IBM TRAINING

A15

Implementing Web Services for SOA in CICS TS V3.1

Leigh Compton, IBM Advanced Technical Support

IBM
SYSTEM z9 AND zSERIES EXPO
October 9 - 13, 2006

Orlando, FL

Abstract

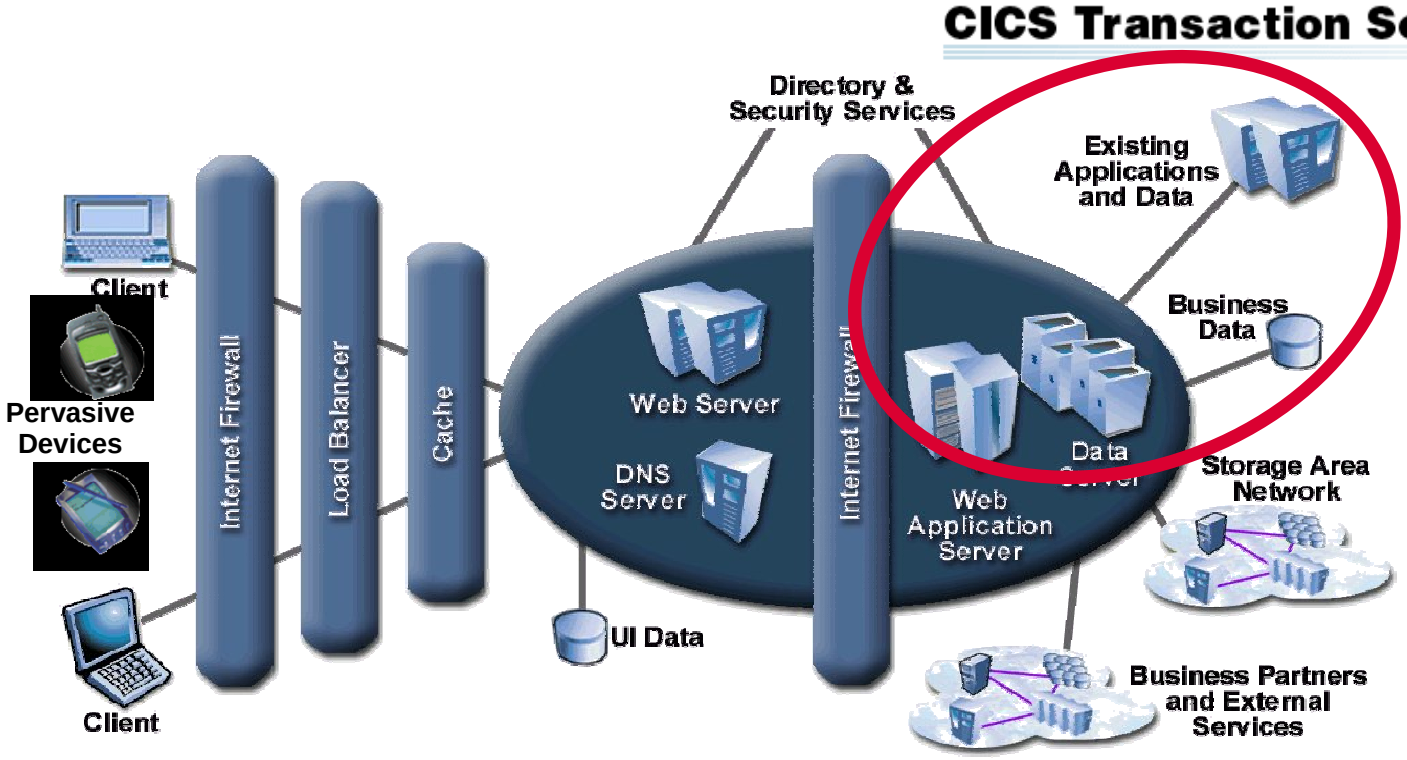
Support for Web services in the newest release of CICS Transaction Server is a fast developing capability proving to be highly popular with many customers. This session will cover the system and application considerations when using Web services in conjunction with CICS applications. This includes how to configure CICS systems to support SOAP and Web services, implementing workload management when hosting Web services, and how build XML/SOAP interfaces for CICS applications.

Acknowledgements

Agenda

- **What are SOA and Web services**
  - Why do we care?
  - A bit of History
- **Web Services in CICS**
  - CICS system configuration
    - Resource definition
  - CICS runtime support
  - Application development
    - Enablement styles
    - Available tools
- **Summary**

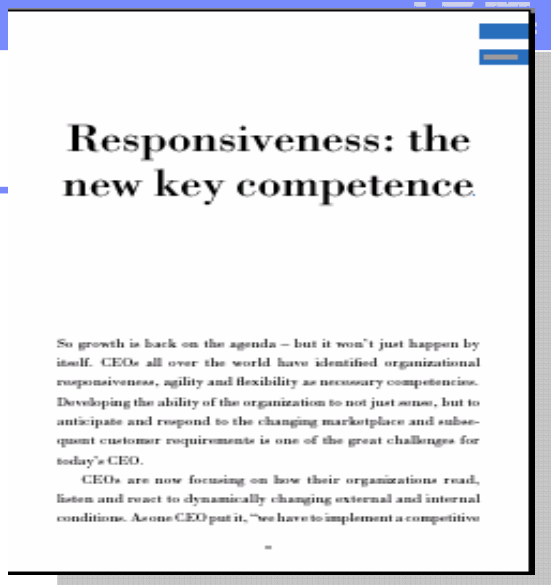
CICS and Business



- ✓Over 35 years and $1 Trillion invested in Applications ... IDC
- ✓Over $1 trillion processed/day
- ✓Over 30 billion transactions/day
- ✓Most people use CICS

Combining the reliability and security of CICS software with the flexibility of SOA

I/T Challenges


Responsiveness: the new key competence

So growth is back on the agenda – but it won't just happen by itself. CEOs all over the world have identified organizational responsiveness, agility and flexibility as necessary competencies. Developing the ability of the organization to not just sense, but to anticipate and respond to the changing marketplace and subsequent customer requirements is one of the great challenges for today's CEO.

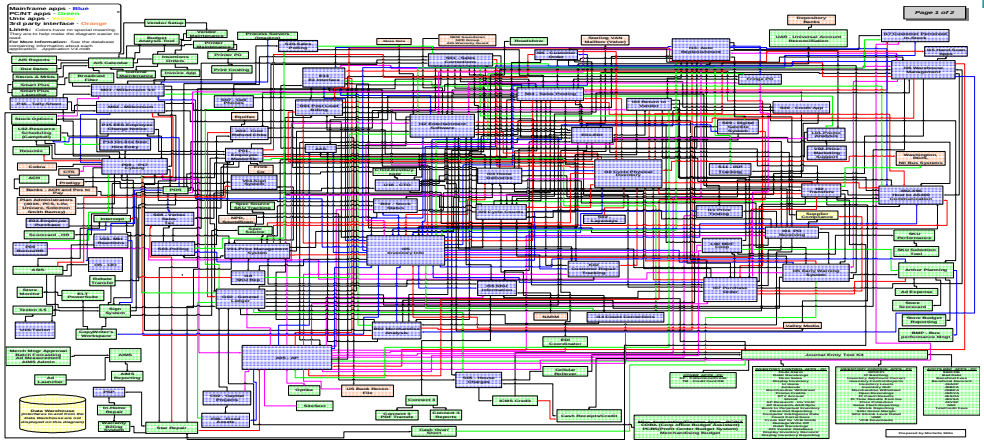
CEOs are now focusing on how their organizations read, listen and react to dynamically changing external and internal conditions. As one CEO put it, "we have to implement a competitive

Source: IBM Global CEO Survey, Feb 2004

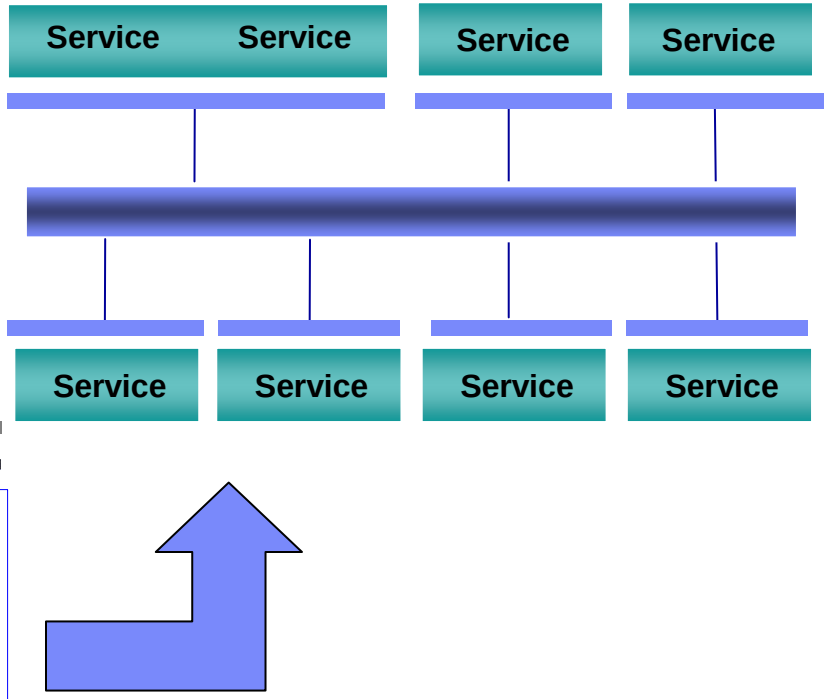
- IT Imperatives
  - Support growth
    - Improve flexibility and responsiveness
  - Keep costs in check
    - Do more with less
- Driving Need to Transform and Integrate Existing Applications
  - Extend existing applications to new audiences and opportunities
  - Exploit existing resources and skills
  - Improve performance of existing workloads for faster response times and reduced costs
  - Improve system management to enable management of more with less
  - Simplify the development process to reduce application development costs and time to deployment

Integration - Roadblocks

- Complexity
- Multiple APIs
- Hidden Interfaces
- Custom coded connections
- Not designed for change
- Lack Standards
- Not easily reusable



