IMS18

IMS XML, SOAP and Web Services Solutions

Christopher Holtz

IMS Development

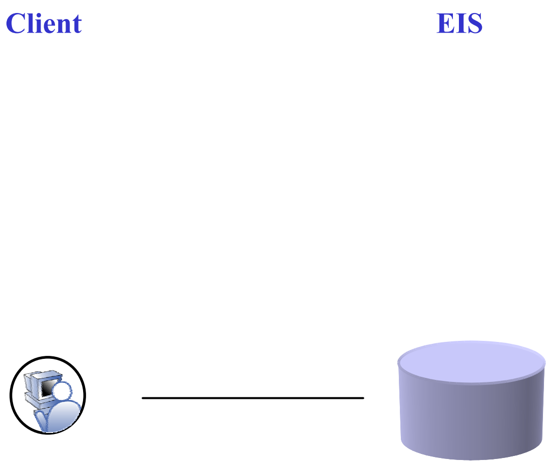
Silicon Valley Laboratory

IBM Corporation

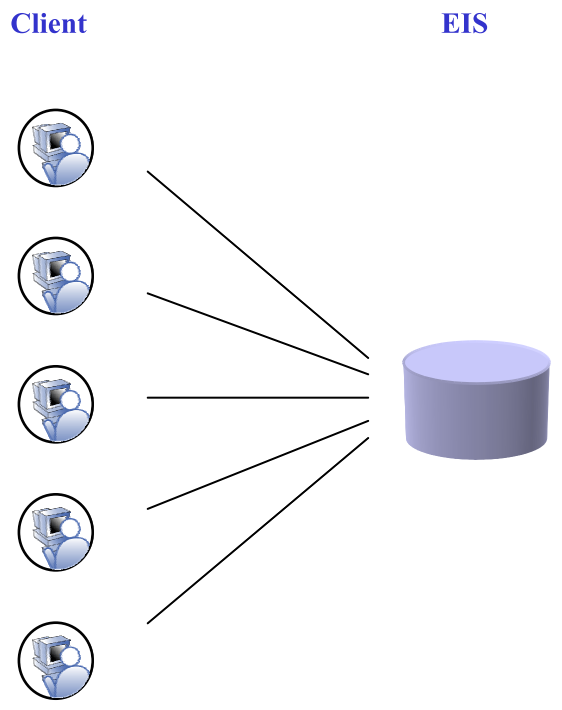
Overview

- History – Client/Server to SOA in 5 minutes
- SOA / Web Services Concepts
  - XML
  - SOAP
  - WSDL
  - UDDI
- IMS and SOA
  - CAM
  - MFS Web Services
  - XML in IMS Connect
  - IMS SOAP Gateway

