Federated Identity Manager, to validate and issue security tokens in Web services. This enables CICS to send and receive messages that contain a wide variety of security tokens, such as SAML assertions and Kerberos tokens, to interoperate securely with other Web services.

You can configure the CICS-supplied security handler to define how CICS should interact with an STS. The <wsse_handler> element in the pipeline configuration file now includes additional elements and attributes to configure this support. CICS can either validate or exchange the first security token or the first security token of a specific type in the message header. If you want more sophisticated processing to take place,

CICS provides a separate Trust client interface that you can use in a custom message handler. You can use the Trust client instead of the security handler or in addition to it.

WS-Trust Overview



The foil attempts to show in a schematic fashion what the function of WS-Trust and the STS are. A message inbound to CICS may have security credentials which CICS cannot handle, such as a SAML assertion. By using the Trust handler in the pipeline, CICS can request an STS, such as TFIM, for alternative credentials which CICS is able to understand. Typically this would be a RACF userid token. The STS is able to understand and issue credentials as requested and CICS must trust the STS to issue valid credentials.

If CICS needs to make an outbound web services call to a provider which requires security credentials which CICS does not understand, then the STS can be used to convert from something which CICS does understand to something which the remote service provider understands. It could convert RACF credentials into SAML assertions for example.

One of the benefits of using WS-Trust is that it allows CICS to communicate with requesters and providers of service using security credentials which cannot be processed by CICS and RACF.

Externals

Now we look at some of the new external interfaces provided.

INVOKE WEBSERVICE

- If an application wants to invoke a web service
  - EXEC CICS INVOKE WEBSERVICE
- Allows a CICS application program to invoke a web service
- The command is threadsafe

Here is the syntax diagram for invoke webservice. Full details concerning the command can be found in the CICS infocenter.

INVOKE WEBSERVICE

```
►►──INVOKE─WEBSERVICE(name)──CHANNEL(name)──OPERATION(data-area)──────────►
►─┬──────────────────────────────────────────────────────────────────────►◄
  └─URI(data-area)─┘
```

This command is threadsafe.

SOAPFAULT

- If an application needs to issue a SOAP fault message
  - EXEC CICS SOAPFAULT CREATE
  - EXEC CICS SOAPFAULT ADD
  - EXEC CICS SOAPFAULT DELETE
- Using the API takes care of whether the SOAP message is SOAP 1.1 or SOAP 1.2 automatically.
- All of the commands are threadsafe

Here is the syntax diagram for SOAPFAULT CREATE. Full details of all the commands can be found in the CICS infocenter.

SOAPFAULT CREATE

```
►►──SOAPFAULT CREATE──┬─FAULTCODE(cvda)──────────────────────────────┬──►
                      ├─CLIENT───────────────────────────────────────┤
                      ├─SERVER───────────────────────────────────────┤
                      ├─SENDER───────────────────────────────────────┤
                      ├─RECEIVER─────────────────────────────────────┤
                      └─FAULTCODESTR(data-value)──FAULTCODELEN(data-value)─┘

►──FAULTSTRING(data-value)──FAULTSTRLEN(data-value)──┬─NATLANG('en')──────┬──►
                                                     └─NATLANG(data-value)─┘

►──┬──────────────────────────────────────────────────────────────────────┬──►
   └─ROLE(data-value)──ROLELENGTH(data-value)─┘

►──┬──────────────────────────────────────────────────────────────────────┬──►
   └─FAULTACTOR(data-value)──FAULTACTLEN(data-value)─┘

►──┬──────────────────────────────────────────────────────────────────────┬──►◄
   └─DETAIL(data-value)──DETAILLENGTH(data-value)─┘ └─FROMCCSID(data-value)─┘
```

This command is threadsafe.

IMPACT IBM

Summary

- CICS Support of Web Services
  - Allows for re-use of existing business assets
    - No change to application code
  - Allows for development of new CICS applications using web services
- CICS infrastructure support
  - CICS utilities
    - WSDL to language structure generation (batch tool)
    - Language structure to WSDL generation (batch tool)
    - Runtime support for XML to COMMAREA and vice versa mapping
  - Resource definitions on-line
    - URIMAP
    - PIPELINE
    - WEBSERVICE
  - EXEC support for outbound calls and fault messages
- Monitoring, statistics and problem determination support

101

IMPACT IBM

CICS Support of Web Services allows for the reuse of existing business assets in a Web Services environment. Existing COMMAREA application programs can function as a service provider without change.