Actual application architecture for consumer electronics company



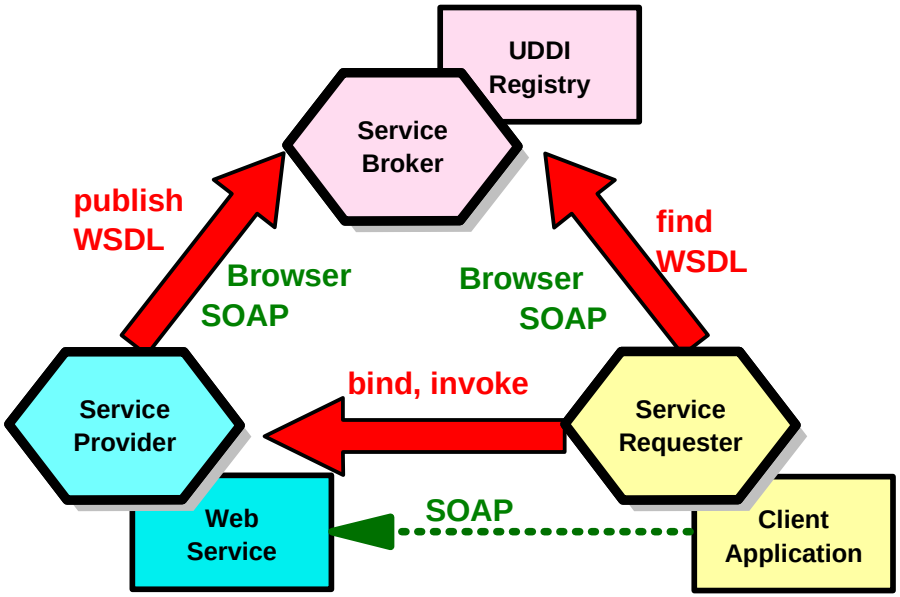
RESULT → Greater Business Responsiveness

Interoperability: A Bit of History

- **1987 -** Sun Microsystems developed Open Network Computing (ONC) RPC for its Network File System (NFS)
- **1987 -** Apolo Computer developed the Network Computing System (NCS) RPC
- **1989 -** Open Software Foundation (OSF, now The Open Group) issued a RFT for Distributed Computing Environment (DCE)
- **1989 -** The Object Management Group (OMG) formed and began the Common Object Request Broker Architecture (CORBA)
- **1990 -** Microsoft based its RPC initiatives on modified version of the DCE/RPC
- **1991 -** DCE 1.0 released by OSF, CORBA 1.0 shipped
- **1996 -** Microsoft shipped the Distributed Component Object Model (DCOM)
- **1996 -** CORBA 2.0 shipped with the Internet Inter-ORB Protocol (IIOP)
- **1997 -** Sun shipped JDK 1.1 with RMI, Microsoft announced COM+
- **1999 -** Sun shipped J2EE with RMI/IIOP for CORBA interoperate
- **1999 -** Simple Object Access Protocol (SOAP) appeared

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):**
  - An organization to ensure interoperability



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

Web Services - Notes

“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 an interface that describes a collection of operations that are network accessible through standardized XML messaging.

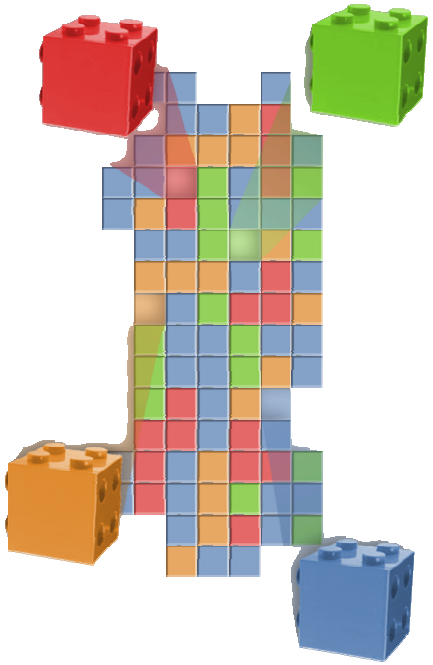
A Web service is described using a standard, formal XML notion, called its service description. It covers all the details necessary to interact with the service, including message formats (that detail the operations), transport protocols and location.

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.

Service Oriented Architecture

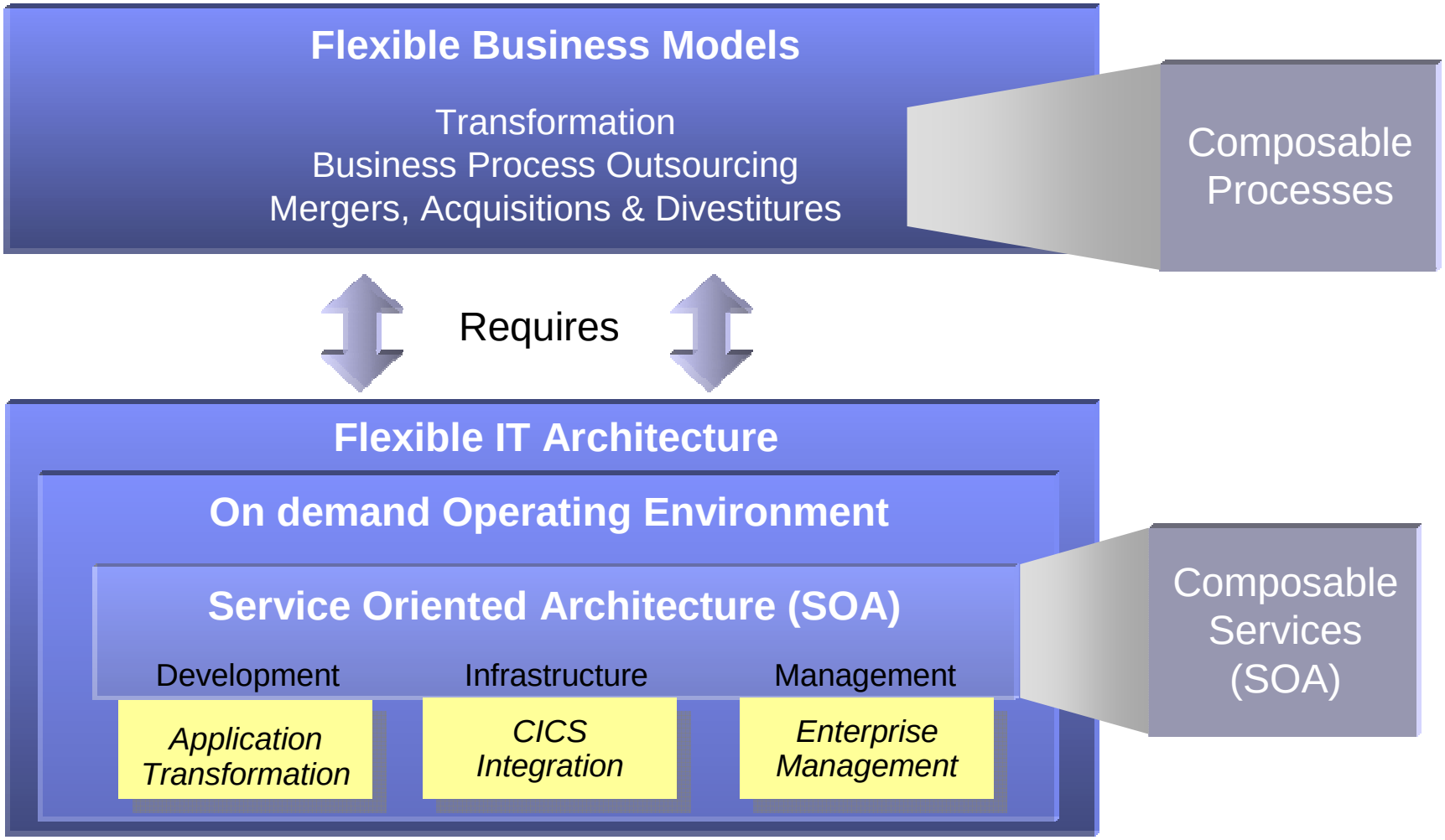
- **Align with business – quickly compose new business processes**
  - The decision of which Services are invoked
  - The decision of what order Services are invoked
  - The transformation of data output from one Service and input to another



Using services as building blocks to build an application or process

Service Oriented Architecture

Core of a Flexible IT Environment Supporting Today's On Demand Businesses



Reasons to use Web Services in CICS

- **Transform Existing Applications**
- **Extend existing applications to new audiences and opportunities**
- **Exploit existing resources and skills**
- **Improve performance of existing workloads for faster response times and reduced costs**
- **Improve system management to enable management of more with less**
- **Simplify the development process to reduce application development costs and time to deployment**

Web Services - Notes

Customers are looking to redefine their applications quickly and effectively to meet their business demands. There is a need for rapid business process adaptation and reshaping. Application maintenance
consuming 60-80% of IT budgets and staff turnover or retirement lessens individual programmer familiarity with existing systems, application maintenance efficiency is key driver.

There is also a need to meet increasing development workloads. The growth in complexity of development platforms and integration needs will force organizations to turn away from code-centric development practices in exchange for more
efficient development paradigms. They need better tooling to deliver more effective and efficient development processes.

Industry adoption and proliferation of Web Services capabilities into development platforms and tools are making it easier for companies to adopt a service-based development approach. The need for richer than HTML experiences and
disconnected operations will lead most companies to adopt multiple user interfaces delivery architectures.

Finally, Because of recent pressures for cost reductions and market demand for better processes, we expect continued pressure from business executives to switch to new, business-differentiating activities. There will be a continued strong drive from business for process improvements.