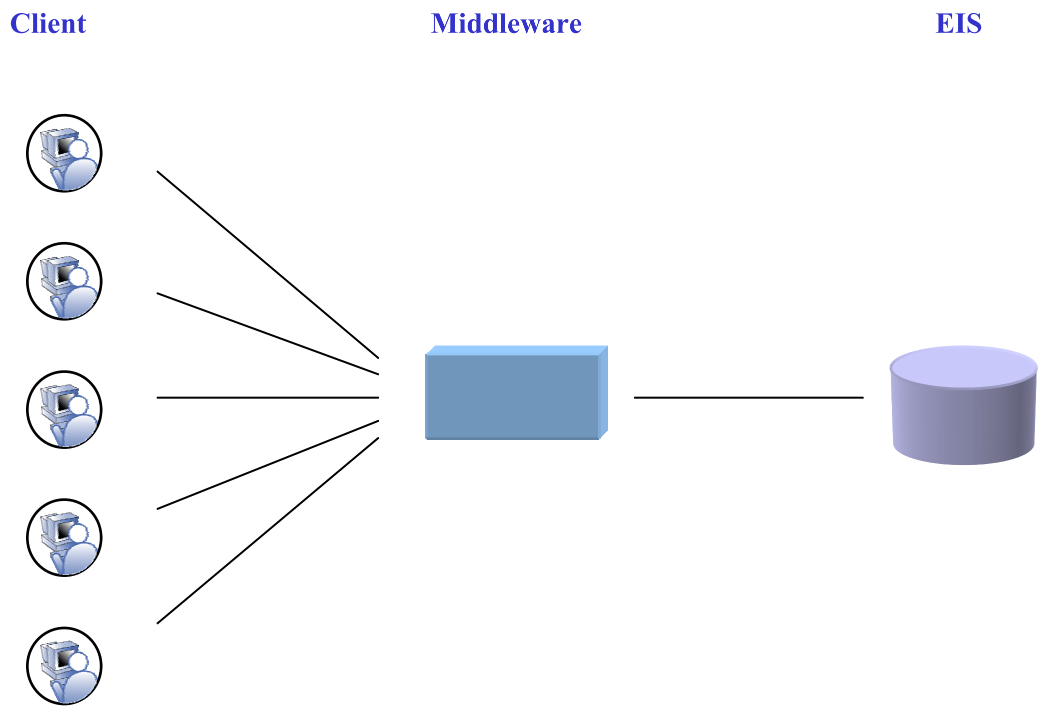
Client Server Architecture



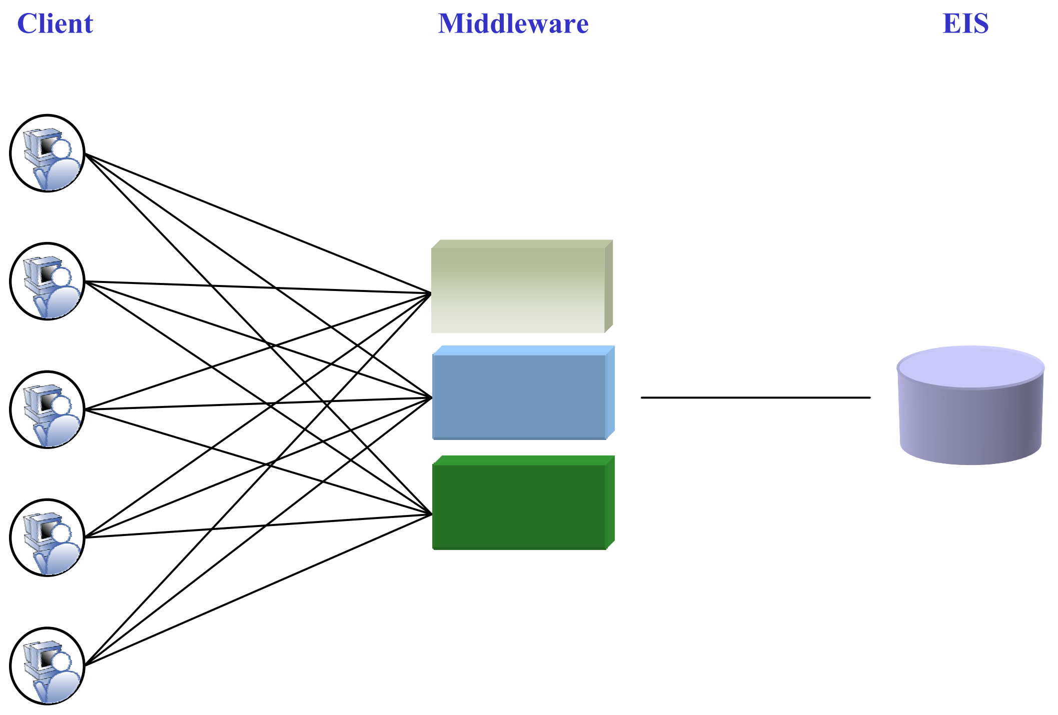
Client Server Architecture



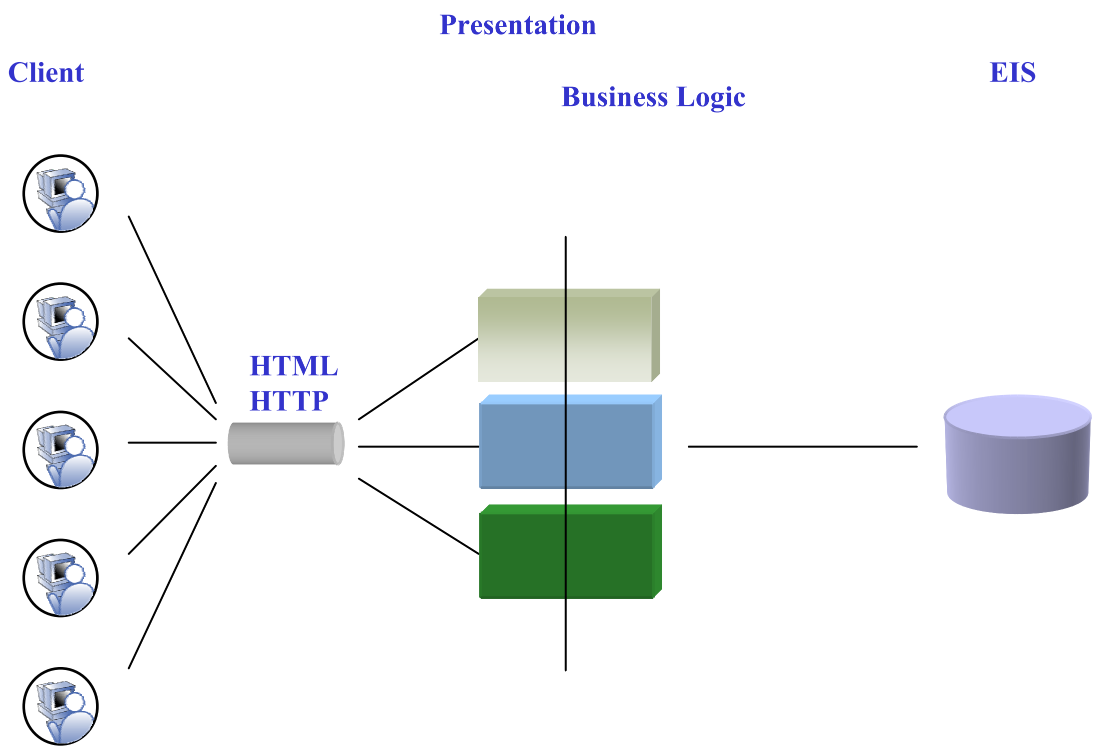
3-tier Architecture



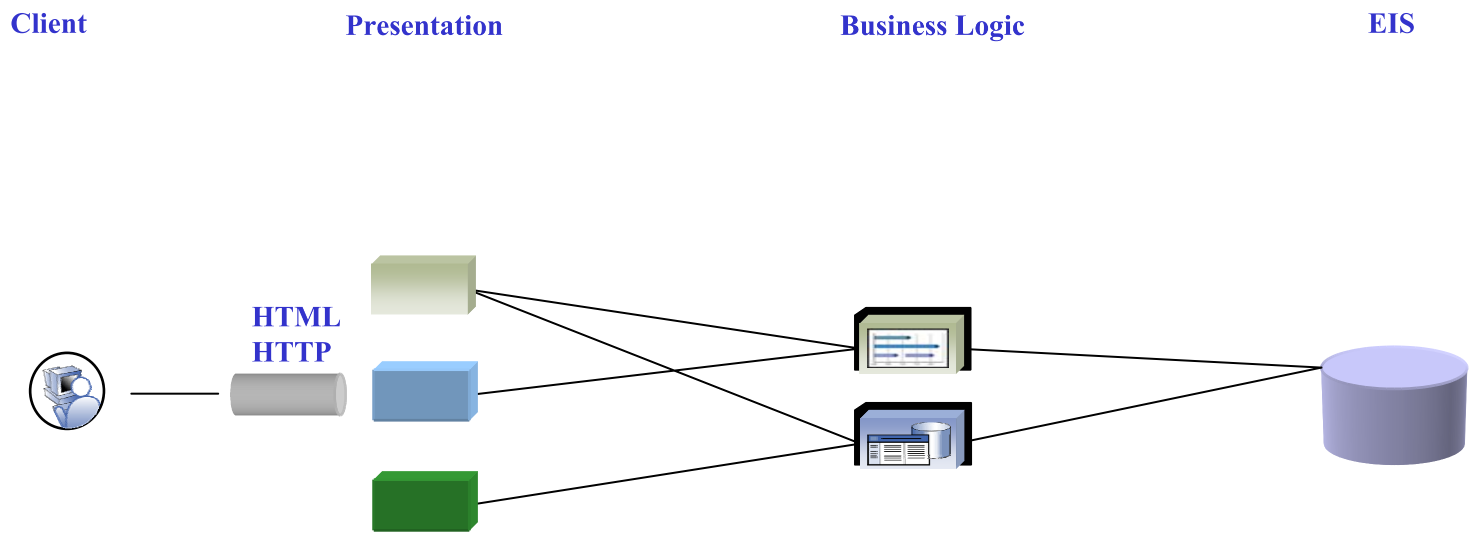
3-tier Architecture



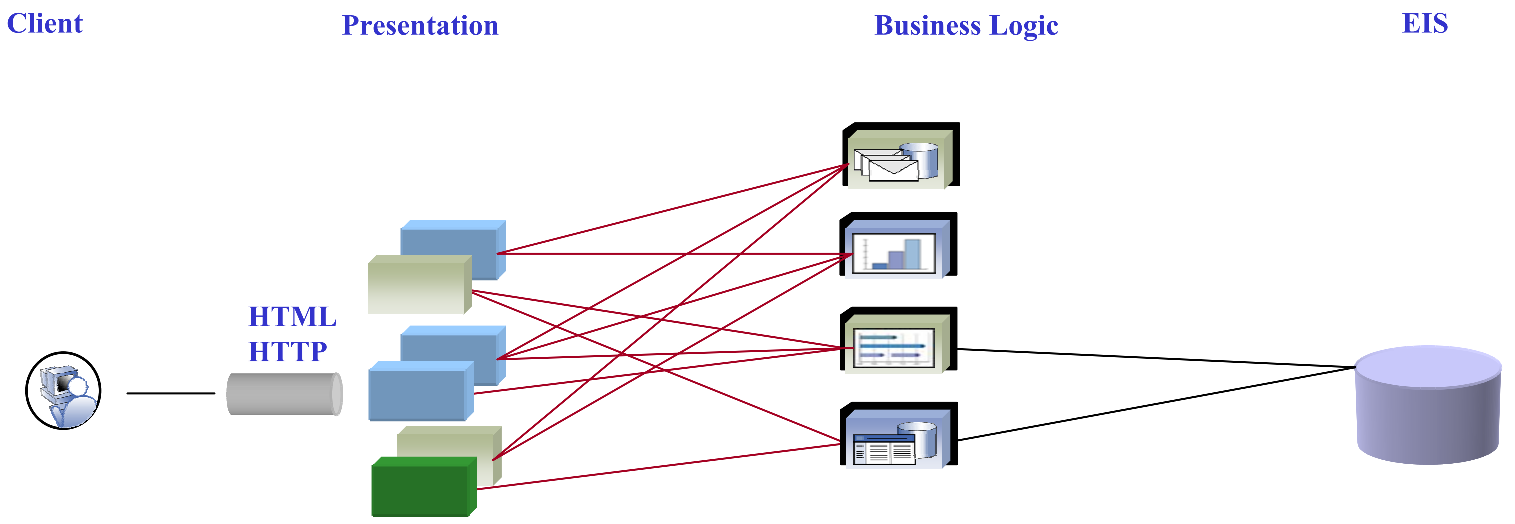
3-tier Architecture



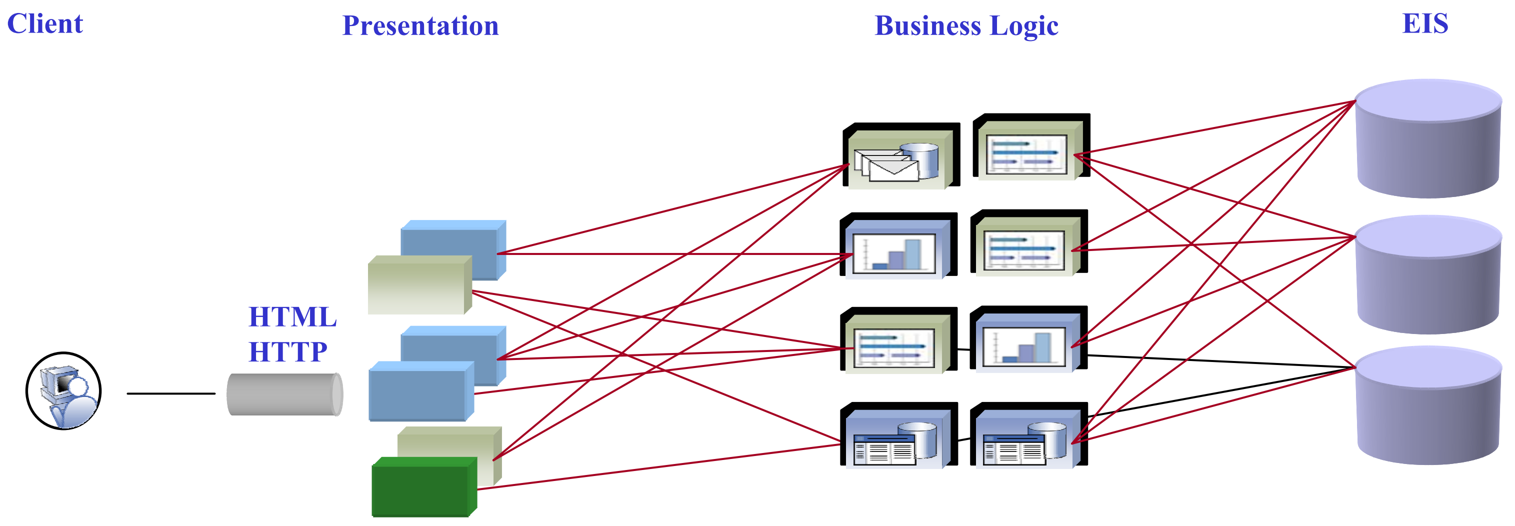
4-tier Architecture



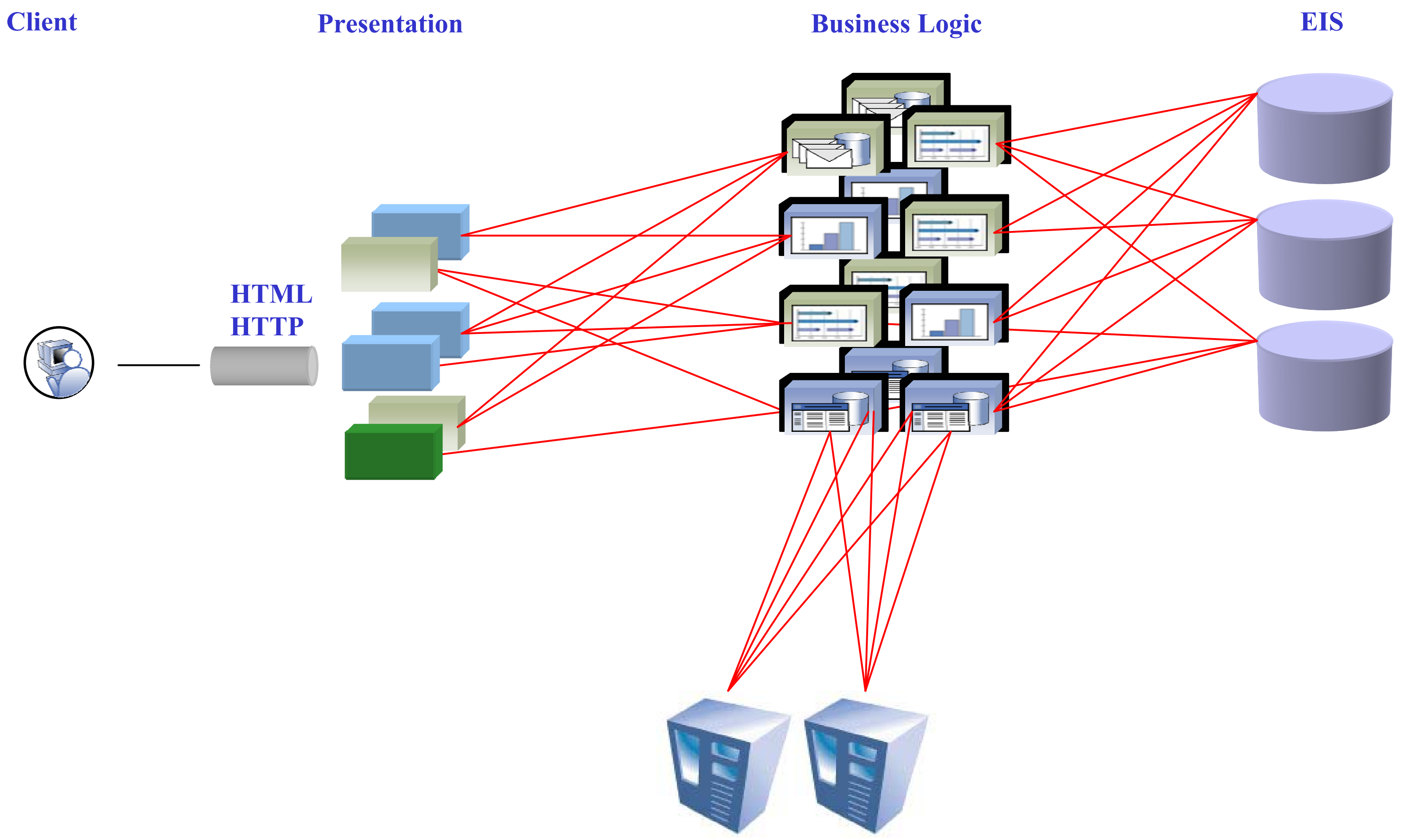
4-tier Architecture



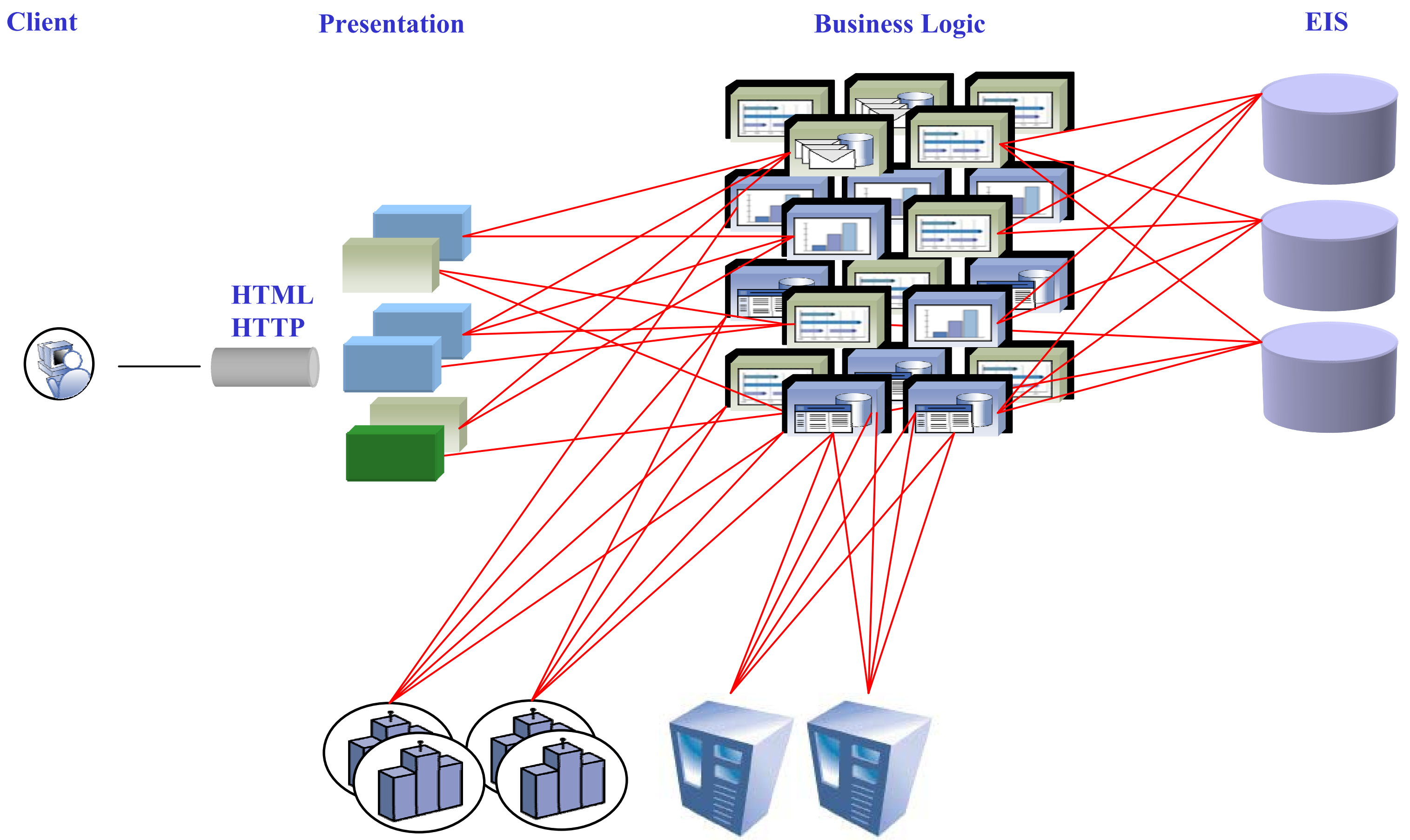
4-tier Architecture



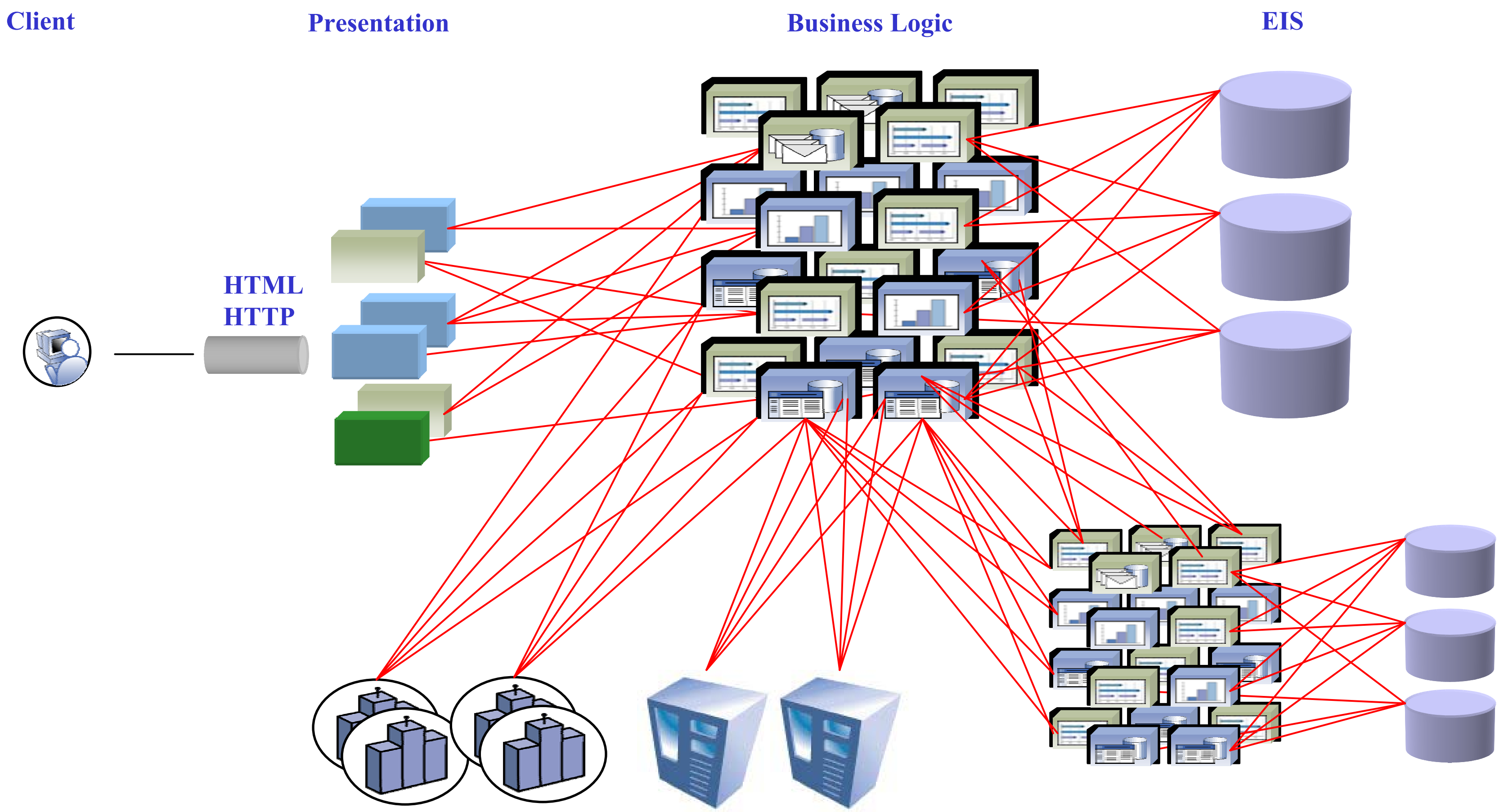
4-tier Architecture



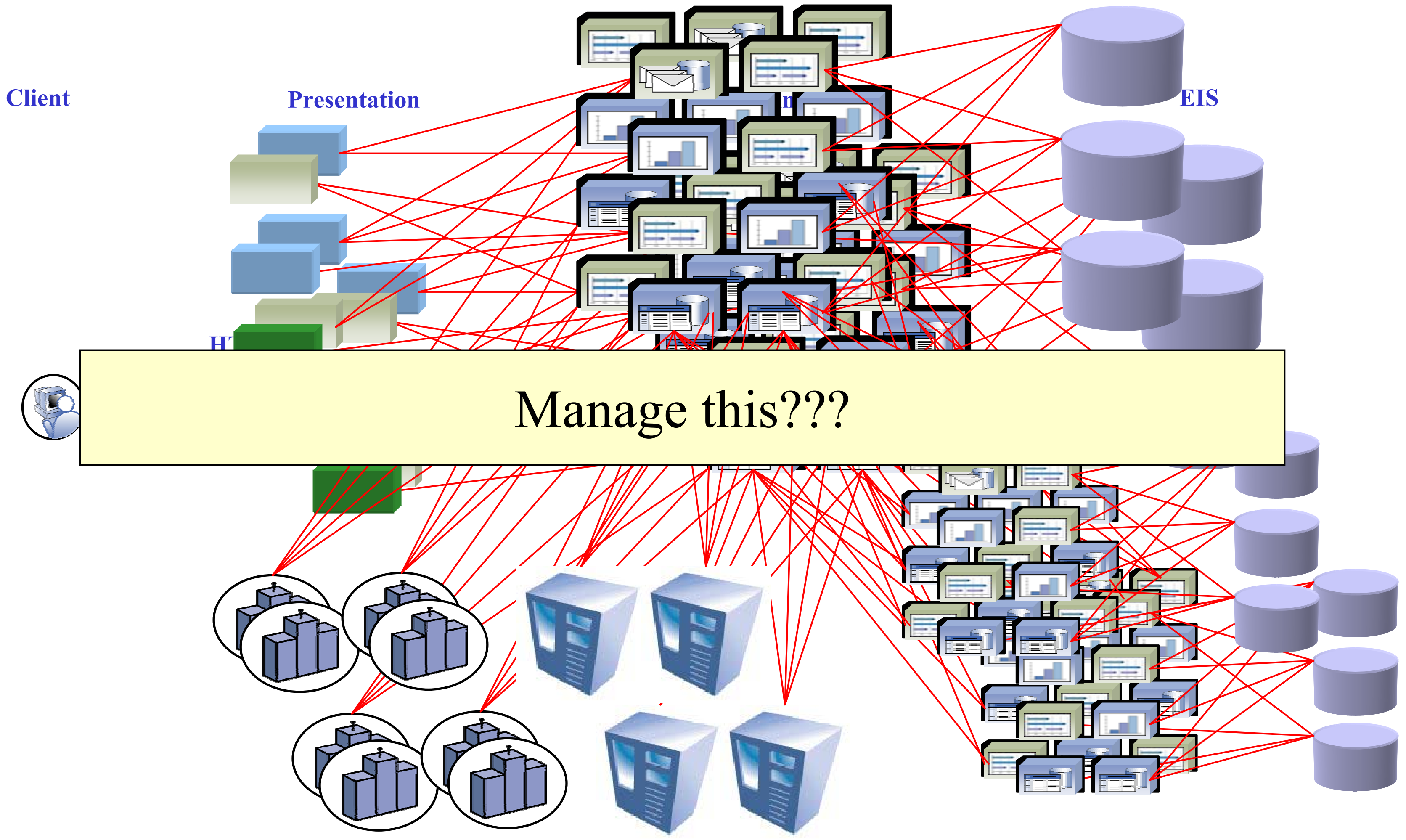
4-tier Architecture



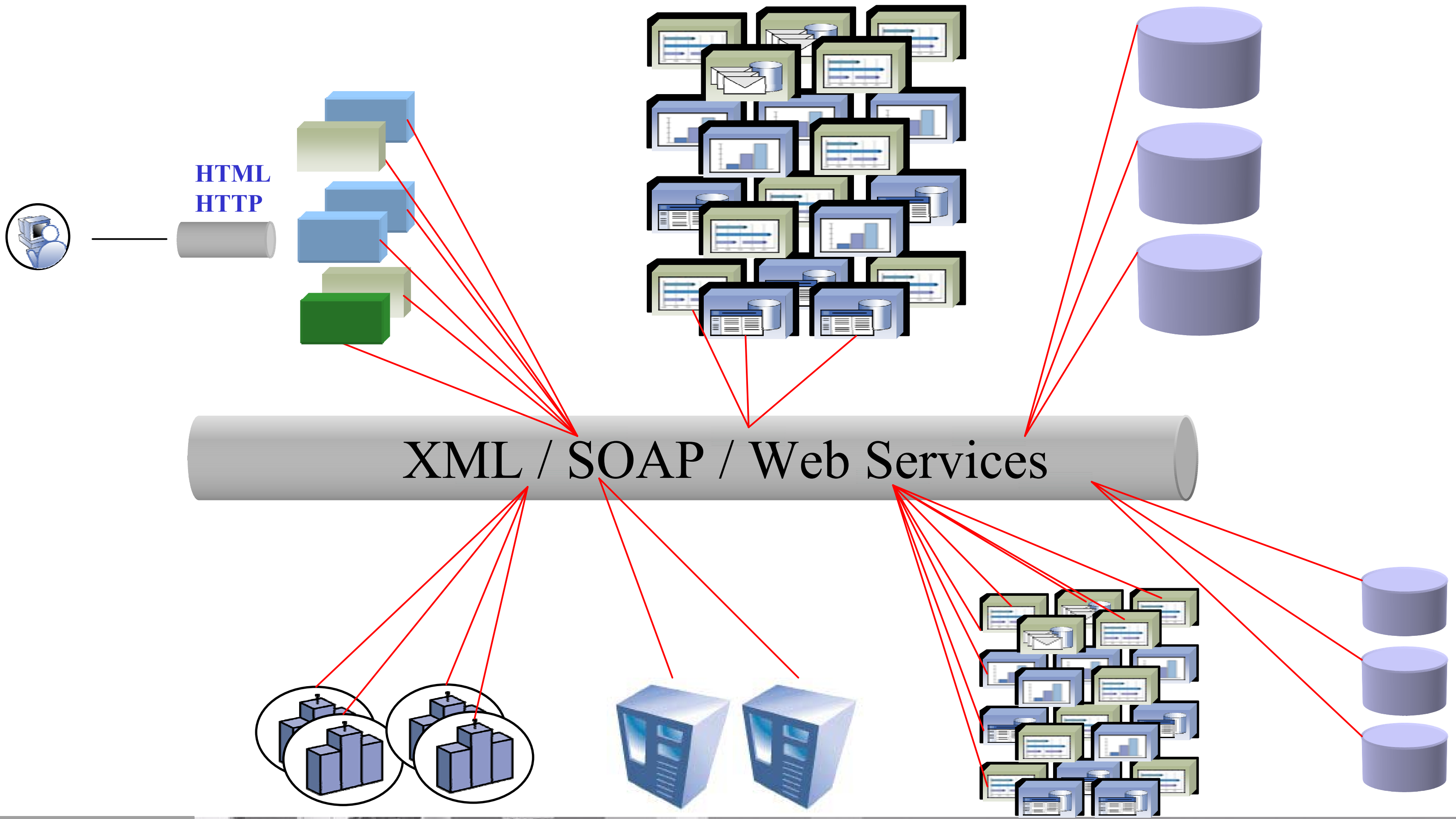
4-tier Architecture



4-tier Architecture



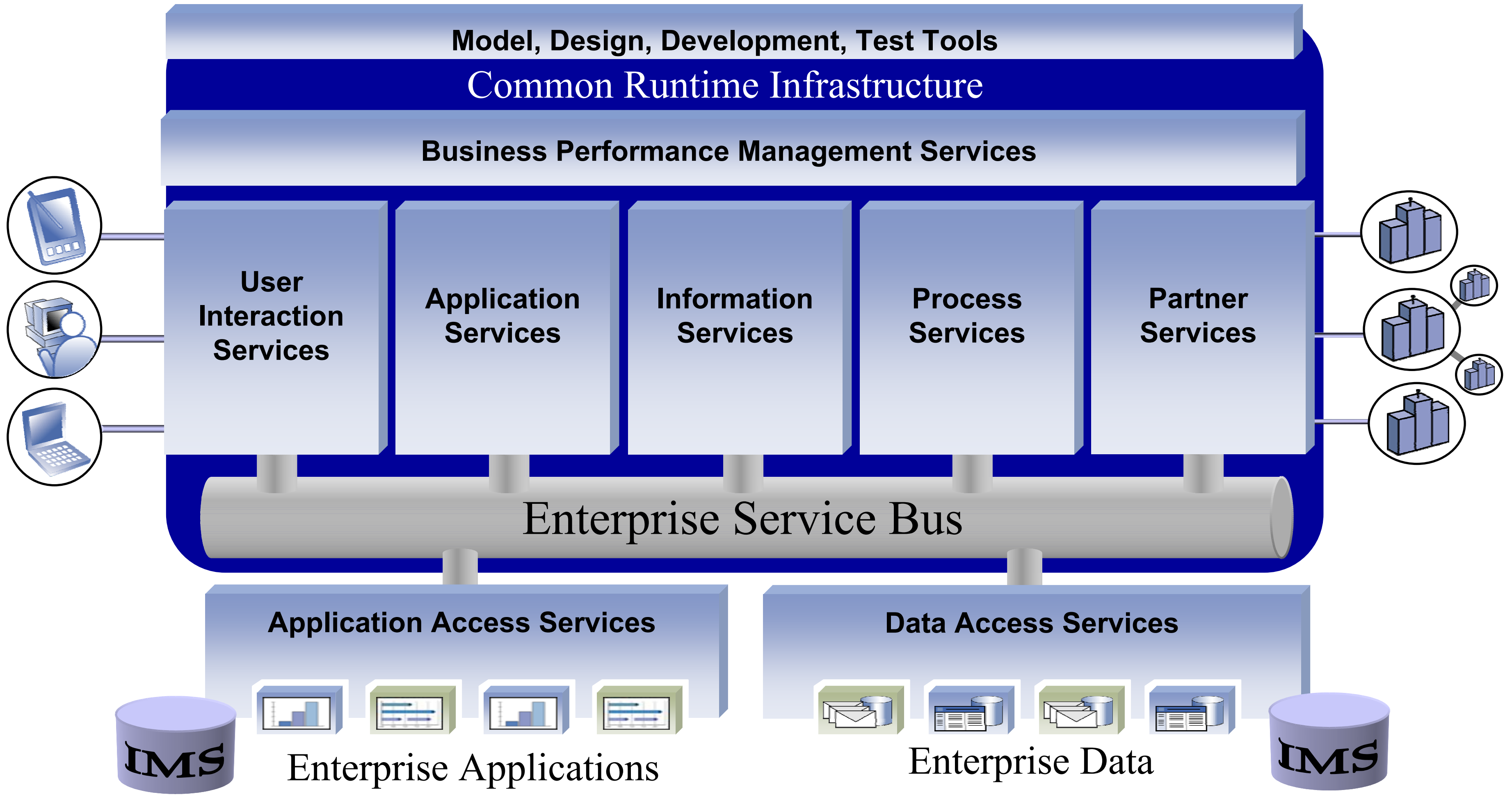
XML / SOAP / Web Services



e-business powered by IMS

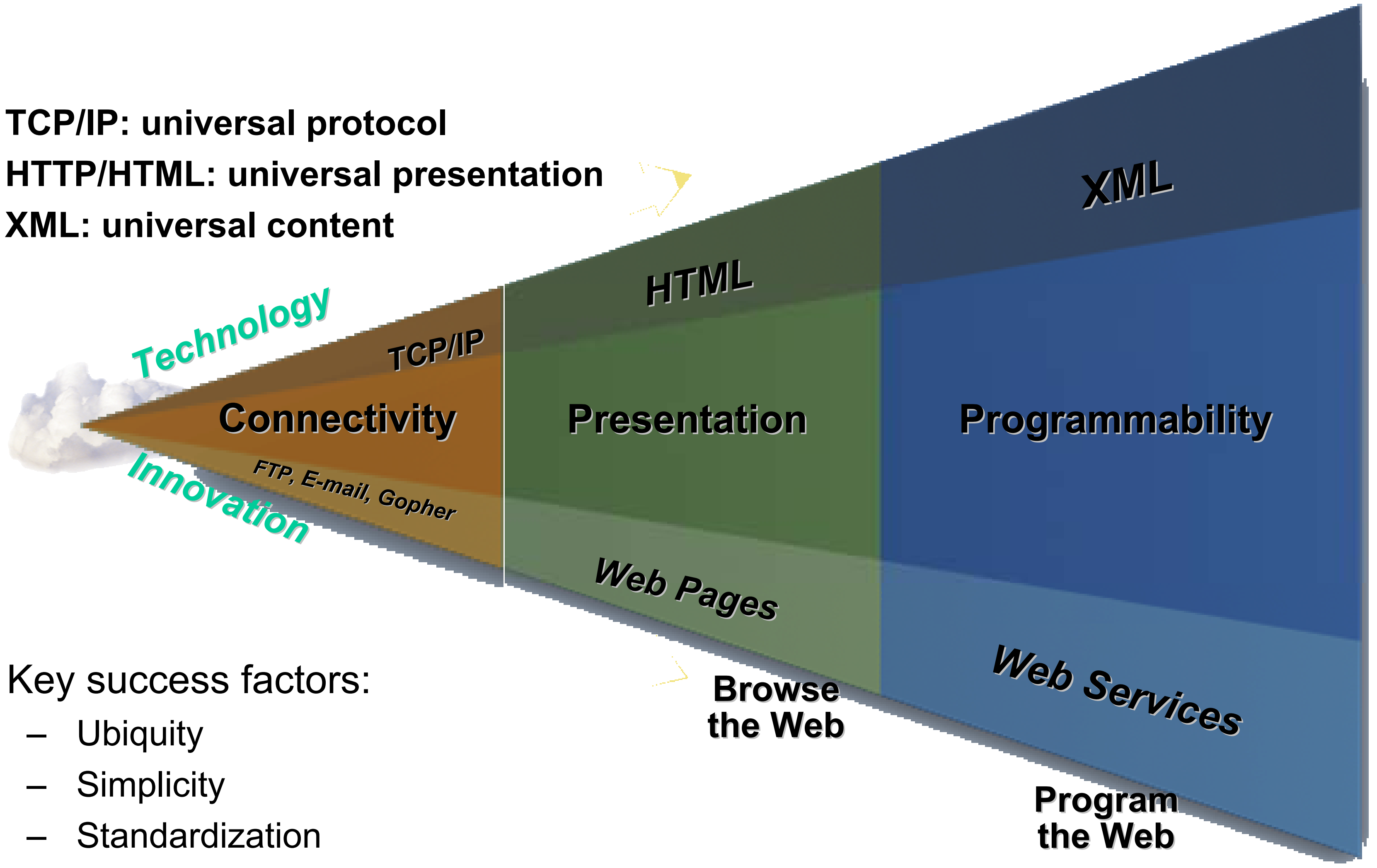
IBM Service Oriented Architecture

ESB Is a Defining Element


Model, Design, Development, Test Tools
Common Runtime Infrastructure
Business Performance Management Services
User Interaction Services
Application Services
Information Services
Process Services
Partner Services
Enterprise Service Bus
Application Access Services
Data Access Services
IMS
Enterprise Applications
Enterprise Data
IMS

The Path to Web Services

- **TCP/IP: universal protocol**
- **HTTP/HTML: universal presentation**
- **XML: universal content**



- Key success factors:
  - Ubiquity
  - Simplicity
  - Standardization

Goals / Requirements for the SOA

1. Interoperability between systems and programming languages:
   ➔ *communication protocol*

2. A platform independent syntax for describing the interfaces to the functional units (services):
   ➔ *interface definition language*

3. Mechanism for discovering a service for use at run time or during the design phase:
   ➔ *computer* accessible *search & discovery facilities*

4. Protect the data being exchanged between services and/or consumer of services
   ➔ *security*

Core Technologies of Web Services

XML (extensible markup language)

- underlies most of the specifications used for Web services
- generic language used to describe any kind of content in a structured way, separated from presentation on a specific device

SOAP (Simple Object and Access Protocol)

- programming language & platform neutral protocol allowing a client to call a remote service (XML based RPC & messaging protocol)

WSDL (Web Services Description Language)

- XML based interface & implementation description language
- defines service access information

UDDI (Universal Description Discovery and Integration)