The CICS Web Services support allows the development of new CICS applications that function as a service requester invoking an existing web service.

CICS provides utilities that will generate a language structure (copybook) from existing Web Service Definition Language (WSDL) or can generate WSDL from an existing language copybook. The utility also generates information about the XML to COMMAREA transformation that allows CICS to provide the message adapter function. In this role, CICS will be able to map the XML structure into an existing COMMAREA and after the service provider application has finished, map the COMMAREA back to an XML structure for subsequent transmission to the requester. CICS has the capability of not only using a COMMAREA to pass information to the service program but can also use the new CHANNEL/CONTAINER constructs, eliminating the 32k restriction that a COMMAREA imposes.

CICS provides RDO support for the definition of the URI mapping to a web service, definition and configuration of the pipeline process and definition of the actual web service.

CICS provides the standard qualities of services for web support including monitoring, statistics and problem determination support.

102

IMPACT IBM

More Information

- Web Services Guide - A new book in the CICS Infocenter for CICS TS V3
- "Using web services for business integration" (red book)
  - http://www.redbooks.ibm.com/abstracts/sg246583.html
- "Implementing CICS Web services" (red book)
  - http://www.redbooks.ibm.com/redpieces/abstracts/sg247206.html
- CICS
  - CICS TS V3.2 infocenter available from 2007 June 14 at
    - http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp
  - http://www.ibm.com/cics
  - Session 4133 "Web Services, Security and Transactions"
- Service-Orientated Architecture
  - http://www-306.ibm.com/software/solutions/soa/
- Web Services
  - http://www-306.ibm.com/software/solutions/webservices
  - CICS Update, May 2005, pp 14 - 22

IMPACT IBM

The foil shows where you can obtain more information should you desire to do so.

IBM 2007 IMPACT

IBM

Questions
and
Answers

impact·venture*

IBM 2007 IMPACT

IBM

The end.

106

IMPACT IBM

Additional Material

IMPACT IBM

Intentionally nearly blank.

A Pipeline Configuration File

- Pipeline XML configuration for a service provider

```
<?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/pipeline
provider.xsd">
 <service>
  <terminal_handler>
   <cics_soap_1.1_handler/>
  </terminal_handler>
 </service>
 <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```


The simplest provider pipeline configuration:

A single CICS supplied SOAP 1.1 handler as the terminal handler to parse the soap envelope.

AppHandler is specified as DFHPITP. This is the CICS module to be invoked by the pipeline.

The CICS Infocenter gives several examples of configuration files which are more complex than the one shown in the foil.

Specifications

- The runtime support is for
  - CICS TS V3.1
    - WSDL 1.1
    - SOAP 1.1 and SOAP 1.2
    - WS-I Basic Profile 1.1
    - XML 1.0
    - WS-I Simple SOAP Binding Profile 1.0
  - CICS TS V3.2 there is **also** support for
    - WSDL 2.0
    - SOAP Message Transmission Optimization Mechanism
    - XML-binary Optimized Packaging
    - WS-Trust

WSDL 1.1 specification	http://www.w3.org/TR/wsdl
SOAP 1.1 specification	http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
SOAP 1.2 specification	http://www.w3.org/TR/soap12-part0/
WS-I Basic Profile 1.1	http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html
XML 1.0 specification	http://www.w3.org/TR/2004/REC-xml-20040204/
WS-I SSBP 1.0 specification	http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0-2004-08-24.html
WSDL 2.0	http://www.w3.org/TR/wsdl20-primer/
MTOM	http://www.w3.org/TR/soap12-mtom/
XOP	http://www.w3.org/TR/xop10/
WS-Trust	http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-spec-cd-01.html

The specifications supported can be found at the URLs listed above.

Web Services Statistics

- Pipeline statistics
  - Name
  - Configuration file
  - Shelf directory
  - WSBIND directory
  - Use count
- URIMAP statistics
  - Name
  - Status
  - Webservice name
  - PIPELINE name
  - etc.
- Web Service statistics
  - Name
  - Program interface
  - Message validation
  - PIPELINE name
  - URIMAP name
  - WSBIND file
  - WSDL file
  - Porttype
  - Endpoint
  - Program name
  - Use count

Statistics data for URIMAPs