Web Services Introduction…

- **CICS Transaction Server V3.1 supports the following Web Services standards**
  - HTTP 1.0 and 1.1: Transport layer *
  - SOAP 1.1 and 1.2: Describes the Web Services message formats
  - WSDL: Describes the interface to a Web Service
  - WS-I Basic Profile 1.1: Interoperability between providers and requesters using SOAP
  - WS-Coordination: Extensible coordination framework
  - WS-AtomicTransaction: For transactionality
  - WS-Security: Authentication and encryption of all or part of a message
    - Will be delivered through the service channel at a later date

** WebSphere MQ can also be used as the transport layer*

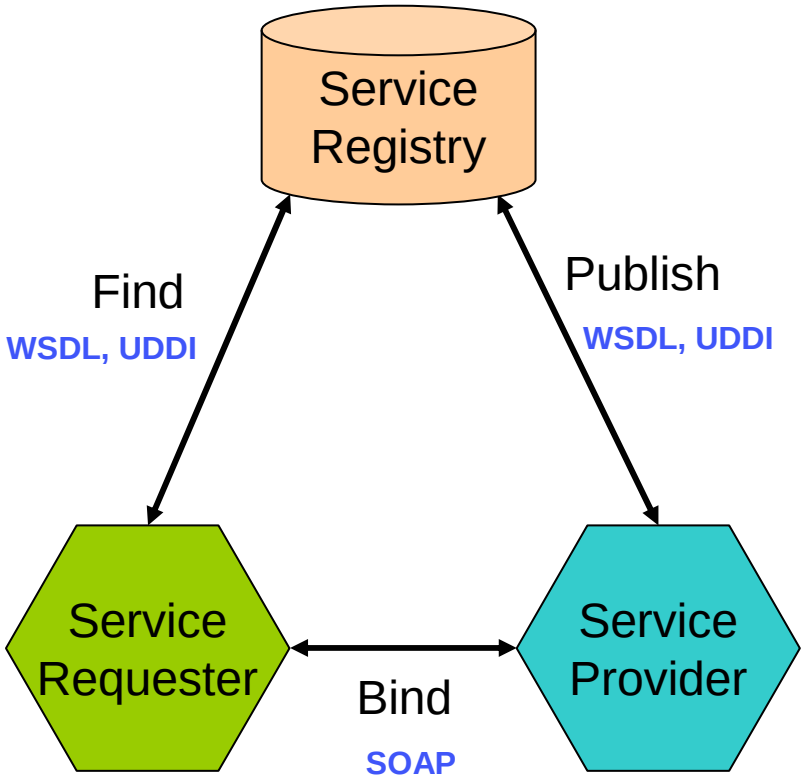
Web Services - Notes

CICS TS V3.1 provides capabilities to enable CICS based applications to be integrated with a Service Oriented Architecture (SOA), enabling them to be exposed as Web Services. CICS has the ability to act as a Web Services service provider and service requestor which means it can be seen as a full participant in this B2B world. The infrastructure provided as part of CICS TS V3.1 includes a distributed transaction coordination capability compatible with the WS-AtomicTransaction specification. It will also include a WS-Security compatible implementation for securing SOAP messages. This will be delivered, via the service channel, at a later date

WSDL 1.1 specification is here http://www.w3.org/TR/wsdl
SOAP 1.1 specification is here http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
SOAP 1.2 specification is here http://www.w3.org/TR/soap12-part0/
XML 1.0 specification is here http://www.w3.org/TR/2004/REC-xml-20040204/

Web Services Introduction…

- **Architecture**
  - Service provider
    - "Owns" the service
    - Creates the WSDL
    - Publishes the WSDL
    - Processes requests
  - Service requester
    - "Finds" the service
    - Binds to the service
      - *Invokes the service using the WSDL*
  - Service Registry
    - Hosts the service description
    - Optional for statically bound requesters



Web Services - Notes

The Web services architecture is based upon interactions between three components: a service provider, a service requester, and an optional service registry.

The Service Provider: This is the platform that hosts access to the service.

The Service Requester: This is the application that is looking for and invoking or initiating an interaction with a service..

The Service Registry: The registry is a place where service providers publish their service descriptions, and where service requesters find them. The registry is an optional component of the Web services architecture.For statically bound service requesters a service provider can send the description directly to service requesters. Likewise, service requesters can obtain a service description from other sources besides a service registry, such as a local file or an FTP site.

The interactions between the components involve the following operations:

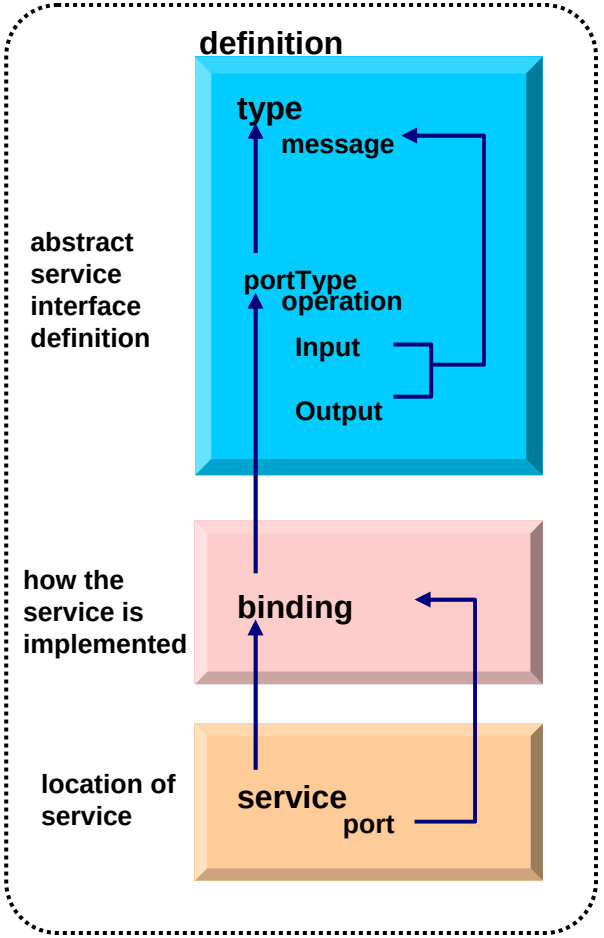
Publish: In order to be accessible, a service needs to “publish” its description such that the requester can subsequently find it. Where it is published can vary depending upon the requirements of the application.

Find: The service requester uses a find operation to retrieve the service description from the registry. The find operation may be involved in two different lifecycle phases for the service requester: at design time in order to retrieve the services interface description for program development, and at runtime in order to retrieve the services binding and location description for invocation.

Bind: The service requester uses the service description to connect to the service provider and interact with the web service implementation.

Web Services Introduction…

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



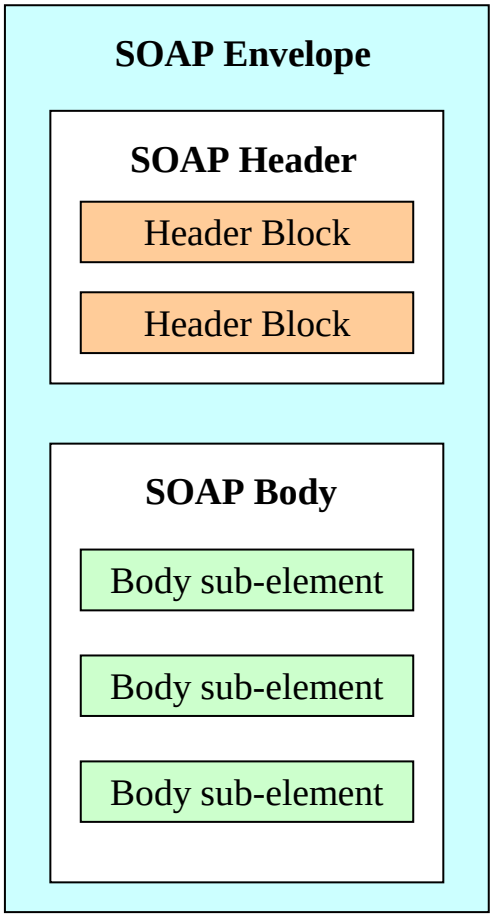
Web Services - Notes

Web service descriptions are expressed in the XML application known as Web Service Description Language (WSDL). The structure of WSDL allows a service description to be partitioned into 2 parts. An abstract service interface definition that describes the interfaces of the service, and makes it possible to write programs that implement, and invoke, the service. A concrete service implementation definition that describes the location on the network (or endpoint) of the provider's Web service, and other implementation specific details, and that makes it possible for a service requester to connect to the service provider. A WSDL document uses the following major elements.

<types> A container for data type definitions using some type system (such as XML Schema). Defines the data types used within the message. The <types> element is not required when all messages consist of simple data types.
<message> Specifies which XML data types are used to define the input and output parameters of an operation.
<portType> Defines the set of operations supported by one or more endpoints. Within a <portType> element, each operation is described by an <operation> element.
<operation> Specifies which XML messages can appear in the input and output data flows. An operation is comparable with a method signature in a programming language.
<binding> Describes the protocol, data format, security and other attributes for a particular <portType> element.
<port> Specifies the network address of an endpoint, and associates it with a <binding> element.
<service> Defines the Web service as a collection of related endpoints. A <service> element contains one or more <port> elements.

Web Services Introduction…

- **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



Web Services - Notes

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.