- client side API and SOAP based server implementation used to store & retrieve information of service providers and Web services (registry mechanism)

What is XML

- A Standardized, Simple, and Self-Describing Markup Language for documents containing structured or semi-structured information.

```
<?xml version="1.1"?>
<Presentation>
   <title>XML-DB</title>
   <length>60</length>
   <presenter>
      <lastName>Holtz</lastName>
      <firstName>Christopher</firstName>
   </presenter>
   <Comments session="e-business Design Review">
      <comment>Loved It</comment>
      <comment>Can't wait for GA</comment>
   </Comments>
</Presentation>
```

Why XML…

- Standard Internet Data Exchange Format
  - Self-Describing
  - Handles encoding

    <xml? version="1.1" encoding="ebcdic-cp-us"?>

  - Handles byte ordering

    <OrderNumber>**110203**</OrderNumber>

  - Human Legible?
  - Easily Parsed

  - **Standard!**

The XML Schema Definition Language

An XML language for defining the legal building blocks of a valid XML document

An XML Schema:

- defines elements and attributes that can appear in a document
- defines which elements are child elements
- defines the order and number of child elements
- defines whether an element is empty or can include text
- defines data types for elements and attributes
- defines default and fixed values for elements and attributes

Defines an agreed upon communication contract for exchanging XML documents

Core Technologies of Web Services

XML (extensible markup language)

- underlies most of the specifications used for Web services
- generic language used to describe any kind of content in a structured way, separated from presentation on a specific device

SOAP (Simple Object and Access Protocol)

- programming language & platform neutral protocol allowing a client to call a remote service (XML based RPC & messaging protocol)

WSDL (Web Services Description Language)

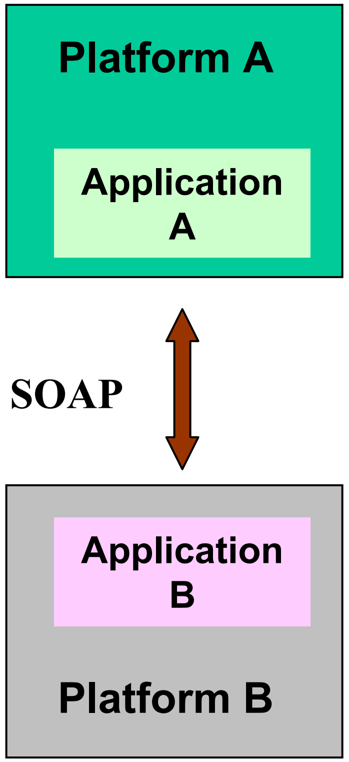
- XML based interface & implementation description language
- defines service access information

UDDI (Universal Description Discovery and Integration)

- client side API and SOAP based server implementation used to store & retrieve information of service providers and Web services (registry mechanism)

Simple Object Access Protocol (SOAP)

- What is SOAP?
  - **XML based protocol for exchanging structured information in a loosely-coupled distributed environment**
  - **Communicate with and expose features to distributed applications**
  - **Separate content from mode of transport**
  - **Platform-independent, language-neutral**
  - **Open standard**
  - **Simple Messaging to RPC**