```
URIMAPs
_______
  URIMAP Name. . . . . . . . . . :  INQUIRE1
  URIMAP Enable Status . . . . . :  Enabled
  URIMAP Usage . . . . . . . . . :  Pipeline
  URIMAP Scheme. . . . . . . . . :  HTTP
  URIMAP Host. . . . . . . . . . :  *
  URIMAP Path. . . . . . . . . . :  /exampleApp/inquireSingle
  TCPIPSERVICE name. . . . . . . :  SAMPLE
  WEBSERVICE name. . . . . . . . :  INQUIRE1
  PIPELINE name. . . . . . . . . :  SAMPLE
  Templatename . . . . . . . . . :
  HFS File . . . . . . . . . . . :
  Analyzer . . . . . . . . . . . :  No
  Converter. . . . . . . . . . . :
  Transaction ID . . . . . . . . :  CPIH
  Program name . . . . . . . . . :
  Redirection type . . . . . . . :  None
  Location for redirection . . . :
  URIMAP reference count . . . . :          4
  Disabled . . . . . . . . . . . :          0
  Redirected . . . . . . . . . . :          0
```

Statistics data for PIPELINES

```
PIPELINEs
_________
  PIPELINE Name. . . . . . . . . :  SAMPLE
  PIPELINE Enable Status . . . . :  Enabled
  Configuration file . . . . . . :  /u/dbeard1/pipeline/testrun.cfg
  Shelf directory. . . . . . . . :  /u/dbeard1/sampbind/
  WSDIR pickup directory . . . . :
  PIPELINE use count . . . . . . :          14
```

Statistics data for WEBSERVICES

```
WEBSERVICE Name . . . . . . . . . :  INQUIRE2
  WEBSERVICE Status . . . . . . . :  Inservice
  Last modified date and time . . :  12/09/2005 / 10:07:04
  URIMAP name . . . . . . . . . . :
  PIPELINE name . . . . . . . . . :  SAMPLE
  Web service description (WSDL). :  /u/chrisb/WasV6/wsdl/inquireCatalog.wsdl
  Web service binding file. . . . :  /u/dbeard1/sampbind/inquireCatalog.wsbind
  Web service WSDL binding. . . . :  DFH0XCMNHTTPSoapBinding
  Endpoint. . . . . . . . . . . . :
  Validation. . . . . . . . . . . :  No
  Program interface . . . . . . . :  Commarea
  Program name. . . . . . . . . . :  DFH0XCMN
  Container . . . . . . . . . . . :
  WEBSERVICE use count. . . . . . :          6
```

IMPACT IBM

Important APARs

- PK12805
  - "Provides a Java based application programming interface to the CICS Webservice Assistant"
  - Provides fixes to a few minor bugs. It is also a pre-req for PK15904
- PK15904
  - "Provide NILLABLE ATTRIBUTE support for CICS Webservices"
  - Provides general support for attributes, not just NILLABLE
  - Provides significant performance improvement
    - 4% for small SOAP messages, up to 20% for large ones!

IMPACT IBM

PK12805 (ptfs UK09028 and UK09039) provides a Java interface to the Web services assistants. This might be of interest to you. The APAR also provides fixes to a few minor problems which became apparent post-GA of CICS TS V3.1.

PK15904 (ptfs UK11615 and UK11616) is an essential piece of maintenance. As well as the nillable attribute support, the APAR provides general support for attributes.

Probably the most important provision for general users of Web services in CICS is the performance improvement. For small SOAP messages (single output element) the improvement is about 4%, for large messages (1000 output elements) the gain is about 20%. If you are planning to use Web services in CICS, you should plan to apply this APAR.

Conclusion: Apply both sets of maintenance if you want to use Web services.

IMPACT IBM

Important APARs continued

- APAR PK24515
  - provides WS-Security support
  - Session C31 gives all the details
- APAR PK23547
  - provides enhanced capability and diagnostics for CICS WEBSERVICEs
  - adds support to the assistants for
    - base64 encoding
    - COMP-1 (float)
    - COMP-2 (double)
    - COBOL Level-88 toleration

IMPACT IBM

APAR PK24515 provides support for WS-Security related specifications for CICS TS V3.1. This support was announced some time ago and is now available. See session 4133 for full details of this support.

APAR PK23547 is a refresh of the CICS Web services assistants (DFHWS2LS and DFHLS2WS) from the development stream and includes both new capability and bug fixes. It adds support for: encoding and decoding binary data using the base64 encoding; converting COMP-1 (float) and COMP-2 (double) data types; configurable character array mapping options; specifying the TRANSACTION and USERID fields for auto-generated URIMAP resources; configurable CCSIDs per WEBSERVICE; toleration of COBOL Level-88 fields; enhanced support for zoned decimal fields in COBOL; detailed messages and SOAP Faults in the event of conversion errors.