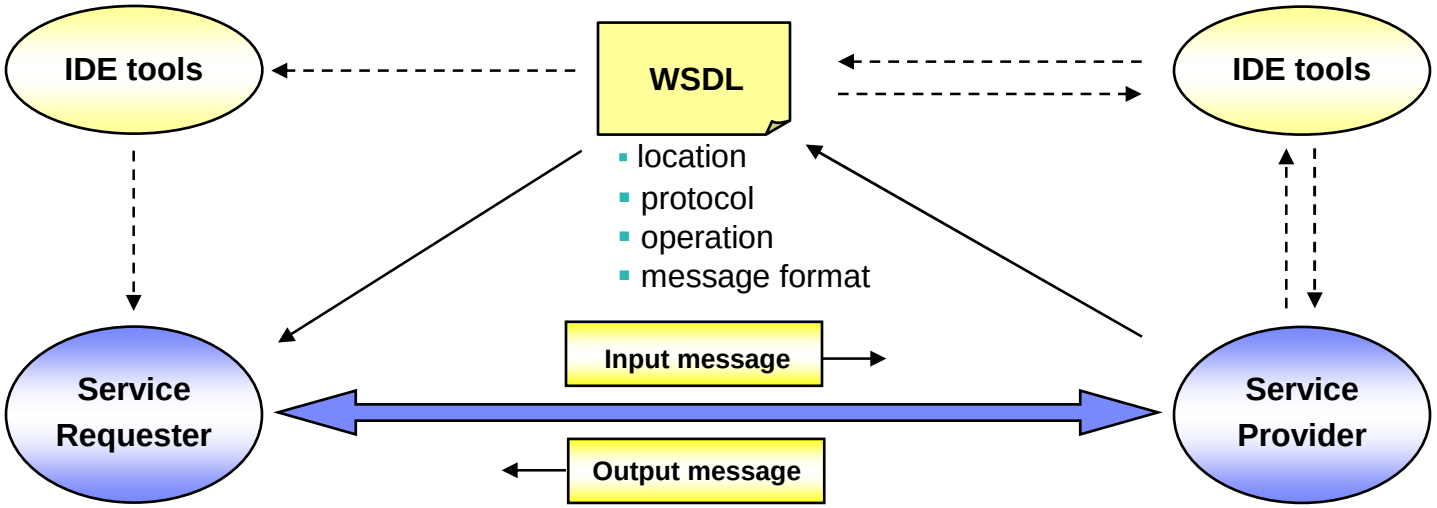
The SOAP <body>, a mandatory element, containing information intended for the ultimate recipient of the message.

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**



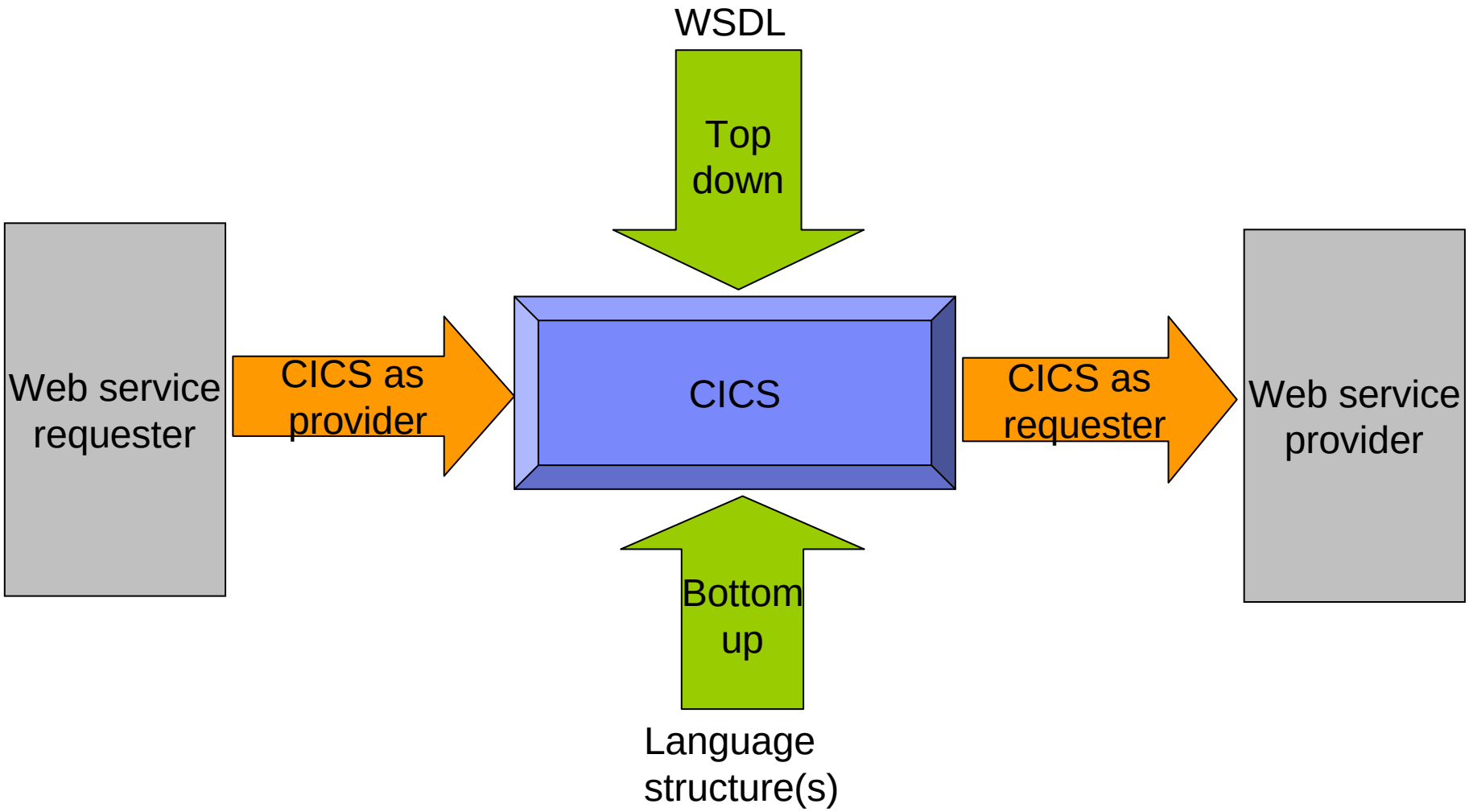
Web Services - Notes

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

Web Services Enablement Styles



Web Services - Notes

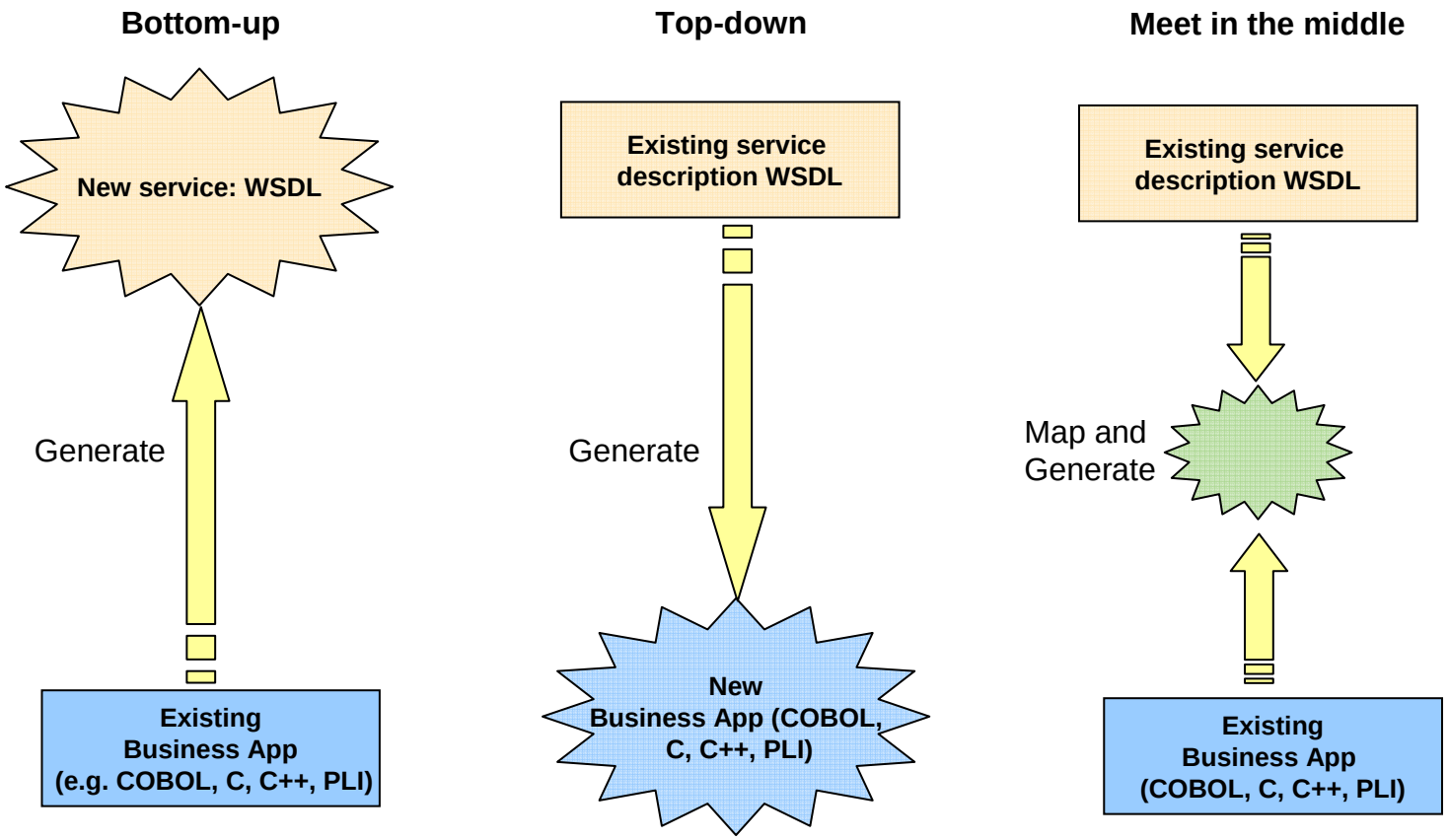
Top down – means that we start with a WSDL document and convert it into a form useable as a CICS Web Service.

Bottom up – means that we start with a language structure and convert it into a form useable as a CICS Web Service.

Requester – means that CICS is the user of a Web Service provided elsewhere.

Provider – means that CICS is the host of the Web Service which is being invoked from elsewhere.

Web Services Enablement Styles…



Web Services - Notes

There are three types of approaches in a Web service application development.

With the top down approach, the development will start with a WSDL. Users will define the interface and created the WSDL, then create a Web service application based on that interface. The interface will be well defined but will need to create the whole service provider application.

With the bottom up approach, an existing application will be used for the service provider application. The WSDL will be created based on the interface which the application uses. It is the easiest way to implement a Web service, but the interface to the service requester may not be very suitable.

The meet in the middle approach is where there is an existing application, but using a better interface for the service requester. In this case, a wrapper program will be used to take the convert the existing application interface to and from the interface to the requester. It will have both advantages of the other approaches, with the suitable interface and low development cost.