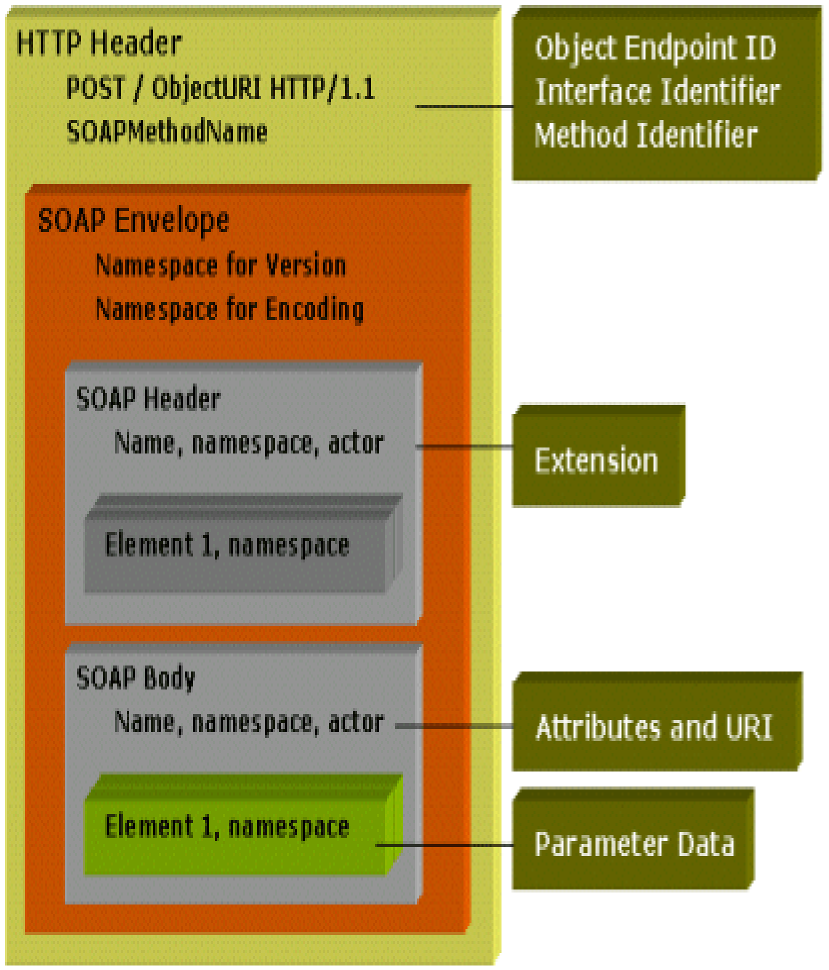


$$\boxed{\text{SOAP} = \frac{\text{XML}}{\text{HTTP}}}$$

The SOAP 1.1 construct

A SOAP message contains 4 parts:

1. Envelope: defines the content of the message
2. Header (optional): contains header information
3. Body: contains call and response information
4. Fault : contains errors



Build input message in XML

SOAP Client
(web page,
client application,
another web service,
etc.)

```
<arg0 xsi:type="ns7:INPUTMSG"
        xmlns:ns7="http://sample/">
   <in__ll>32</in__ll>
   <in__zz>0</in__zz>
   <in__trcd>IVTNO</in__trcd>
   <in__cmd>DISPLAY</in__cmd>
   <in__name1>HOLTZ</in__name1>
   <in__name2 xsi:nil="true"/>
   <in__extn xsi:nil="true"/>
   <in__zip xsi:nil="true"/>
</arg0>
```

SOAP Body

- Body (required)
  - Contains application-specific message
  - May be encoded variously

```
<soapenv:Body>
   <arg0 xsi:type="ns7:INPUTMSG"
          xmlns:ns7="http://sample/">
      <in__ll>32</in__ll>
      <in__zz>0</in__zz>
      <in__trcd>IVTNO</in__trcd>
      <in__cmd>DISPLAY</in__cmd>
      <in__name1>HOLTZ</in__name1>
      <in__name2 xsi:nil="true"/>
      <in__extn xsi:nil="true"/>
      <in__zip xsi:nil="true"/>
   </arg0>
</soapenv:Body>
```

SOAP Header

- Header (optional)
  - Contains metadata entries about message
  - Specifies which entries must be understood and by which target “actor” in chain of recipients
  - Declares encoding rules (optional)

```
<soapenv:Body>
  <arg0 xsi:type="ns7:INPU
        xmlns:ns7="http://s
    <in__ll>32</in__ll>
    <in__zz>0</in__zz>
    <in__trcd>IVTNO</in_
    <in__cmd>DISPLAY</
    <in__name1>HOLTZ<
    <in__name2 xsi:nil="tr
    <in__extn xsi:nil="true
    <in__zip xsi:nil="true"
  </arg0>
</soapenv:Body>
```

```
<soapenv:Header>
  <ns1:IMSService  xmlns:ns3="IMSSOAP"
       soapenv:mustUnderstand="0" xsi:type="xsd:string">
    Phone Book Service
  </ns1:IMSService>
</soapenv:Header>
```

SOAP Fault

- Fault element (optional)
  - Contained in Body
  - Describes error class (version mismatch, headers not understood, client error, server error)

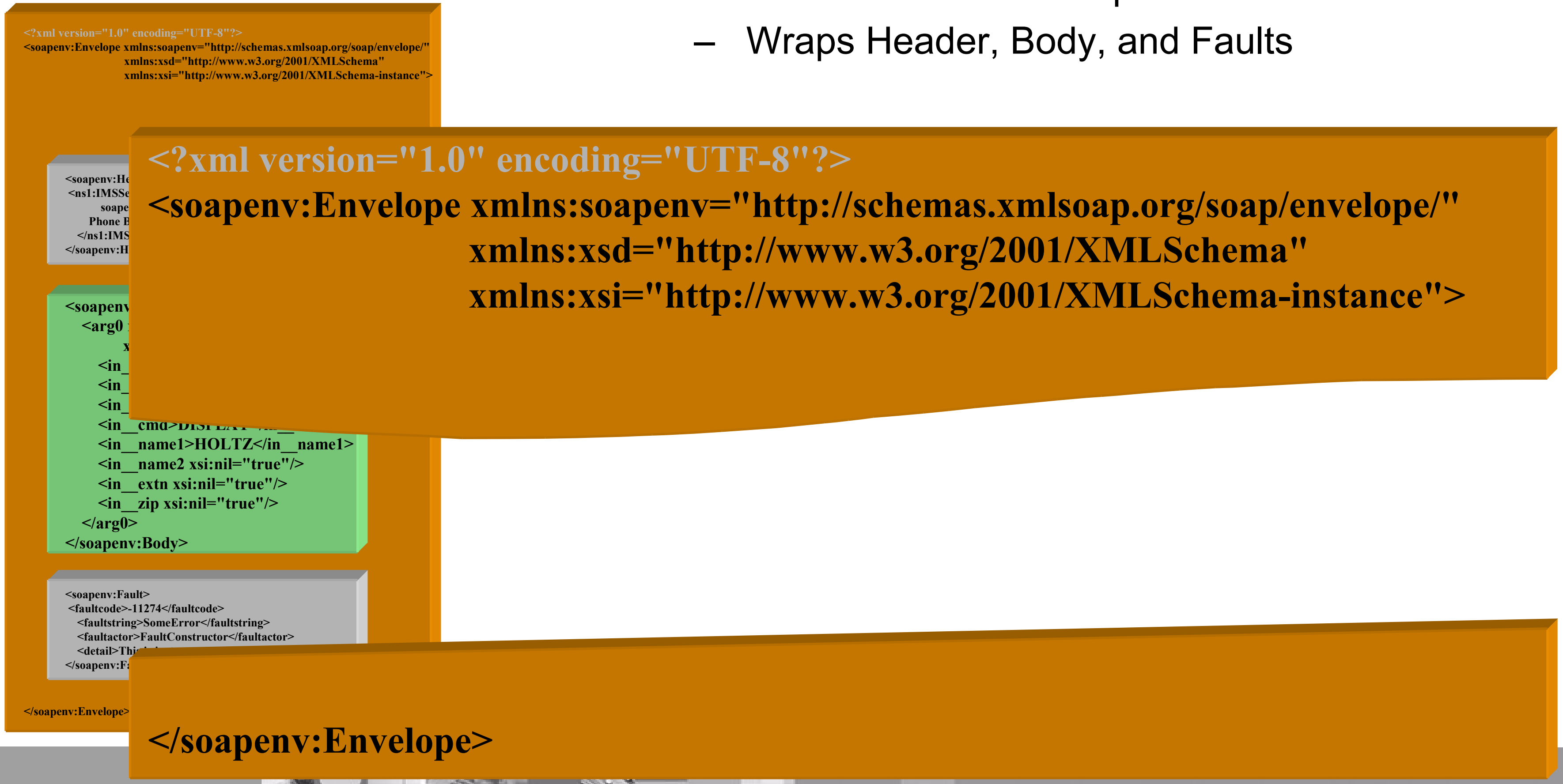
```
<soapenv:Header>
 <ns1:IMSService  xmlns:ns3="IMSSOAP"
          soapenv:mustUnderstand="0" xsi:type="xsd:string">
       Phone Book Service
    </ns1:IMSService>
</soapenv:Header>
```

```
<soapenv:Body>
   <arg0 xsi:type="ns7:INPU
           xmlns:ns7="http://s
      <in__ll>32</in__ll>
      <in__zz>0</in__zz>
      <in__trcd>IVTNO</in_
      <in__cmd>DISPLAY</
      <in__name1>HOLTZ<
      <in__name2 xsi:nil="tr
      <in__extn xsi:nil="true
      <in__zip xsi:nil="true"
   </arg0>
</soapenv:Body>
```

```
<soapenv:Fault>
   <faultcode>-11274</faultcode>
   <faultstring>SomeError</faultstring>
   <faultactor>FaultConstructor</faultactor>
   <detail>This is just an example fault</detail>
</soapenv:Fault>
```

SOAP Envelope

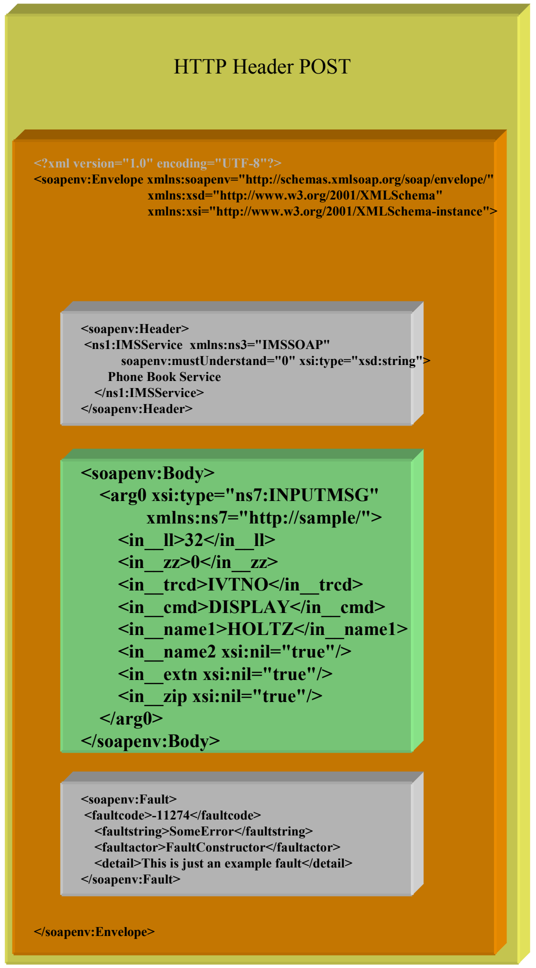
- Envelope (required)
  - Defines SOAP namespace
  - Wraps Header, Body, and Faults



```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <soapenv:He
   <ns1:IMSSe
       soape
     Phone B
   </ns1:IMS
  </soapenv:H

  <soapenv
    <arg0
          x
      <in_
      <in_
      <in_
      <in__cmd>DISPLAY</in__
      <in__name1>HOLTZ</in__name1>
      <in__name2 xsi:nil="true"/>
      <in__extn xsi:nil="true"/>
      <in__zip xsi:nil="true"/>
    </arg0>
  </soapenv:Body>

  <soapenv:Fault>
   <faultcode>-11274</faultcode>
     <faultstring>SomeError</faultstring>
     <faultactor>FaultConstructor</faultactor>
     <detail>Thi
  </soapenv:F

</soapenv:Envelope>
```

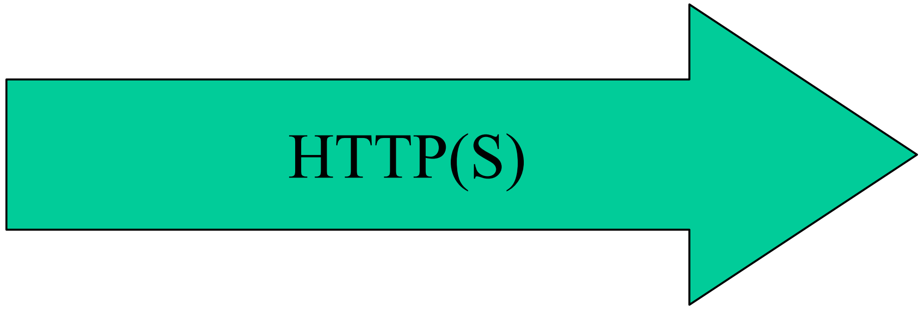
```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

</soapenv:Envelope>
```

HTTP Header and SOAP Call

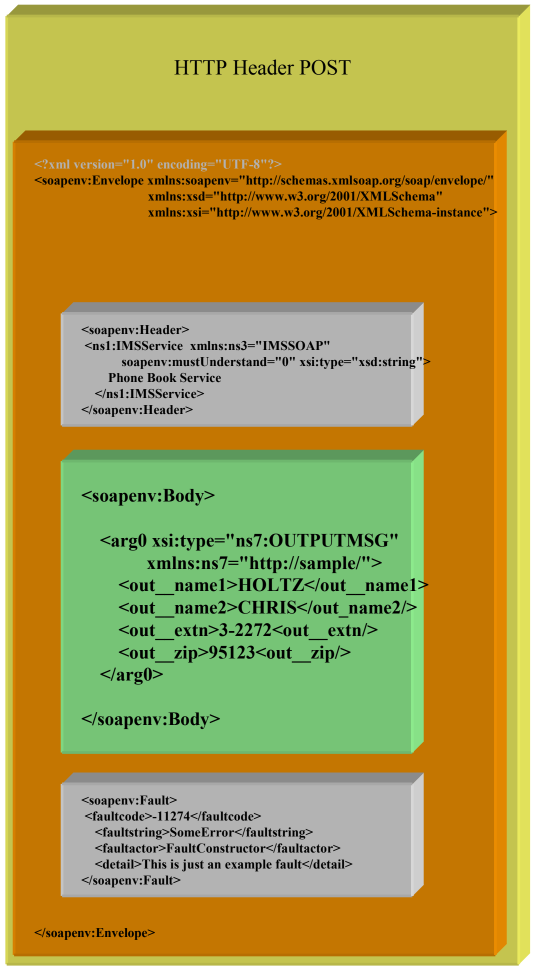


```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <soapenv:Header>
   <ns1:IMSService  xmlns:ns3="IMSSOAP"
         soapenv:mustUnderstand="0" xsi:type="xsd:string">
       Phone Book Service
     </ns1:IMSService>
  </soapenv:Header>

  <soapenv:Body>
     <arg0 xsi:type="ns7:INPUTMSG"
           xmlns:ns7="http://sample/">
       <in__ll>32</in__ll>
       <in__zz>0</in__zz>
       <in__trcd>IVTNO</in__trcd>
       <in__cmd>DISPLAY</in__cmd>
       <in__name1>HOLTZ</in__name1>
       <in__name2 xsi:nil="true"/>
       <in__extn xsi:nil="true"/>
       <in__zip xsi:nil="true"/>
     </arg0>
  </soapenv:Body>

  <soapenv:Fault>
   <faultcode>-11274</faultcode>
     <faultstring>SomeError</faultstring>
     <faultactor>FaultConstructor</faultactor>
     <detail>This is just an example fault</detail>
  </soapenv:Fault>

</soapenv:Envelope>
```