Bottom up approach in CICS TS V3.1

- **An existing CICS application is to be exposed as a web service**
  - Language structure 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 to WSDL
    - The language structure can be COBOL, PL/I, C or C++
  - Use WebSphere Developer for z/OS to convert language structure to WSDL
    - The language structure can be COBOL
  - Publish the generated WSDL, on UDDI for example
  - A file called the WSBind file is also produced

Web Services - Notes

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. WebSphere Developer for z/OS can also be used to convert a COBOL 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.

Top down approach in CICS TS V3.1

- **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 WebSphere Developer for z/OS to generate a COBOL language structure from the WSDL
    - Use the generated language structure in a CICS application program.
- **A file called the WSBind file is also produced.**

Web Services - Notes

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++).

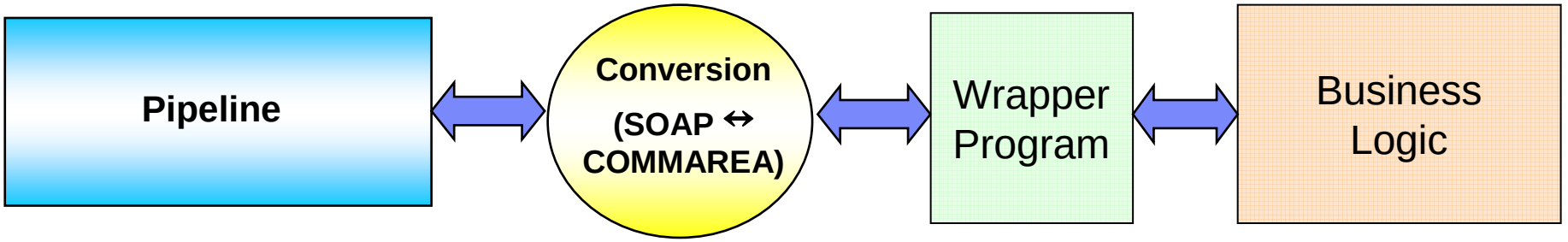
Meet in the middle approach in CICS TS V3.1

- **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
    - WebSphere Developer for z/OS could be used to hide the fields
  - When the application is written in assembler
    - A wrapper program could map the assembler DSECT into a COBOL copybook
  - 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

Web Services - Notes

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



Web Services - Notes

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.

CICS 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

Web Services - Notes

There are a number of interrelated resource definitions required to process a Web Service in CICS 31.

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 3.1 will use a URIMAP definition to map the incoming Universal Resource Identified (URI) to a specific WEBSERVICE definition. The associate 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, if a COMMAREA or CHANNEL is used and how CICS is to transform the message between the SOAP XML format and COMMAREA format.

CICS Resource Definitions…

- **TCPIPSERVICE definition**
  - Required when CICS is a service provider
  - URIMAP matching will occur when protocol (HTTP) is specified

Web Services - Notes

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

CICS Resource Definitions…

- **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*')

Web Services - Notes

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

For CICS as a service provider:
Define
A QLOCAL object that defines the local queue used to store the messages until they are processed
A PROCESS object that specifies the CICS transaction CPIL that will process messages from the local queue

Define the input queue

```
DEFINE
QLOCAL ('queuename')
DESCR ('description')
PROCESS(processname)
INITQ('initqueue')
TRIGGER
TRIGTYPE(FIRST)
TRIGDATA('default target service')
BOTHRESH(nnn)
BOQNAME('requeuename')
```

queuename is the local queue name
processname is the name of the process instance that identifies the application started by the queue manager when a trigger event occurs. The name should match that in the definition of the process object
initqueue is the name of the initiation queue to be used (e.g. as specified in IQ= in CSQCPARM in INITPARM in SIT)
default target service is the default target service to be used if not specified on the request, flows in RFH2 header. The target service is of the form '/string' and is used to match the path of a URIMAP definition. For example, '/ SOAP/test/test1'. Note that the first character must be '/', the next characters do not need to be SOAP. This is a difference from the SOAP for CICS feature, where TRIGDATA('SOAP/target_program') was specified.

CICS Resource Definitions…

- **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)
```

Web Services - Notes

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.

CICS Resource Definitions…

- **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 message handlers that will act on a service request and on the response
  - SHELF
    - HFS directory for CICS use
  - WSDIR
    - Pickup directory for WS Bind files

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

Web Services - Notes

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.

CICS Resource Definitions…

- **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

Web Services - Notes

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.

CICS Resource Definitions…

- **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>
```

Web Services - Notes

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.

CICS Resource Definitions…

- **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
```

Web Services - Notes

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).

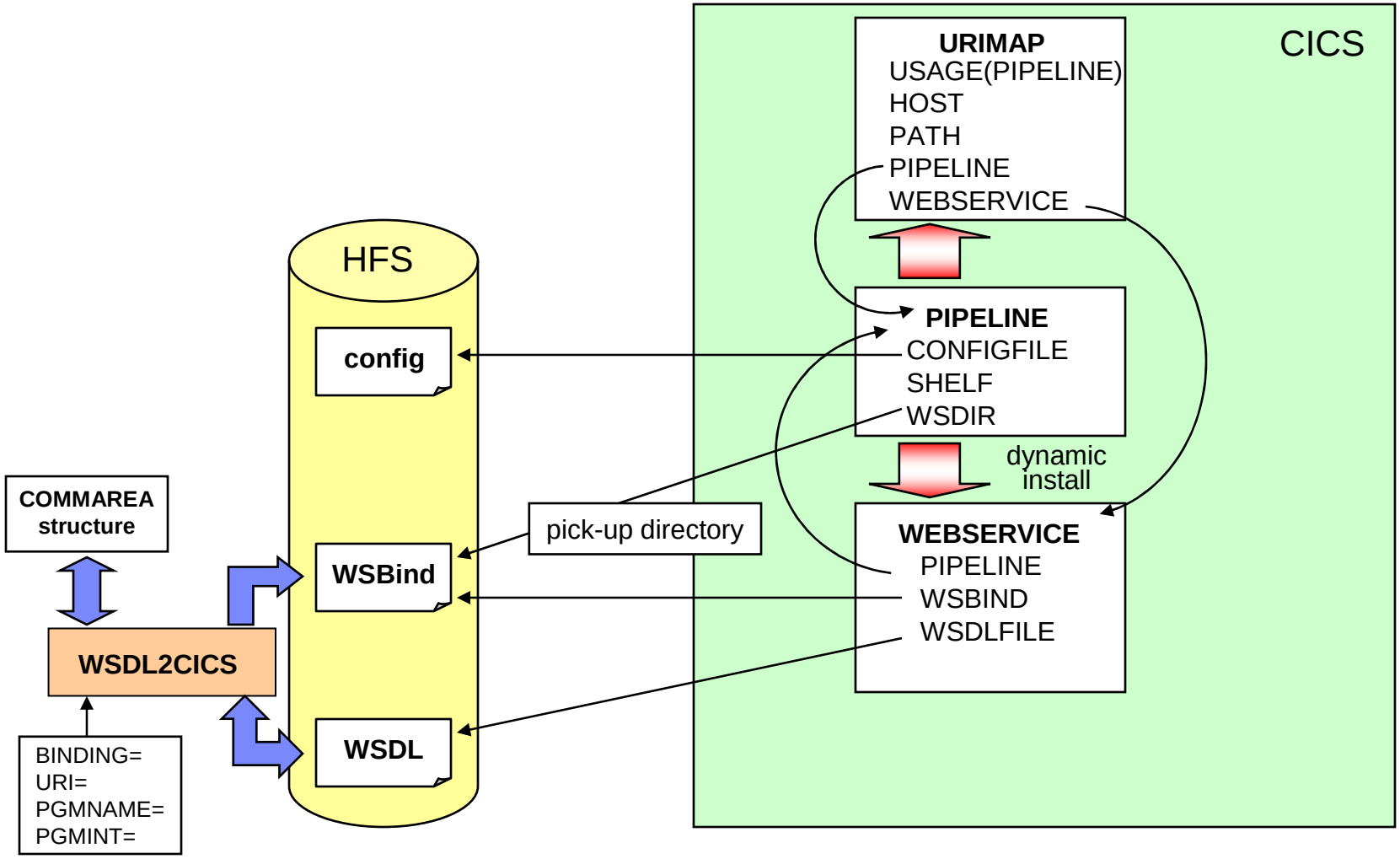
The execution environment contains three components that are specified in the WEBSERVICE attributes:

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.

CICS Web Service Resource Interrelationships



Web Services - Notes

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.

Defining the CICS Web Services Resources

- **Define a TCPIPSERVICE (or WMQ) and a PIPELINE**
- **Then install the PIPELINE definition and 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**

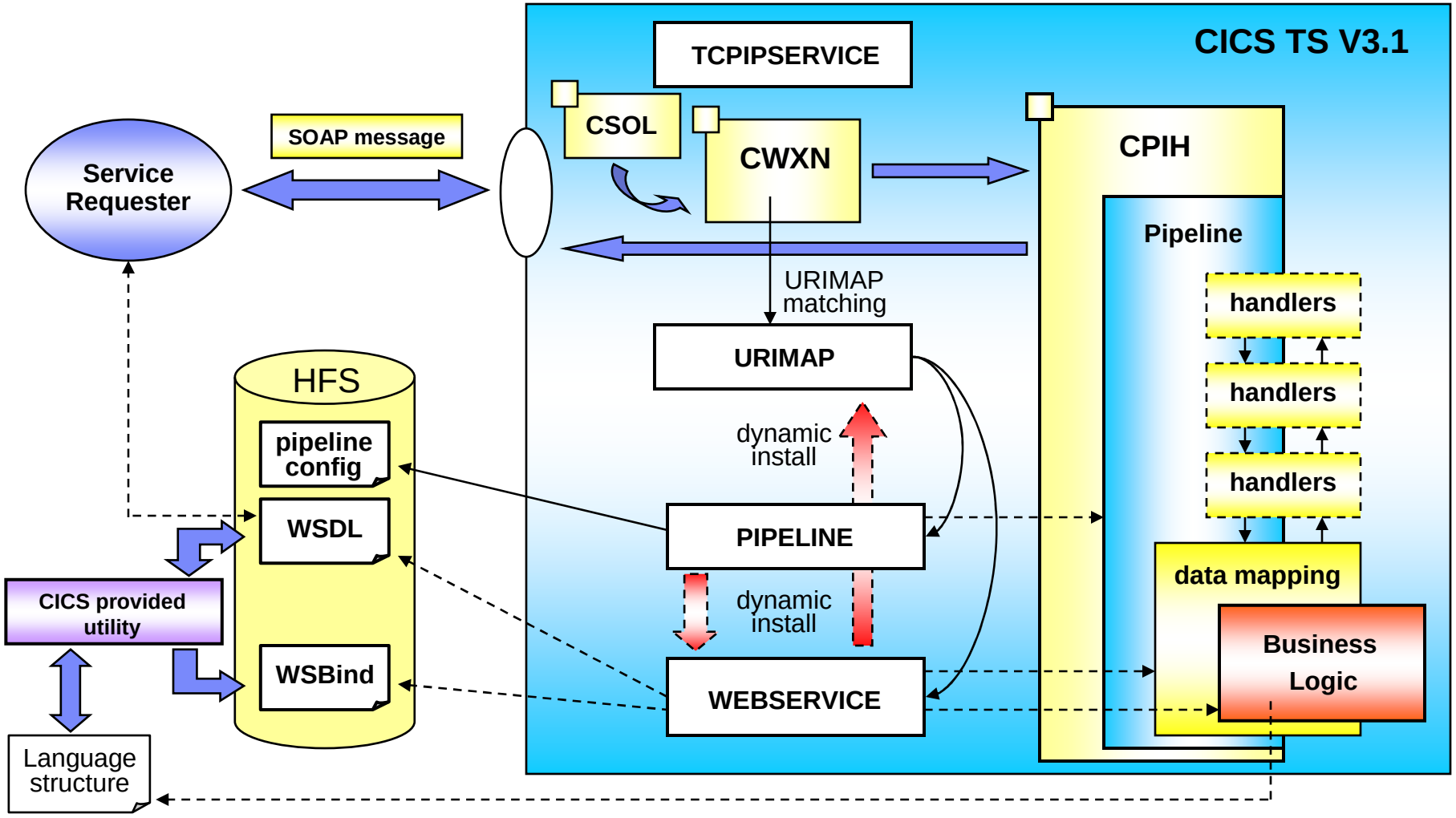
Web Services - Notes

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.

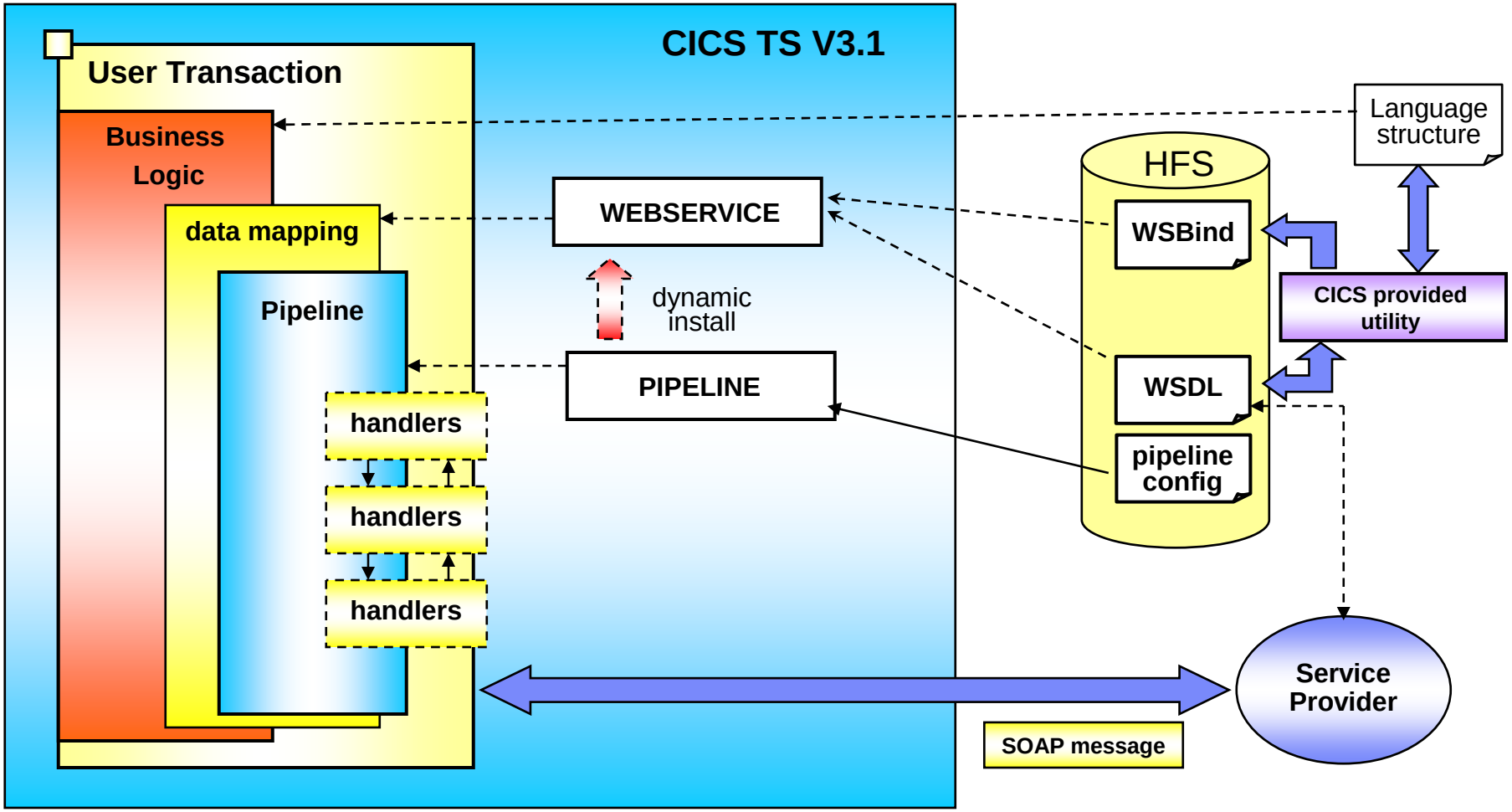
CICS as a service provider



Web Services - Notes

This diagram illustrates the flow and the CICS resources necessary to allow CICS to function as a service provider.

CICS as a service requester

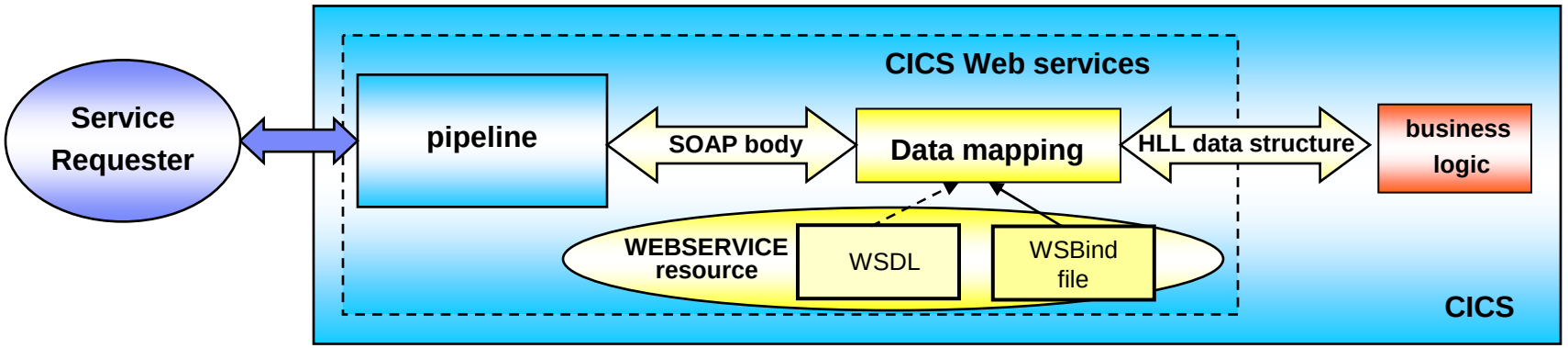


Web Services - Notes

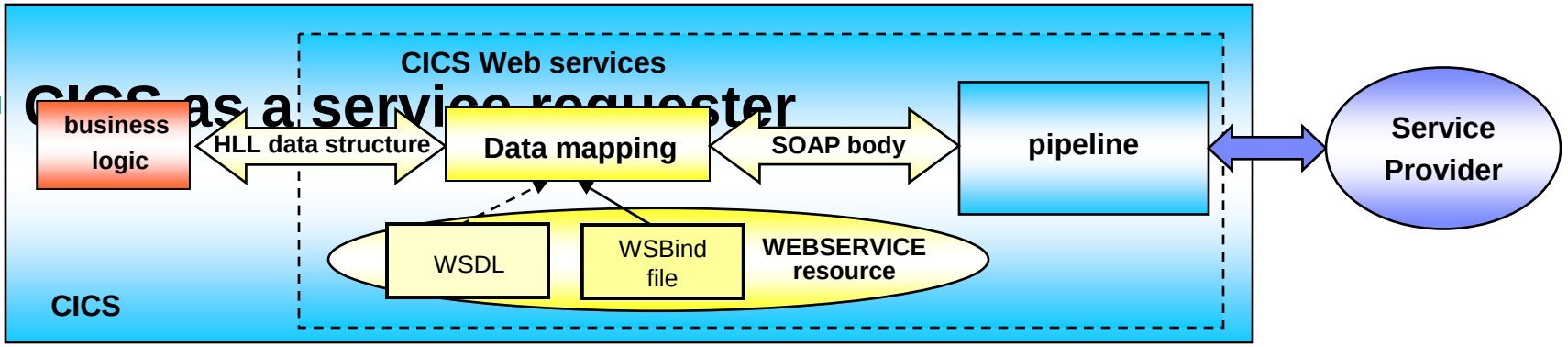
This diagram illustrates the flow and the CICS resources necessary to allow CICS to function as a service requester.

CICS usage of the WSBind file

- **CICS as a service provider**



- **CICS as a service requester**



Web Services - Notes

This diagram shows the usage of the WSBind file and WEBSERVICE resource in the runtime.

When the SOAP message arrives, the data mapping component in the pipeline will use the information from the WSBind file and convert the SOAP message into a language structure. When the message is converted it will use the information in the WEBSERVICE resource and call the target application. When the target application returns with a response language structure, it will convert it back into a SOAP message.

It is mostly the same when CICS is a service requester. The service requester will invoke the Web service processing. The data mapping component will convert the language structure into a SOAP message an run the outbound pipeline. The pipeline will use the information in the WEBSERVICE resource and send the SOAP message outbound. When the response is received, the data mapping component will convert the response SOAP message back into a language structure and pass it back to the service requester program.

The WSDL file is optional. It is only needed if a validation of the SOAP message is required.

Application Programming Interfaces

- **Invoking a Web Service from a CICS application program**
  - CICS as a service requester
    - EXEC CICS INVOKE WEBSERVICE ( ) CHANNEL ( ) URI ( ) OPERATION ( )
      - 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)
      - URI: Universal Resource Identifier of the Web Service (optional)
      - OPERATION: name of the operation to be invoked

Web Services - Notes

The purpose of the command is to invoke the web service named and pass the channel into which the relevant containers have been put.

The container DFHWS-DATA must be created by the requesting application before the INVOKE WEBSERVICE command is issued.

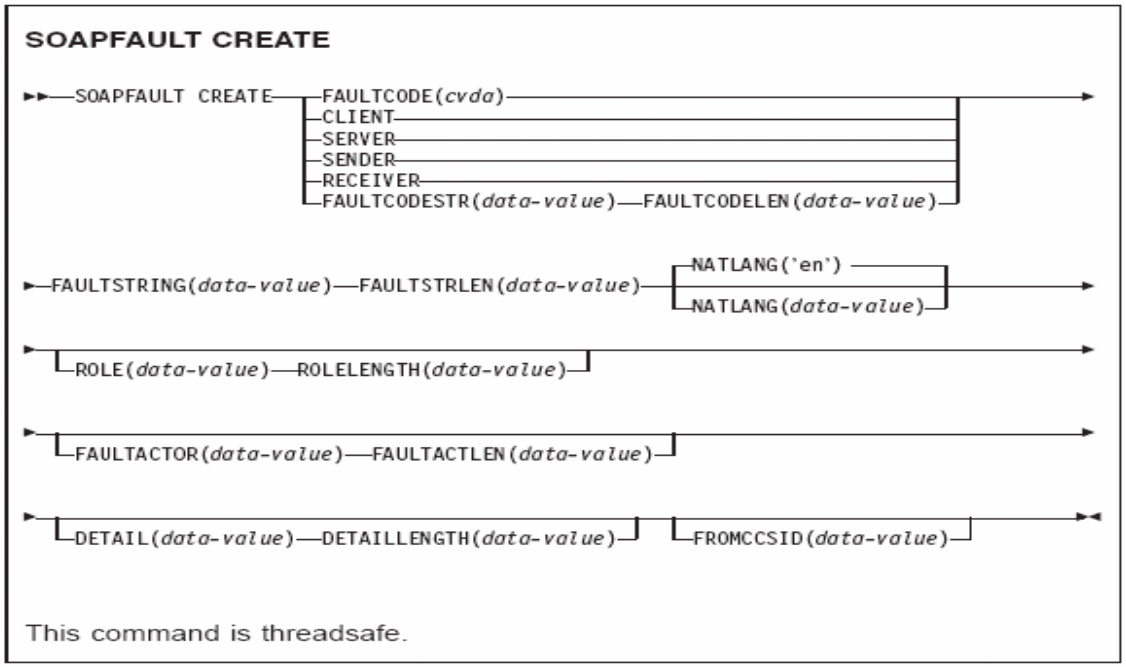
The same container, DFHWS-DATA, will hold the response, if any, from the Web Service.

Application Programming Interfaces…

- **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**

Web Services - Notes

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



Web Services Statistics

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

Web Services - Notes

The foil summarizes the new statistics information which is available for pipelines and web services.

Migration of the CICS TS Version 2 SOAP Feature

- **Coexistence supported for migration**