- HTTP Header
  - POST call
  - Security

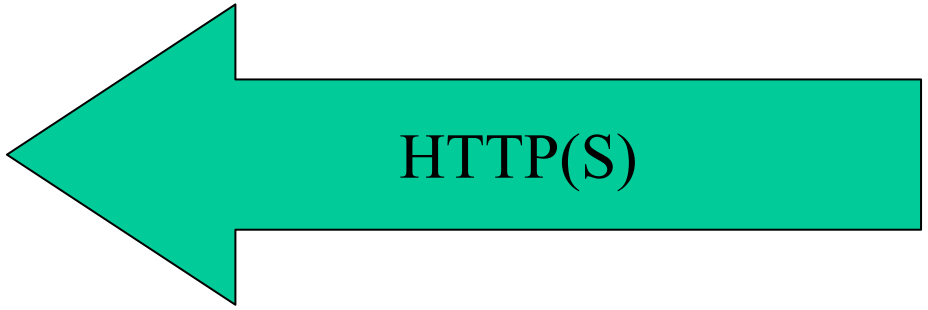


HTTP SOAP Response

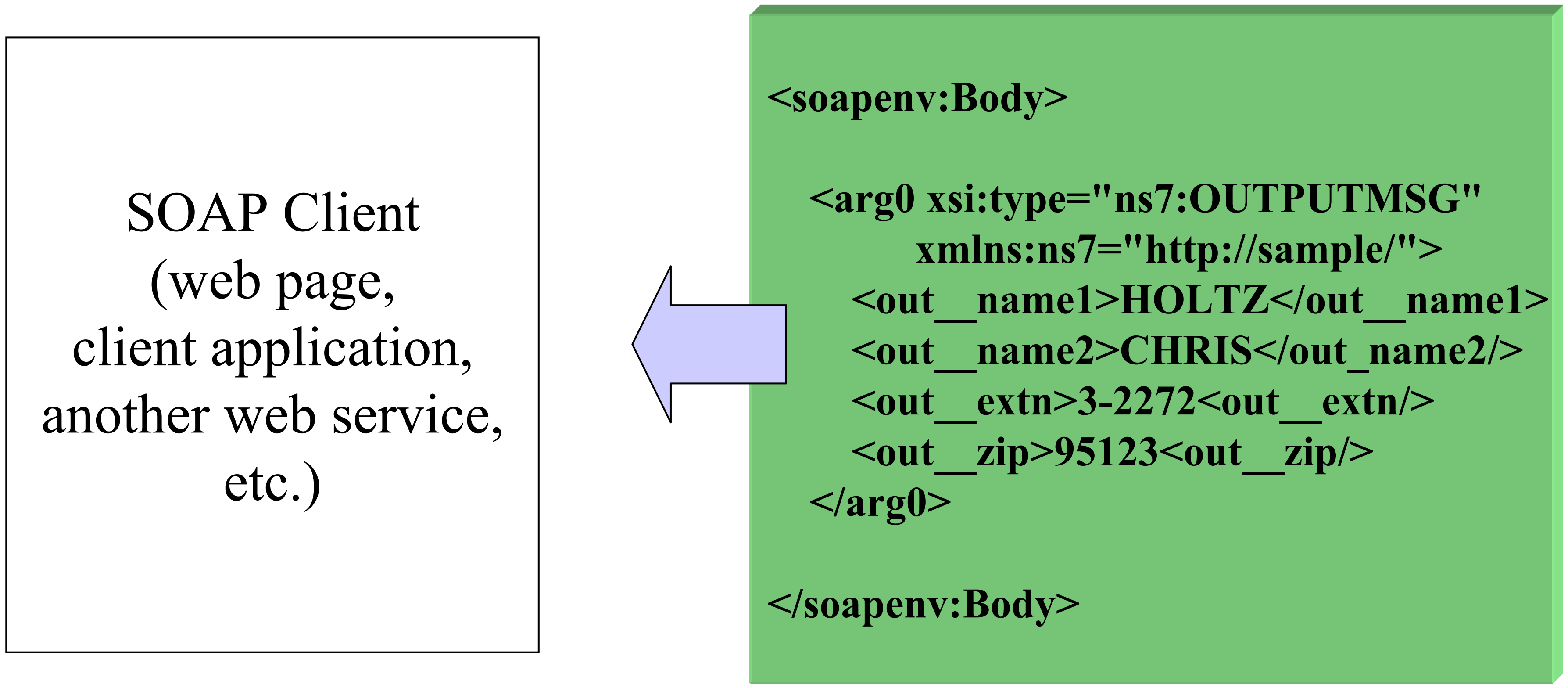


```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <soapenv:Header>
   <ns1:IMSService  xmlns:ns3="IMSSOAP"
         soapenv:mustUnderstand="0" xsi:type="xsd:string">
       Phone Book Service
     </ns1:IMSService>
  </soapenv:Header>

  <soapenv:Body>

     <arg0 xsi:type="ns7:OUTPUTMSG"
           xmlns:ns7="http://sample/">
        <out__name1>HOLTZ</out__name1>
        <out__name2>CHRIS</out_name2/>
        <out__extn>3-2272<out__extn/>
        <out__zip>95123<out__zip/>
     </arg0>

  </soapenv:Body>

  <soapenv:Fault>
   <faultcode>-11274</faultcode>
     <faultstring>SomeError</faultstring>
     <faultactor>FaultConstructor</faultactor>
     <detail>This is just an example fault</detail>
  </soapenv:Fault>

</soapenv:Envelope>
```

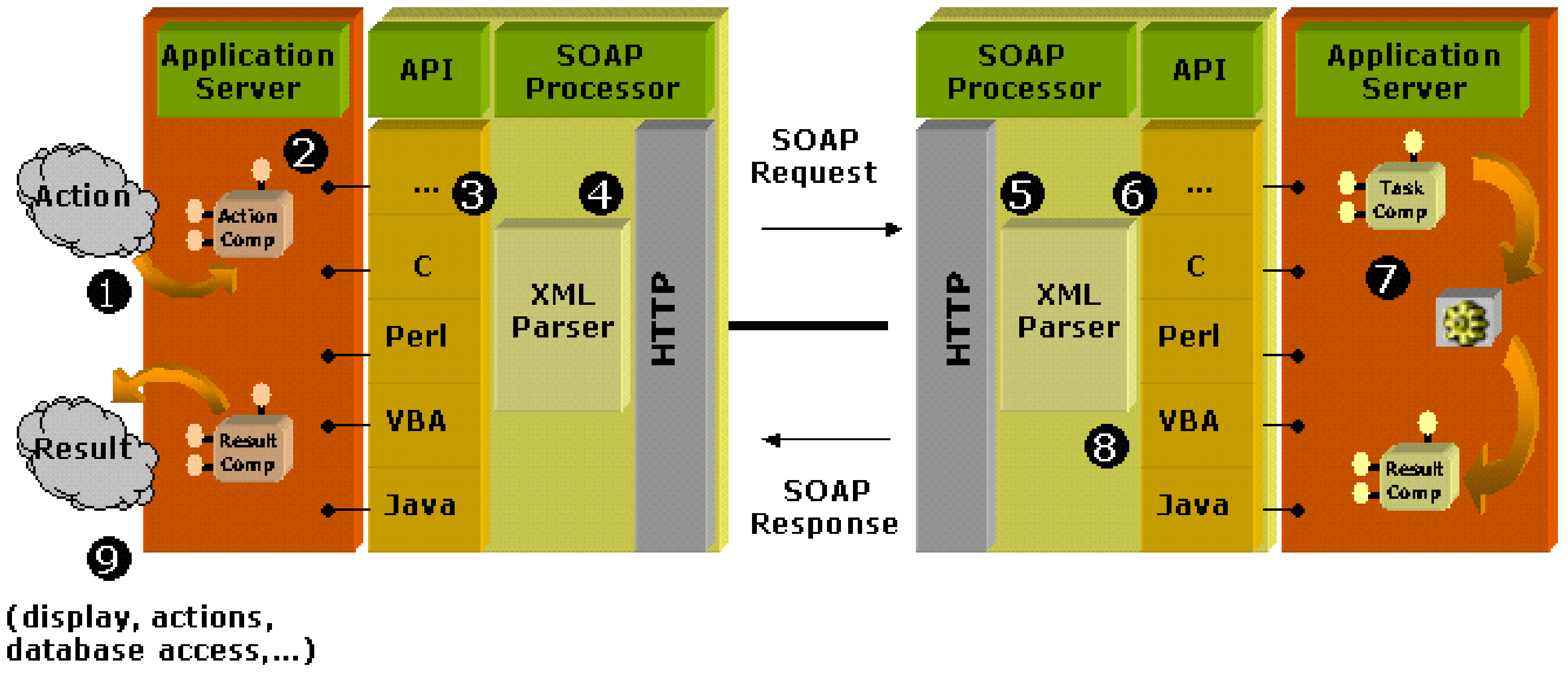
- Faults can be examined
- Output XML is extracted



Convert Output Message From XML



```
<soapenv:Body>

  <arg0 xsi:type="ns7:OUTPUTMSG"
        xmlns:ns7="http://sample/">
    <out__name1>HOLTZ</out__name1>
    <out__name2>CHRIS</out_name2/>
    <out__extn>3-2272<out__extn/>
    <out__zip>95123<out__zip/>
  </arg0>

</soapenv:Body>
```



- A SOAP client formats a message in XML including a SOAP “envelope” element describing the message
- The client sends the message to a SOAP server in the body of an HTTP request
- The server determines whether the message is valid and supported
- The server formats its response in XML and sends it to the client in the body of an HTTP response

Core Technologies of Web Services

XML (extensible markup language)
- underlies most of the specifications used for Web services
- generic language used to describe any kind of content in a structured way, separated from presentation on a specific device

SOAP (Simple Object and Access Protocol)
- programming language & platform neutral protocol allowing a client to call a remote service (XML based RPC & messaging protocol)

WSDL (Web Services Description Language)
- XML based interface & implementation description language
- defines service access information

UDDI (Universal Description Discovery and Integration)
- client side API and SOAP based server implementation used to store & retrieve information of service providers and Web services (registry mechanism)

Interface Definition Language - WSDL

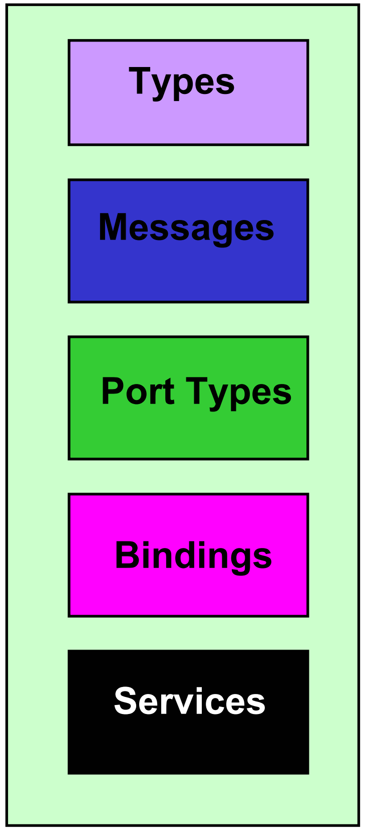
- A Web Services Description Language document describes
  - **Who** owns and is publishing the service
  - **What** the service does
  - **Where** the service resides
  - **How** to invoke the service
- An XML Vocabulary
  - Similar in purpose to IDL but is XML based
- Defines binding for SOAP, HTTP GET/POST and MIME

Provides everything an application needs to communicates with a web service

WSDL Parts

- Types
  - Used to define custom message types
- Messages
  - Abstraction of request and response messages that the client and service need to communicate.
- PortTypes
  - Contains a set of operations.
  - Operations organize WSDL messages.
  - Operation->method name, PortType->java interface
- Bindings
  - Binds the PortType to a specific protocol (typically SOAP over http).
  - You can bind one PortType to several different protocols by using more than one port.
- Services
  - Gives you one or more URLs for the service.

WSDL1.1 Document



PhoneBook Example

WSDL 1.1 Document

Cobol Source

```
01  INPUT-MESSAGE.
    02  IN-LL              PICTURE S9(3) COMP.
    02  IN-ZZ              PICTURE S9(3) COMP.
    02  IN-TRANCODE        PICTURE X(9).
    02  IN-PERSON-NUMBER   PICTURE X(6).

01  OUTPUT-MESSAGE.
    02  OUT-LL         PICTURE S9(3) COMP VALUE +0.
    02  OUT-ZZ         PICTURE S9(3) COMP VALUE +0.
    02  OUT-LASTNAME   PICTURE X(20) VALUE SPACES.
    02  OUT-FIRSTNAME  PICTURE X(20) VALUE SPACES.
```

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="IVTNOService"
  targetNamespace="http://ims.soap.IVTNOService"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:tns="http://ims.soap.IVTNOService"
…
  <complexType name="INPUTMSG">
    <sequence>
      <element name="in__ll">
        <simpleType>
          <restriction base="short"/>
        </simpleType>
      </element>
      <element name="in__zz">
        <simpleType>
          <restriction base="short"/>
        </simpleType>
      </element>
 …
```

Types

WSDL PhoneBook Example

Messages

Port Types

Bindings

Services