  - Version 2 feature may be installed on CICS TS 31
  - Provides the same level of function as on Version 2
- **Migration to CICS TS 31 Web Services requires**
  - Modifying your message adapters to use the new interfaces
  - Review your use of containers. The SOAP for CICS feature uses BTS containers; the Web services support in CICS TS V3;1 does not use BTS. The containers names used in the Web services support, are different from the names used in the SOAP feature
  - Replacing the user-written handlers with SOAP header handlers defined in the PIPELINE configuration file

Notes

The **SOAP for CICS feature**, orderable with CICS TS V2.2 and V2.3, is not orderable with CICS TS V3.1. However, to assist migration for customers who already have this feature, the feature may be used and is supported with CICS TS V3.1, and applications will continue to run. Customers are recommended to migrate to the Web services support capabilities of CICS TS V3.1.

The SOAP for CICS feature relies to a considerable extent upon user-written code, and therefore it is not possible to set out a step-by-step migration task. However, here are some of the things you will need to think about
.
Consider using the Web services assistant to construct and parse SOAP messages.

If you use SOAP messages, but decide not to use the Web services assistant, you may be able to reuse your existing code for constructing and parsing the messages. However, you should consider whether to use the CICS-provided SOAP message handlers, because they are designed to work with SOAP 1.1 and SOAP 1.2 messages.

Review your use of containers. The SOAP for CICS feature uses BTS containers, whereas CICS Transaction Server uses channel containers.

Consider how to migrate the function that was provided by your pipeline programs. The pipeline in the SOAP for CICS feature has a fixed number of user-written programs, each with a designated purpose. The function provided by some of these programs is provided in CICS Transaction Server by the CICS-provided SOAP message handlers, so you may be able to dispense with these programs altogether. The way that pipeline programs communicate with CICS, and with one another, has changed, so you will need to review these programs to see if they can be reused in the new environment.

In the SOAP for CICS feature, you could have just one pipeline for all your service provider applications, and one for all your service requesters. In CICS Transaction Server, you can configure many different pipelines.

Additional Documentation

- **CICS TS 3.1 Release Guide, SC34-6421**
- **CICS TS 3.1 Migration Guide(s)**
- **CICS TS 3.1 URLs**
  - "Home Page"
    - http://www.ibm.com/software/htp/cics/tserver/v31/
  - Library
    - http://www.ibm.com/software/htp/cics/library/cicstsforzos31.html
- **Web Services Guide**
  - A new book in the CICS Infocenter for CICS TS V3.1

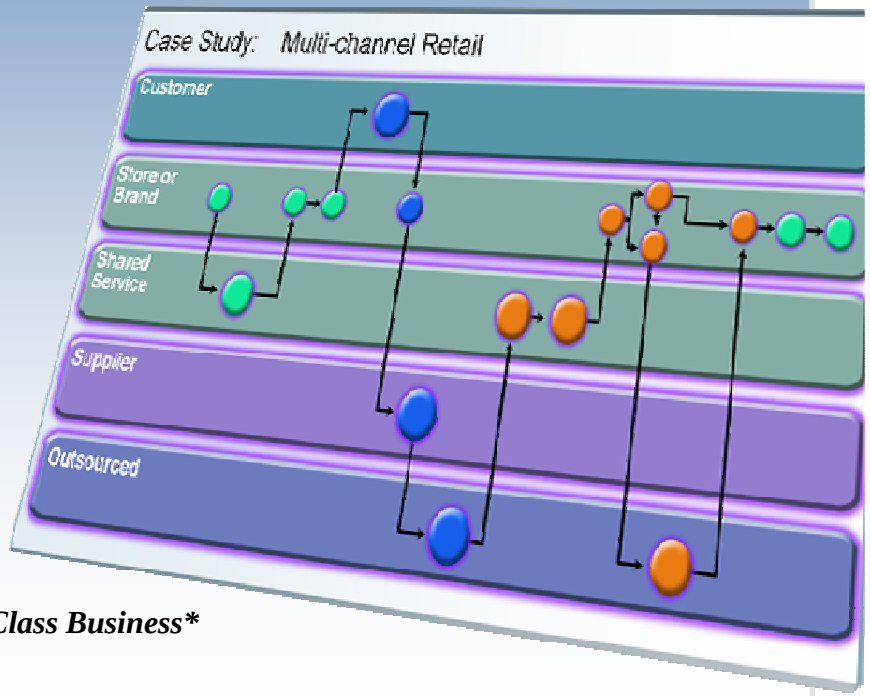
Web Services - Notes

This slide contains pointers to additional documentation.

SOA Business Integration, Flexibility, Reuse

- **Economics: globalization demands flexibility**
- **Business processes: changing quickly and sometimes outsourced**
- **Growth: at the top of the CEO agenda**
- **Reusable assets: can cut costs**
- **Information: greater availability**
- **Crucial for flexibility and becoming an On Demand Business**

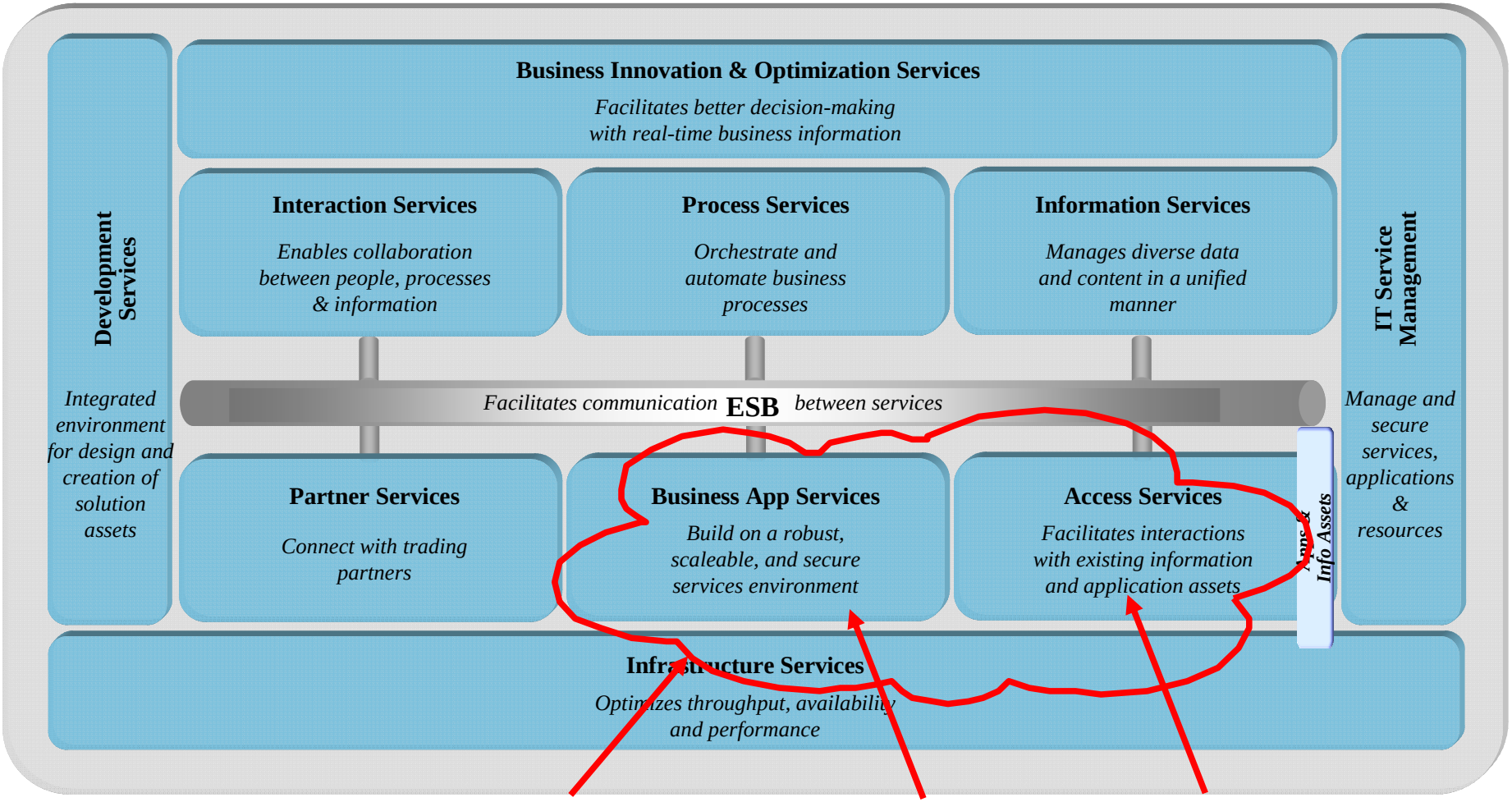
Traditional Business*



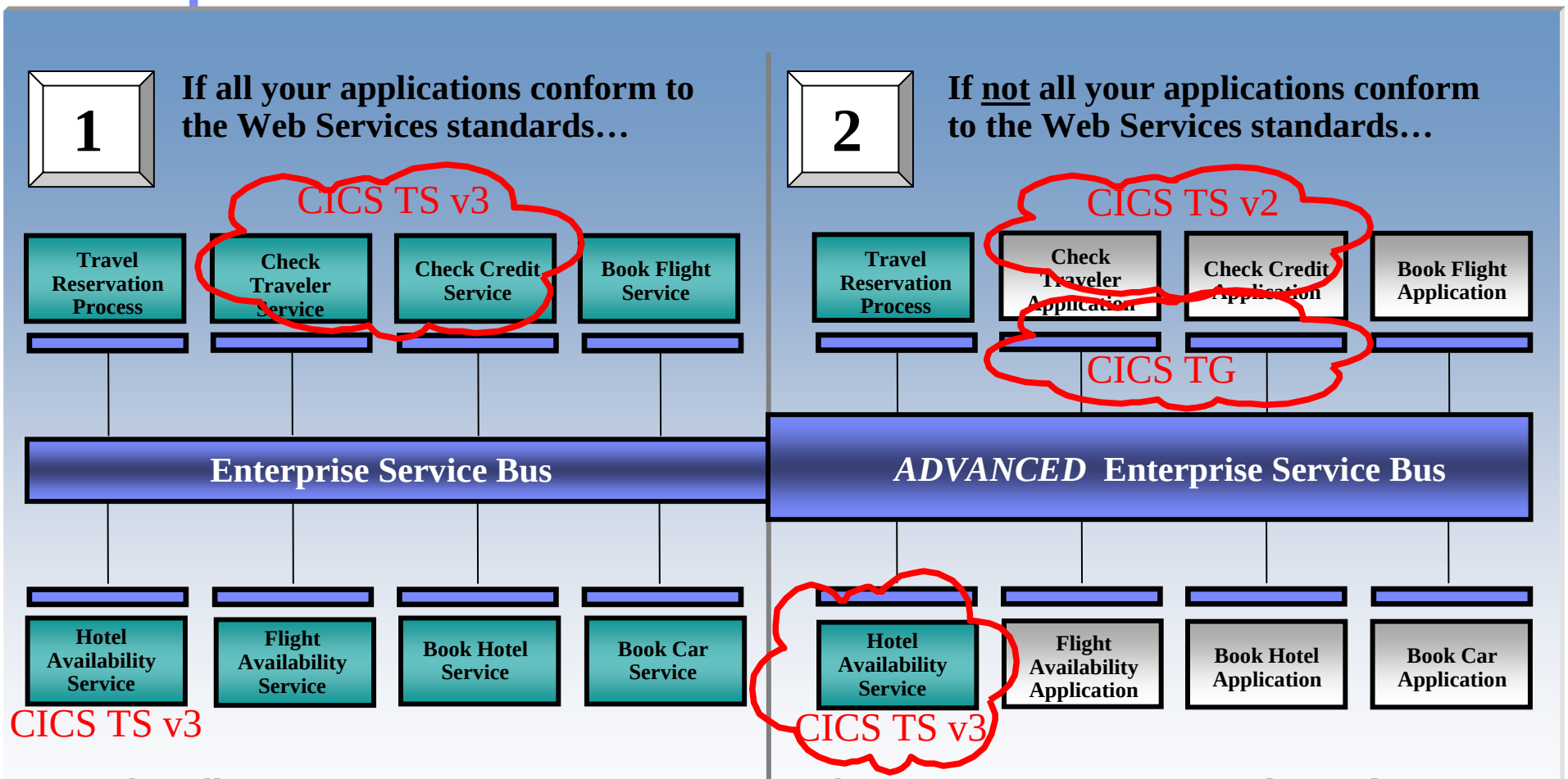
Today's World-Class Business*

*Sources: CBDi

SOA Reference Architecture



Enterprise Service Bus



…then all you may require is an **ESB** focused on standards-based service integration.

…then you may require a more **advanced ESB** focused on the integration of services with existing non-services assets.

CICS Transaction Server V3.1

- **CICS Web Services Integration and Related Features**
  - Web services and the CICS Web services assistant
  - Web services standards support
  - HTTP/1.1 including outbound API and URIMAPs
  - Transport Layer Security, 256-bit encryption, and improved SSL V3
  - Containers and channels
  - Information Center
  - Open Transaction Environment for all thread safe applications
  - XPlink for C & C++ programs

Notes

Looking at a Service Orientated Architecture (SOA) from a business perspective, SOA is a set of business, process, organizational, governance and technical methods to reduce or eliminate frustrations with IT, quantifiably measure I/T's business value and create an agile business environment for competitive advantage.

From a technical perspective, SOA is a way for different computers, from different vendors, with different programs, from different functional areas of the business (or externally to customers, suppliers or vendors) to intelligently talk and exchange data with each other.

The strategy for CICS TS V3 is to deliver a set of capabilities to allow your valuable CICS applications to evolve to participate in flexible business models. This is consistent with IBM's continued investment in the mainframe and joins with the latest versions of WebSphere, DB2, and IMS, enabling companies to deploy Web Services to benefit from a service oriented architecture based on standards.

The key to an On Demand enterprise is to employ flexible business models, implemented through composable processes which in turn call composable services. At the heart of the SOA architecture are the building blocks of Development, Infrastructure and Management.

The focus of CICS TS V3.1 is to align with the building blocks of a Service Orientated Architecture, and to deliver a set of capabilities which provide customer value by enabling business flexibility through IT simplification. The SOA building blocks align to the strategic themes of CICS Transaction Server V3; Application Transformation, CICS Integration and Enterprise Management.

CICS Integration enables re-use of CICS applications, within flexible IT infrastructure, via standard APIs and protocol

Application Transformation enables enhancement of existing applications and construction of new applications, using contemporary programming languages, constructs and tools

Enterprise Management enables effective management of large runtime configurations via modern user interfaces, so that demanding service level objectives can be met.

Summary

- **CICS TS Provides Full Featured support of SOA and Web services**
  - Complement z/OS qualities of service such as high availability, scalability, low cost per transaction, and excellent security
- **CICS TS provides the base for the majority of mainframe applications today**
  - An efficient and optimized runtime for the reuse and transformation of existing CICS applications
  - Provides easy to use services that exploit new technologies by building on CICS skills
  - First class support and management of mixed application types and workloads
- **CICS TS V3.1 available since March 25, 2005**

Increased ease of **CICS Integration**	▪ **Web services capabilities to extend CICS applications to a Services Oriented Architecture** ▪ **Support for industry-leading SSL and TLS protocols**
Enhanced **Application Transformation**	▪ **Ability to leverage single development tool for application transformation and integration** ▪ **Optimized CICS data exchange capabilities**
Improved performance & **Enterprise Management**	▪ **Improved workload throughput** ▪ **Enhanced C and C++ programs performance** ▪ **Extension of the CICSPlex SM Web User Interface**