```
<message name="runIVTNOServiceRequest">
    <part element="xsd1:INPUTMSG" name="INPUT-MSGPart"/>
  </message>
  <message name="runIVTNOServiceResponse">
    <part element="xsd2:OUTPUTMSG" name="OUTPUT-MSGPart"/>
  </message>
  <portType name="IVTNOServiceSOAPIMS">
    <operation name="runIVTNOService">
      <input message="tns:runIVTNOServiceRequest" name="runIVTNOServiceRequest"/>
      <output message="tns:runIVTNOServiceResponse" name="runIVTNOServiceResponse"/>
    </operation>
  </portType>
  <binding name="IVTNOServiceIMSBinding" type="tns:IVTNOServiceSOAPIMS">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="runIVTNOService">
      <soap:operation soapAction="urn:IVTNO" style="document"/>
      <input name="runIVTNOServiceRequest">
        <soap:body encodingStyle="literal" parts="INPUT-MSGPart" use="literal"/>
      </input>
      <output name="runIVTNOServiceResponse">
        <soap:body encodingStyle="literal"
          parts="OUTPUT-MSGPart" use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="IVTNOServiceIMSService">
    <port binding="tns:IVTNOServiceIMSBinding" name="IVTNOServiceSOAPIMSPort">
     <soap:address location="http://9.30.20.157:8081/axis/services/IVTNOServiceSOAPIMSPort"/>
    </port>
  </service>
```

Core Technologies of Web Services

XML (extensible markup language)

- underlies most of the specifications used for Web services
- generic language used to describe any kind of content in a structured way, separated from presentation on a specific device

SOAP (Simple Object and Access Protocol)

- programming language & platform neutral protocol allowing a client to call a remote service (XML based RPC & messaging protocol)

WSDL (Web Services Description Language)

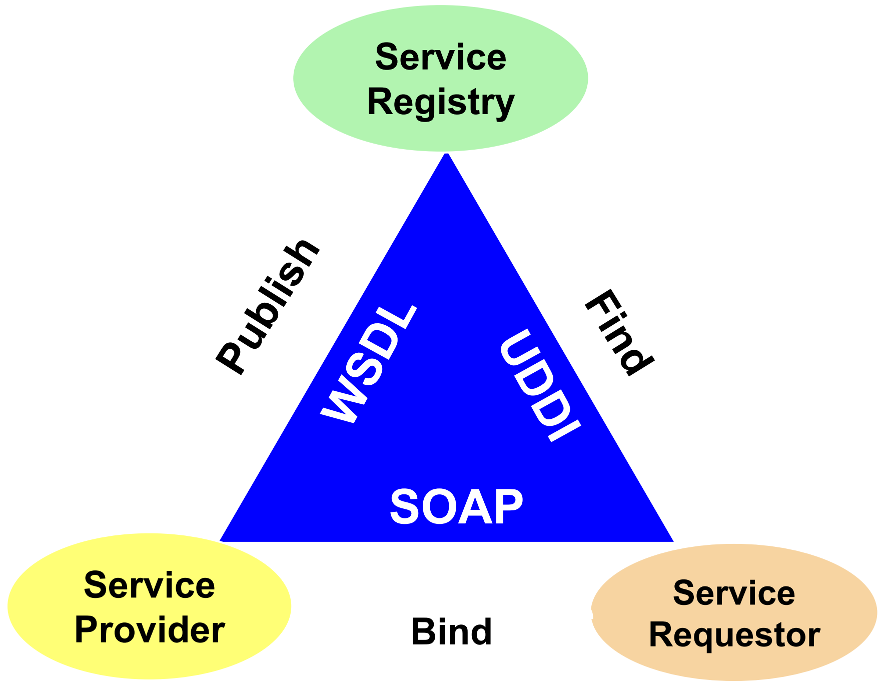
- XML based interface & implementation description language
- defines service access information

UDDI (Universal Description Discovery and Integration)

- client side API and SOAP based server implementation used to store & retrieve information of service providers and Web services (registry mechanism)

Universal Description, Discovery and Integration (UDDI)

- **Services Registry**
  - ➢Support for publishing and locating services
- **Service Provider**
  - ➢Provides e-business services
  - ➢**PUBLISHES** availability of services through registry
- **Service Requestor**
  - ➢**FINDS** needed services via Registry
  - ➢**BINDS** to services via Provider

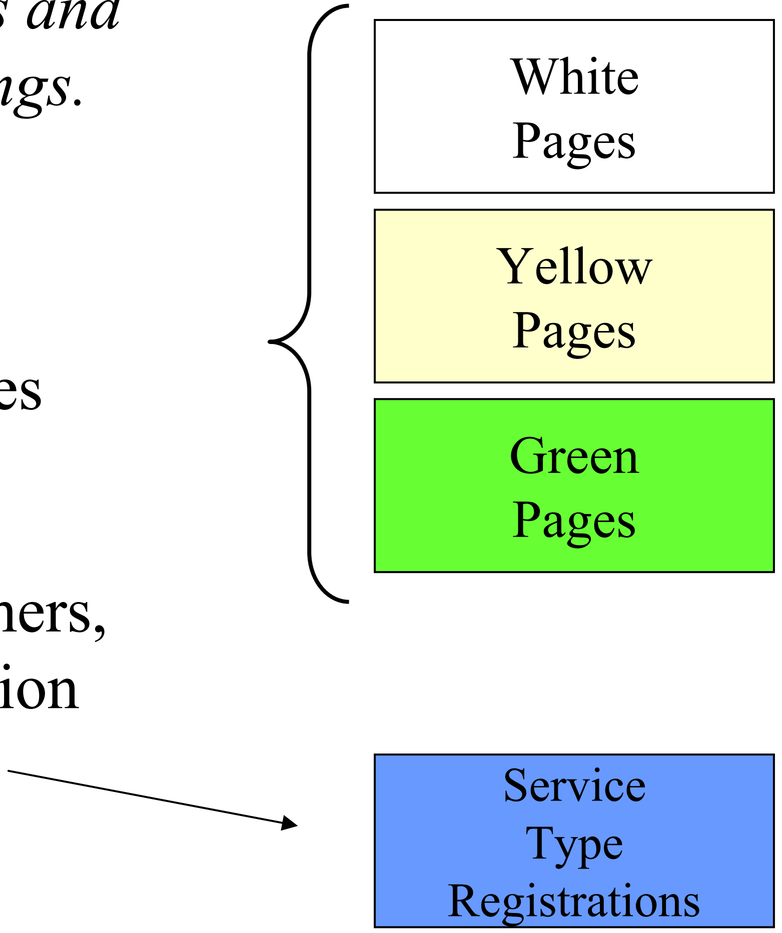


Standards Based specification for service description and discovery

Universal Description, Discovery and Integration (UDDI)

Place to advertise both the business and technical aspects of business offerings.

- Businesses register public information about themselves
- Standards bodies, programmers, businesses register information about their service types

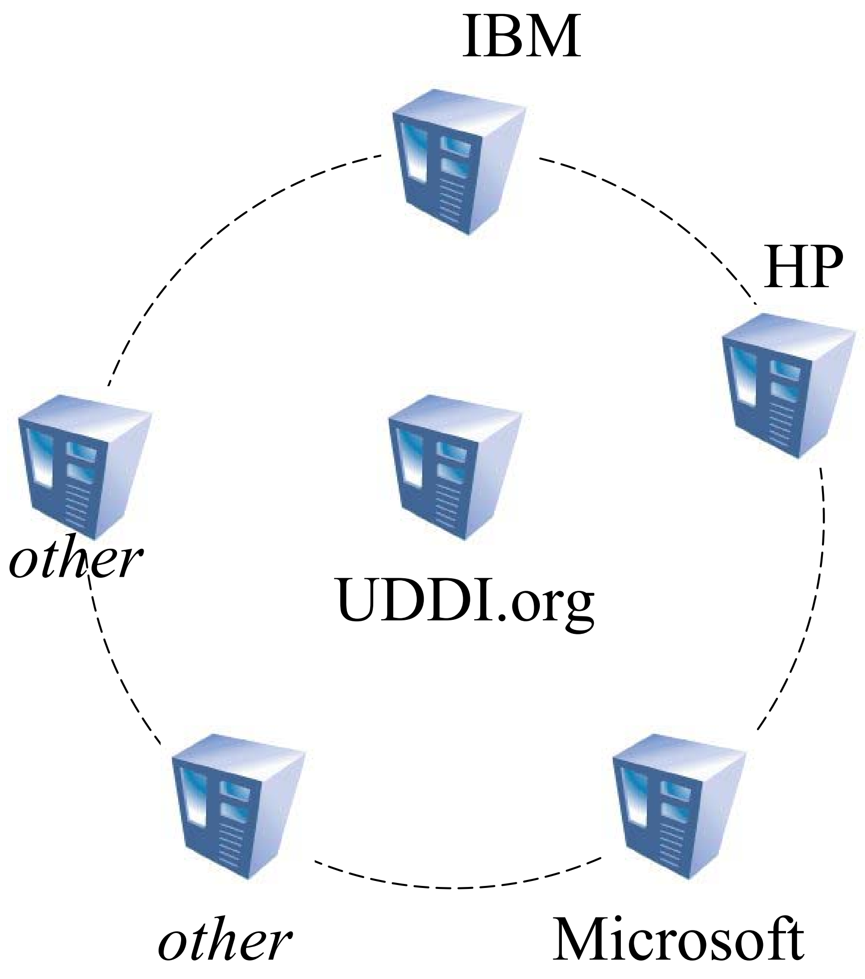


Registry Synchronization

- Peer Nodes (websites)
  - Companies Register with any node
  - Registration Replicated on a daily basis
  - Complete Set of “registered” records available at all nodes

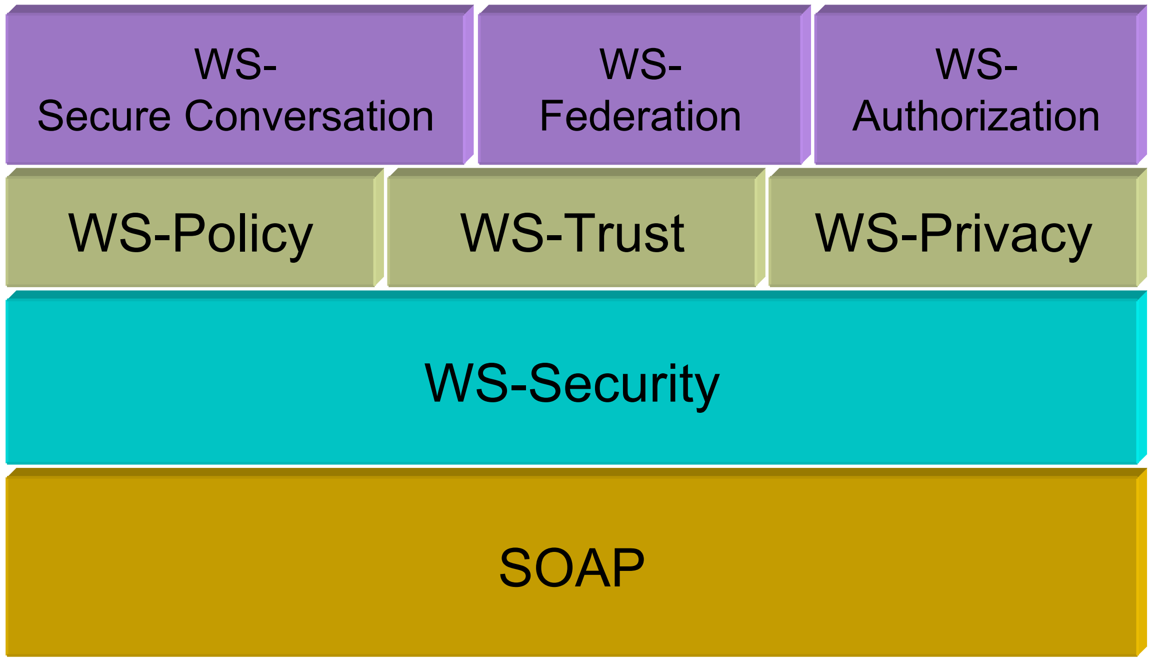
- Common Set of SOAP APIs supported by all nodes

- Compliance enforced by business contract

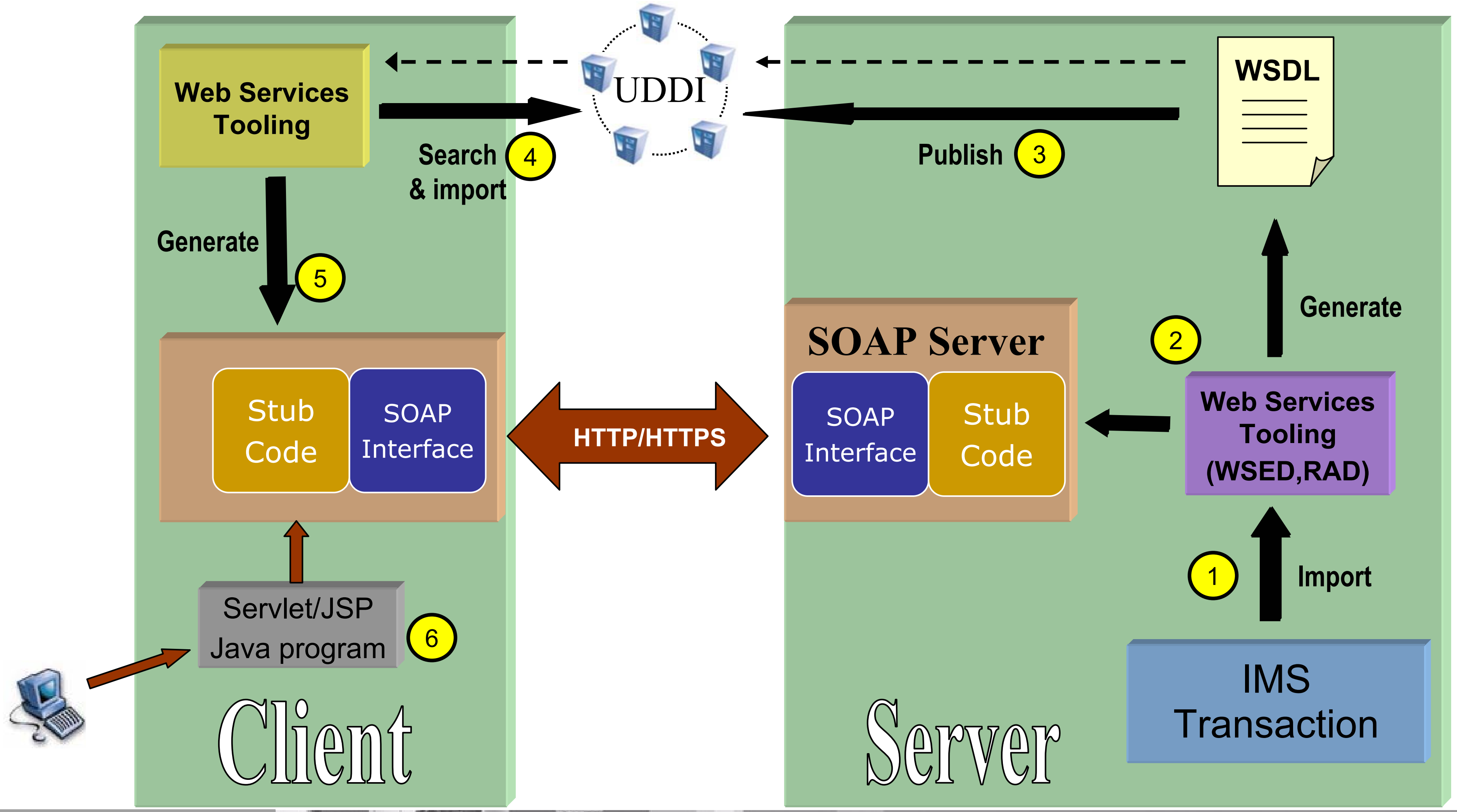


IBM’s UDDI : www-3.ibm.com/services/uddi/

Web Services Security



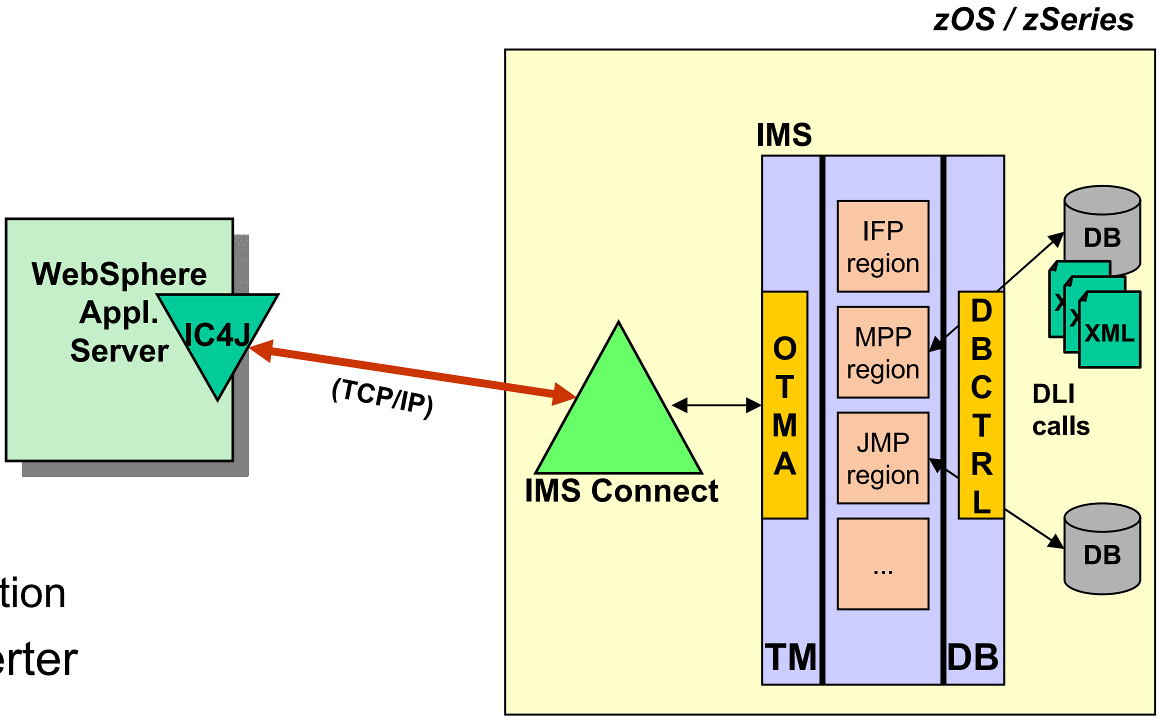
Creating and Using a Web Service



Bringing IMS into the SOA era

- OTMA
- IMS Connect
- IMS Connector for Java

- CAM + WSED / RAD
  - Web Service Generation
  - COBOL XML Converters
  - MFS Web Services Generation
- IMS Connect w/ XML Converter
- SOAP gateway



Common Application Metamodel (CAM)

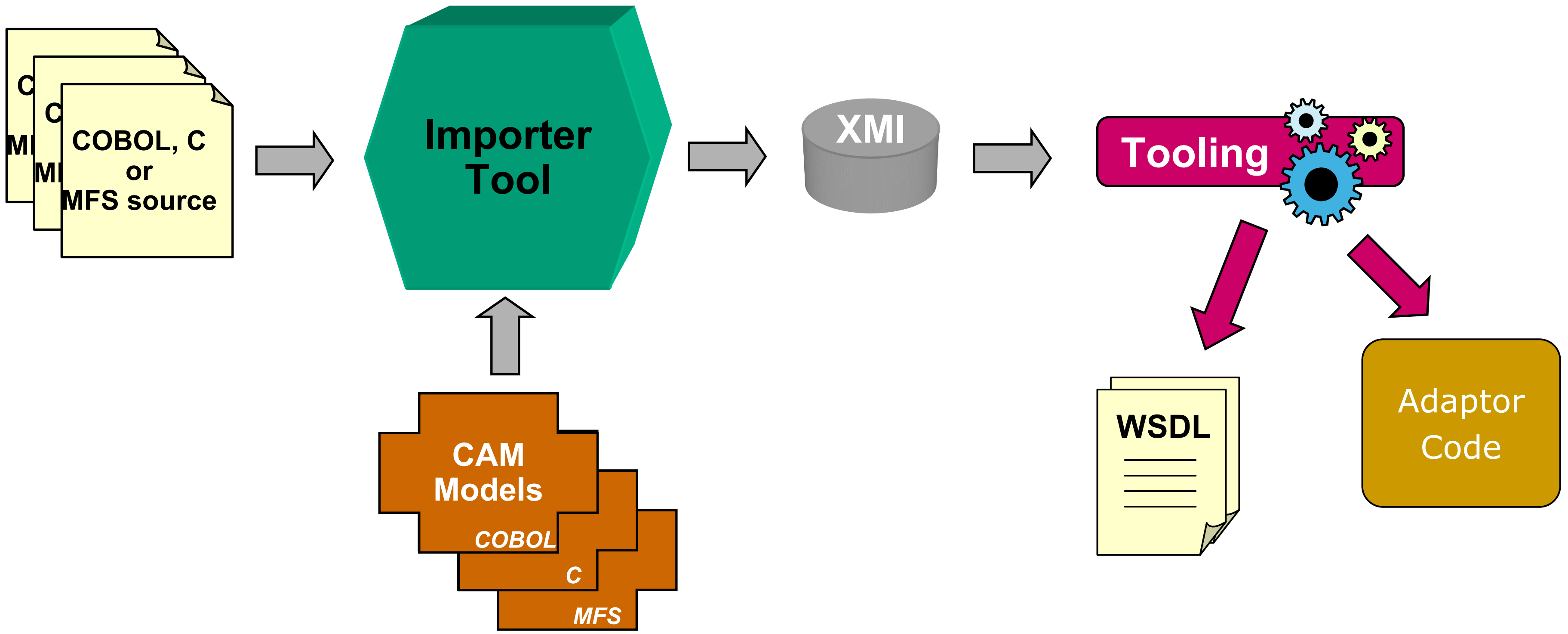
- CAM enables a data structure to be defined in a standardized, platform neutral and language neutral manner:
  ➔ definition is stored in **XML Metadata Interchange** (XMI) format

- A tool would be used to “import“ a data source and create an XMI description, based upon the CAM language model
  Today: support for COBOL, C and MFS data definitions in WebSphere tooling

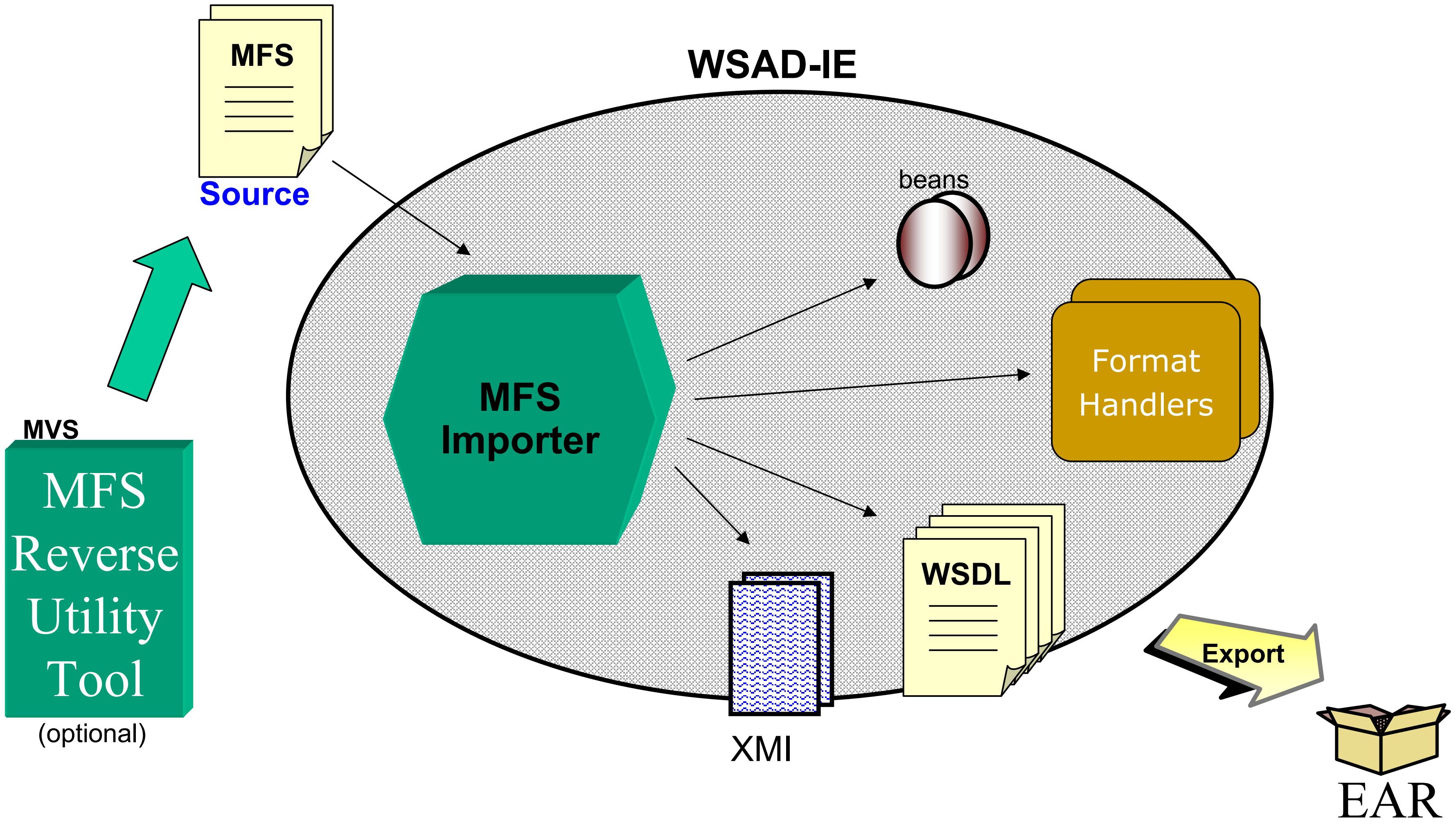
- The generated XMI data can then be used by tools to create:
  - corresponding XML Schema
  - WSDL
  - format handlers (adapters) for transforming data to a different format, e.g. transforming ...
  - ... a Java bean into an IMS input message

Common Application Metamodel (CAM)

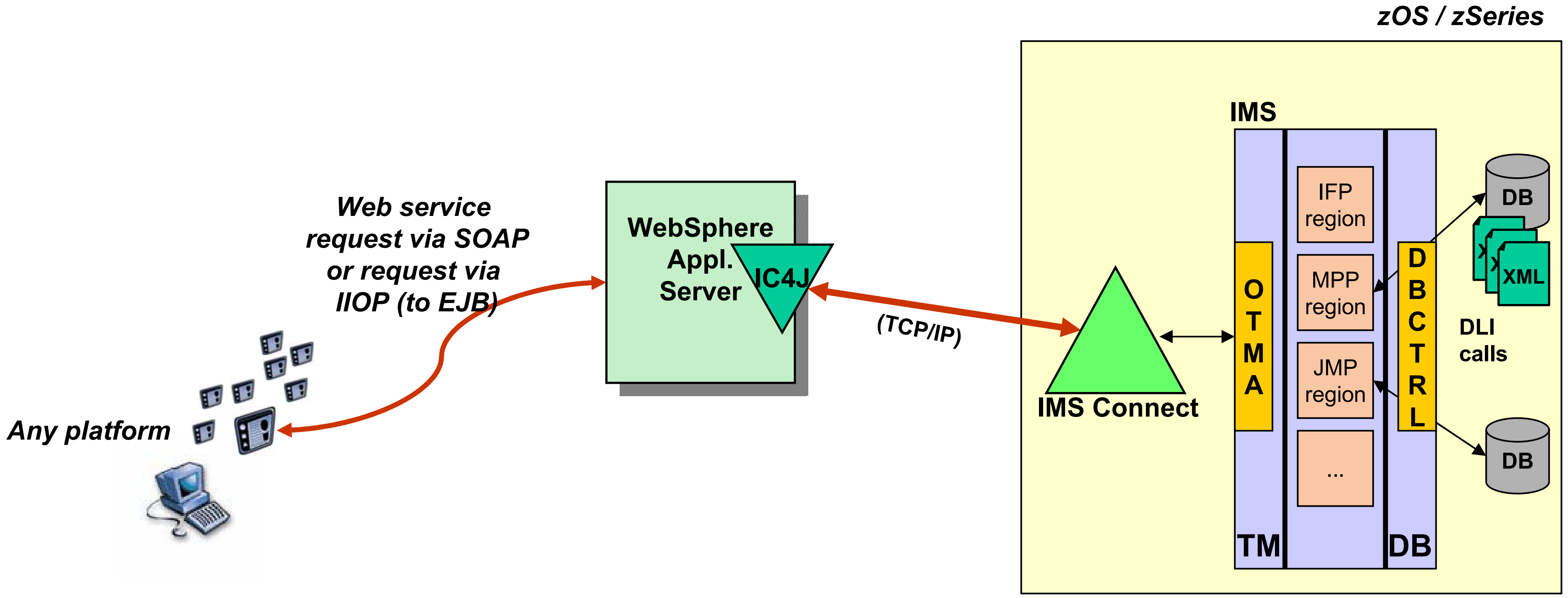
CAM is OMG marketplace standard for EAI



IMS MFS Web Services - Tooling Support



Addition of MFS Web Services

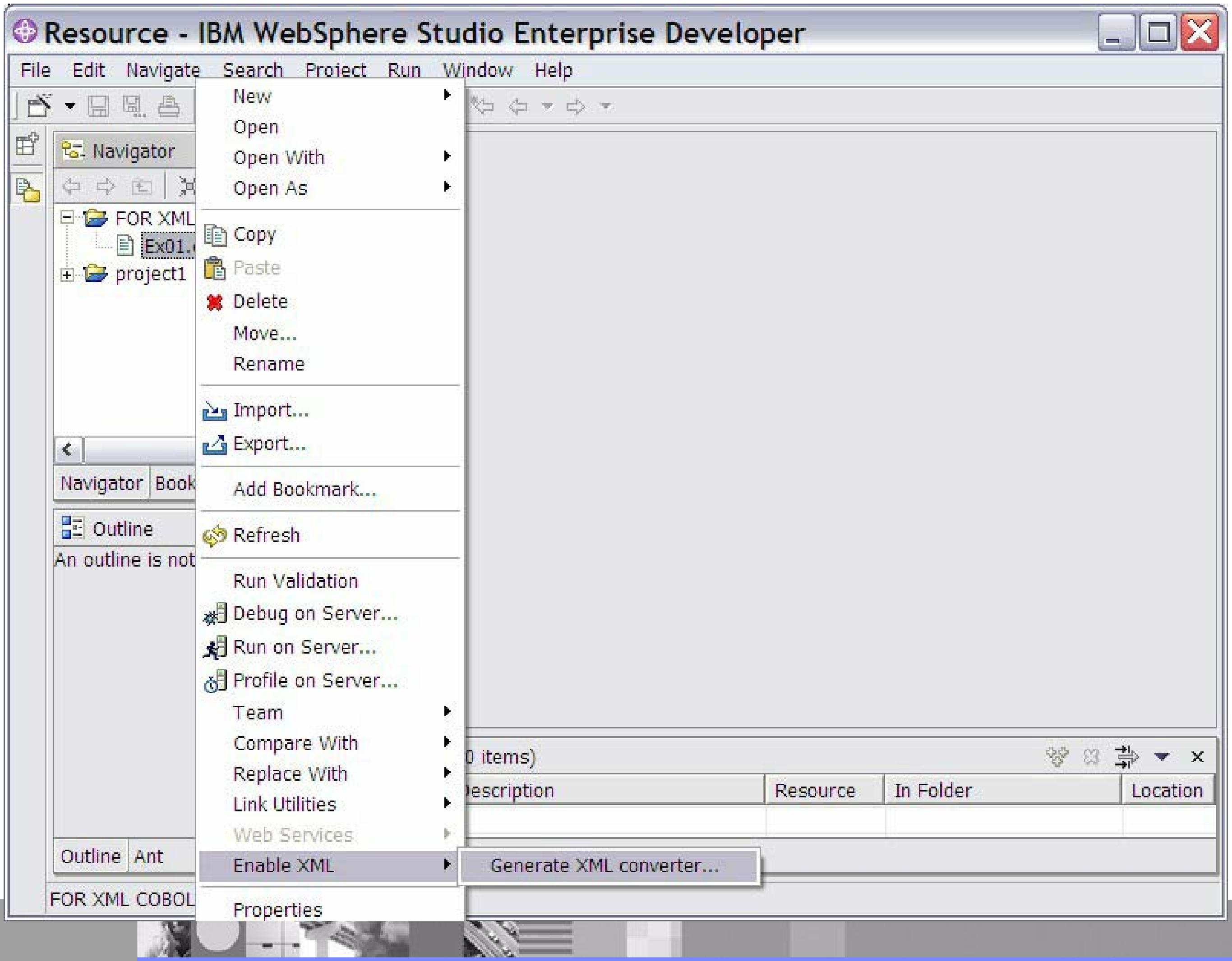


WebSphere Studio Enterprise Developer (WSED)

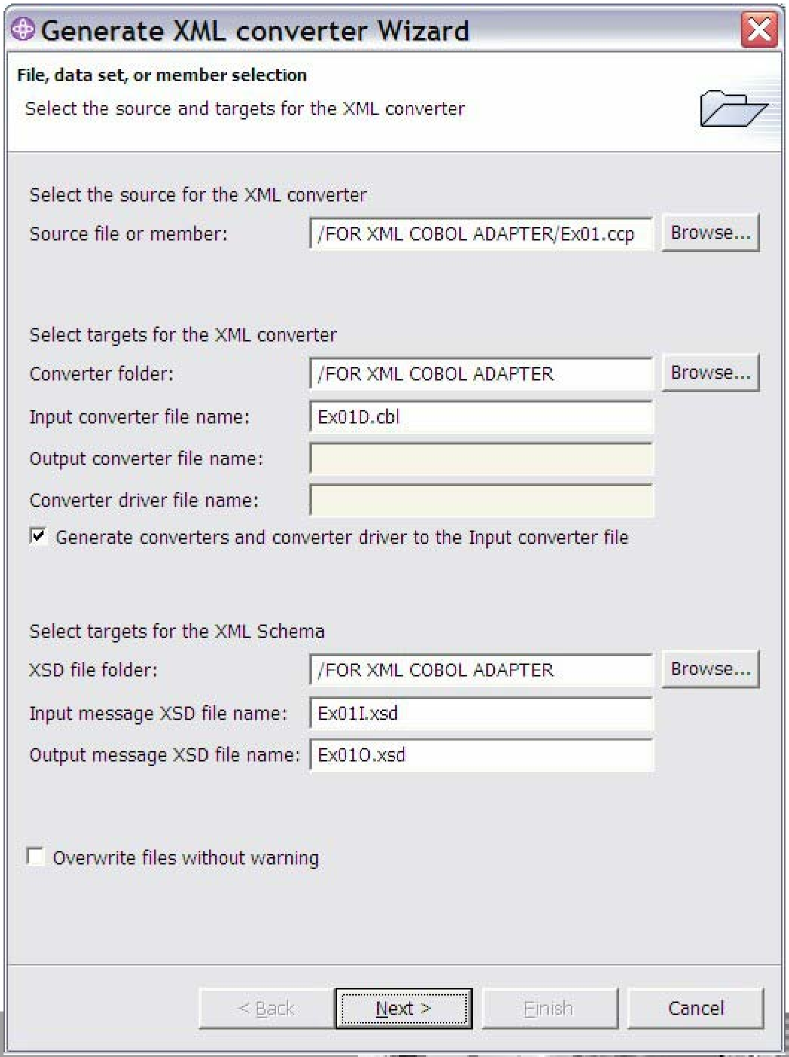
- **IDE for *zOS* traditional applications**
- **V 5.0+ includes XML converter tools supporting COBOL:**
  - **Inbound XML Converter:**
    - ✓ use XML PARSE verb of the Enterprise COBOL V3.1+ compiler to parse incoming XML messages
    - ✓ convert parsed messages to COBOL byte streams
  - **Outbound XML Converter:**
    - ✓ convert output COBOL byte streams to XML messages

Enable legacy IMS COBOL applications processing XML input/ output messages

Generate XML Converter using WSED 5.0+



Generate XML Converter using WSED 5.0+



Generate XML converter Wizard

File, data set, or member selection

Select the source and targets for the XML converter

Select the source for the XML converter

Source file or member:	/FOR XML COBOL ADAPTER/Ex01.ccp

Select targets for the XML converter

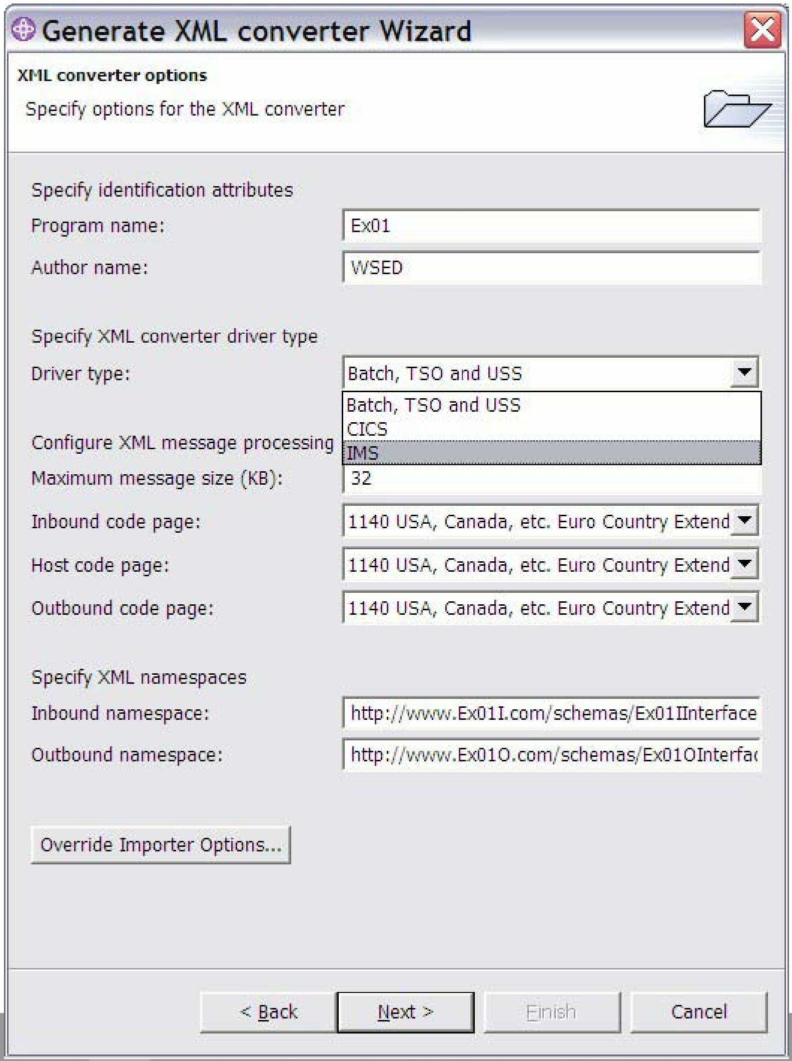
Converter folder:	/FOR XML COBOL ADAPTER
Input converter file name:	Ex01D.cbl
Output converter file name:	
Converter driver file name:	

[✓] Generate converters and converter driver to the Input converter file

Select targets for the XML Schema

XSD file folder:	/FOR XML COBOL ADAPTER
Input message XSD file name:	Ex01I.xsd
Output message XSD file name:	Ex01O.xsd

[ ] Overwrite files without warning



Generate XML converter Wizard

XML converter options

Specify options for the XML converter

Specify identification attributes

Program name:	Ex01
Author name:	WSED

Specify XML converter driver type

Driver type:	Batch, TSO and USS

Batch, TSO and USS
CICS
IMS

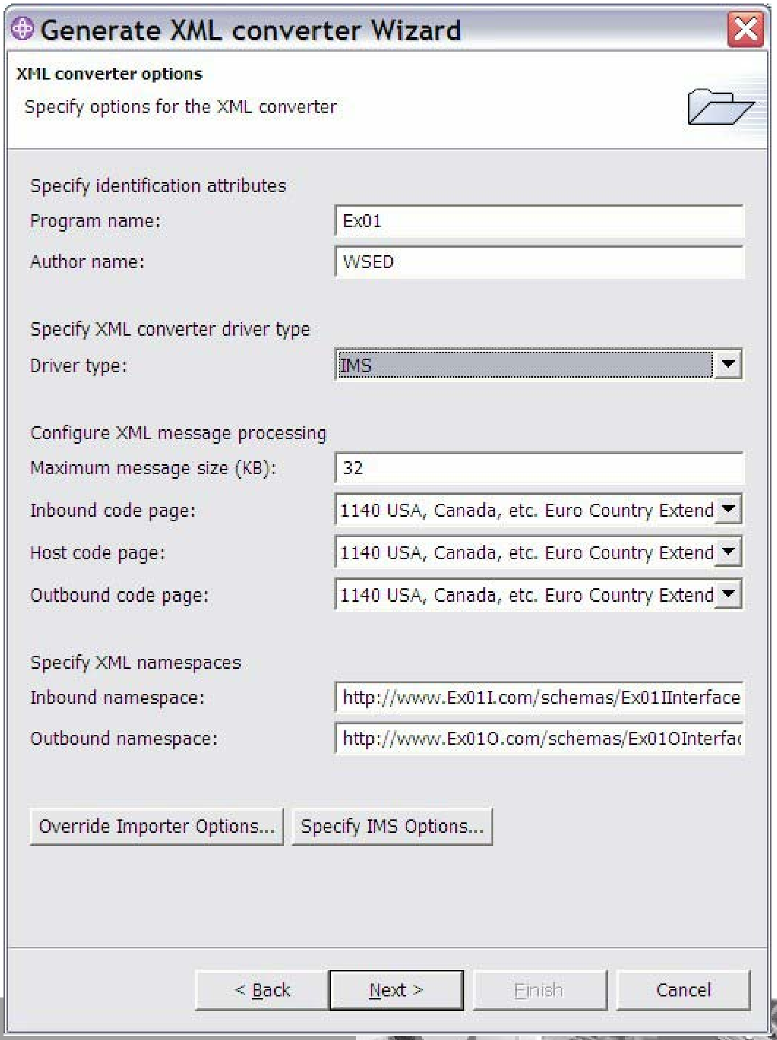
Configure XML message processing

Maximum message size (KB):	32
Inbound code page:	1140 USA, Canada, etc. Euro Country Extend
Host code page:	1140 USA, Canada, etc. Euro Country Extend
Outbound code page:	1140 USA, Canada, etc. Euro Country Extend

Specify XML namespaces

Inbound namespace:	http://www.Ex01I.com/schemas/Ex01IInterface
Outbound namespace:	http://www.Ex01O.com/schemas/Ex01OInterfac

Generate XML Converter using WSED 5.0+



Generate XML converter Wizard

XML converter options

Specify options for the XML converter

Specify identification attributes

Program name:	Ex01
Author name:	WSED

Specify XML converter driver type

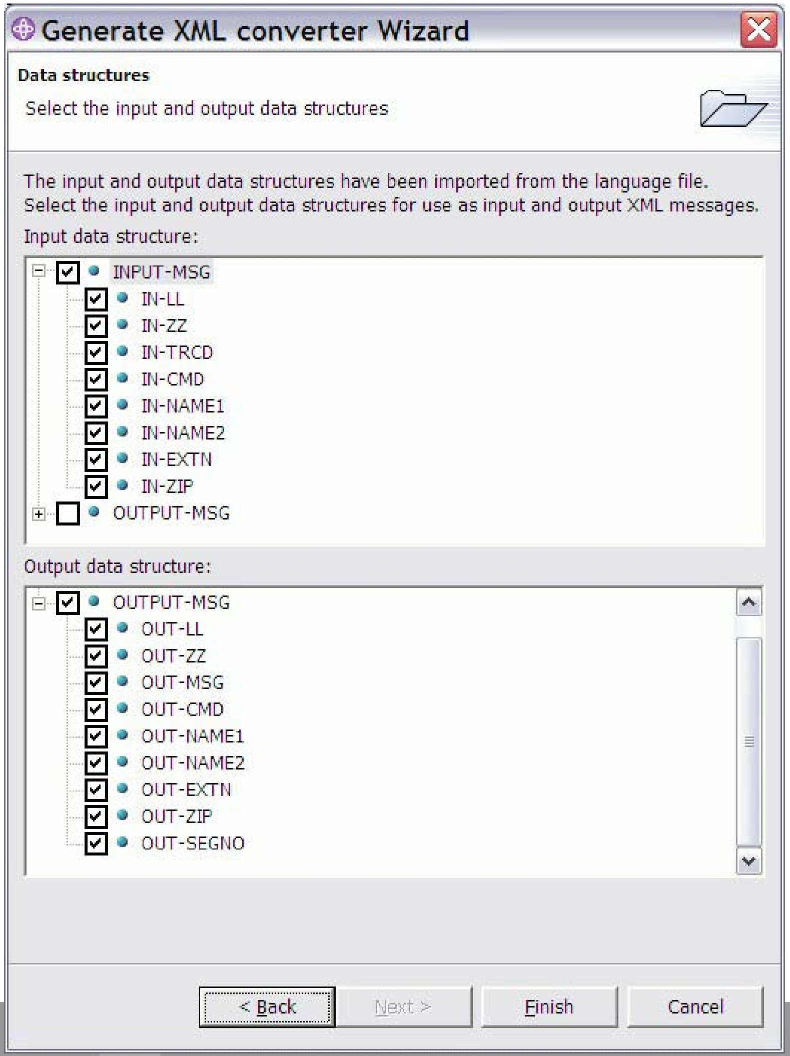
Driver type:	IMS

Configure XML message processing

Maximum message size (KB):	32
Inbound code page:	1140 USA, Canada, etc. Euro Country Extend
Host code page:	1140 USA, Canada, etc. Euro Country Extend
Outbound code page:	1140 USA, Canada, etc. Euro Country Extend

Specify XML namespaces

Inbound namespace:	http://www.Ex01I.com/schemas/Ex01IInterface
Outbound namespace:	http://www.Ex01O.com/schemas/Ex01OInterfac



Generate XML converter Wizard

Data structures

Select the input and output data structures

The input and output data structures have been imported from the language file.
Select the input and output data structures for use as input and output XML messages.

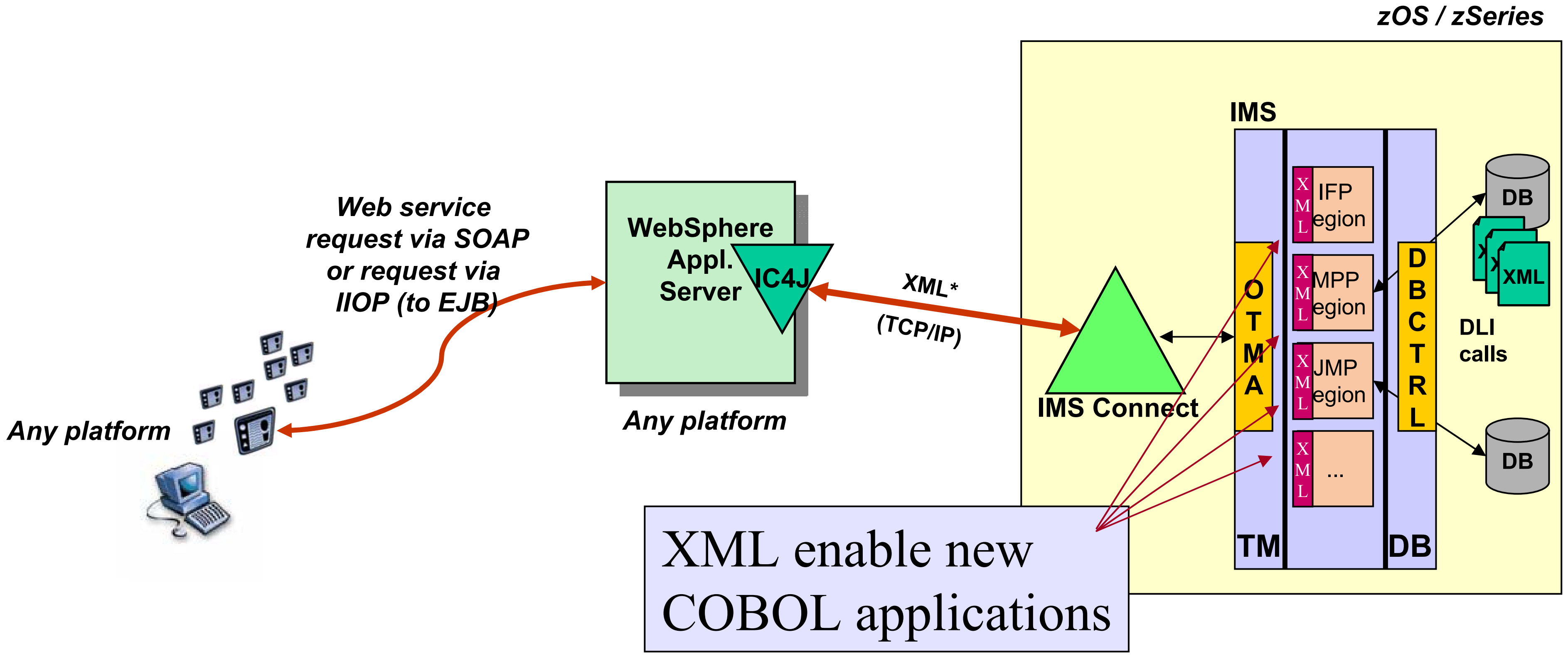
Input data structure:

- ☑ INPUT-MSG
  - ☑ IN-LL
  - ☑ IN-ZZ
  - ☑ IN-TRCD
  - ☑ IN-CMD
  - ☑ IN-NAME1
  - ☑ IN-NAME2
  - ☑ IN-EXTN
  - ☑ IN-ZIP
- ☐ OUTPUT-MSG

Output data structure:

- ☑ OUTPUT-MSG
  - ☑ OUT-LL
  - ☑ OUT-ZZ
  - ☑ OUT-MSG
  - ☑ OUT-CMD
  - ☑ OUT-NAME1
  - ☑ OUT-NAME2
  - ☑ OUT-EXTN
  - ☑ OUT-ZIP
  - ☑ OUT-SEGNO

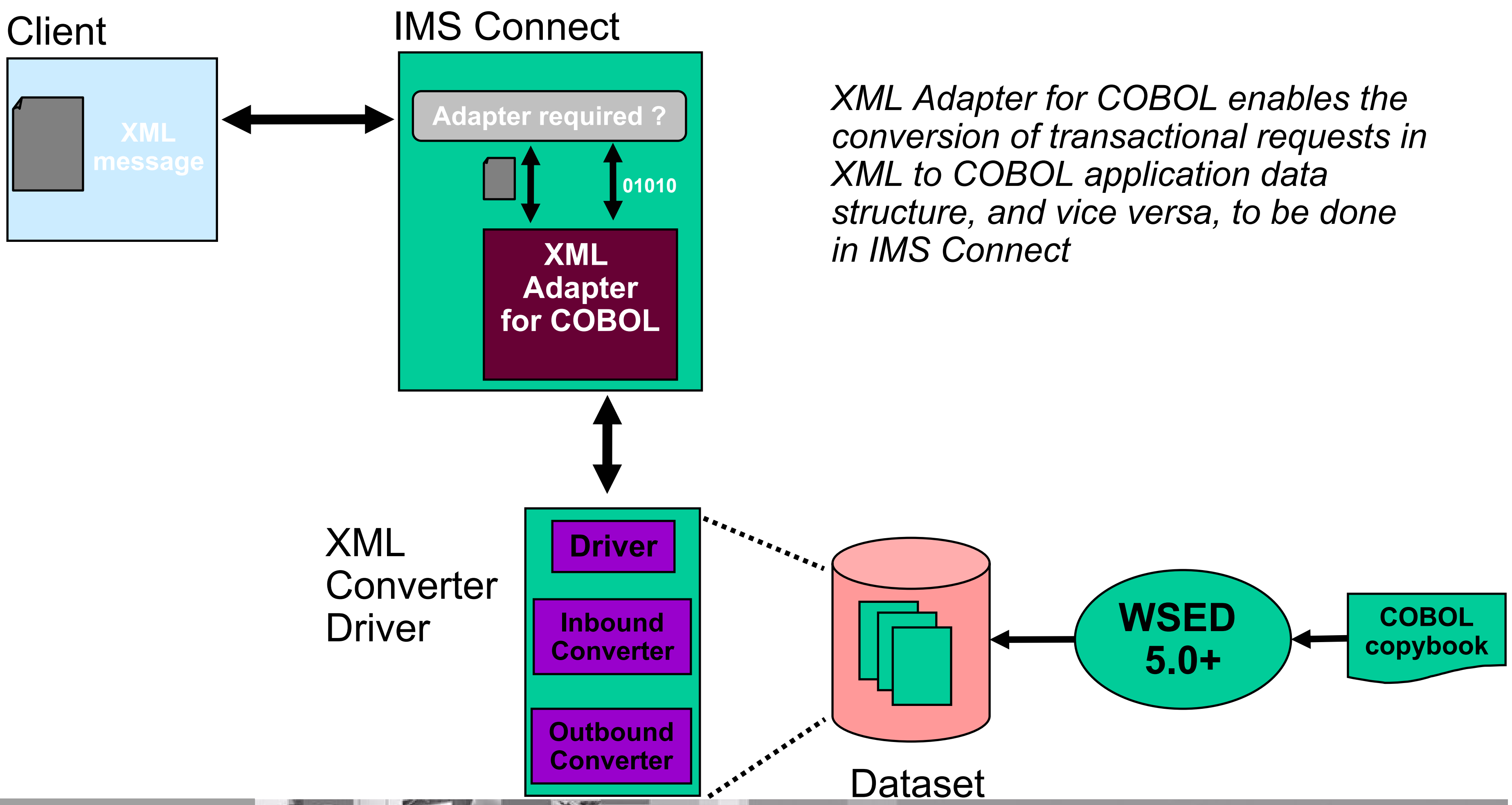
New XML Applications through XML COBOL Adaptors



XML enable new COBOL applications

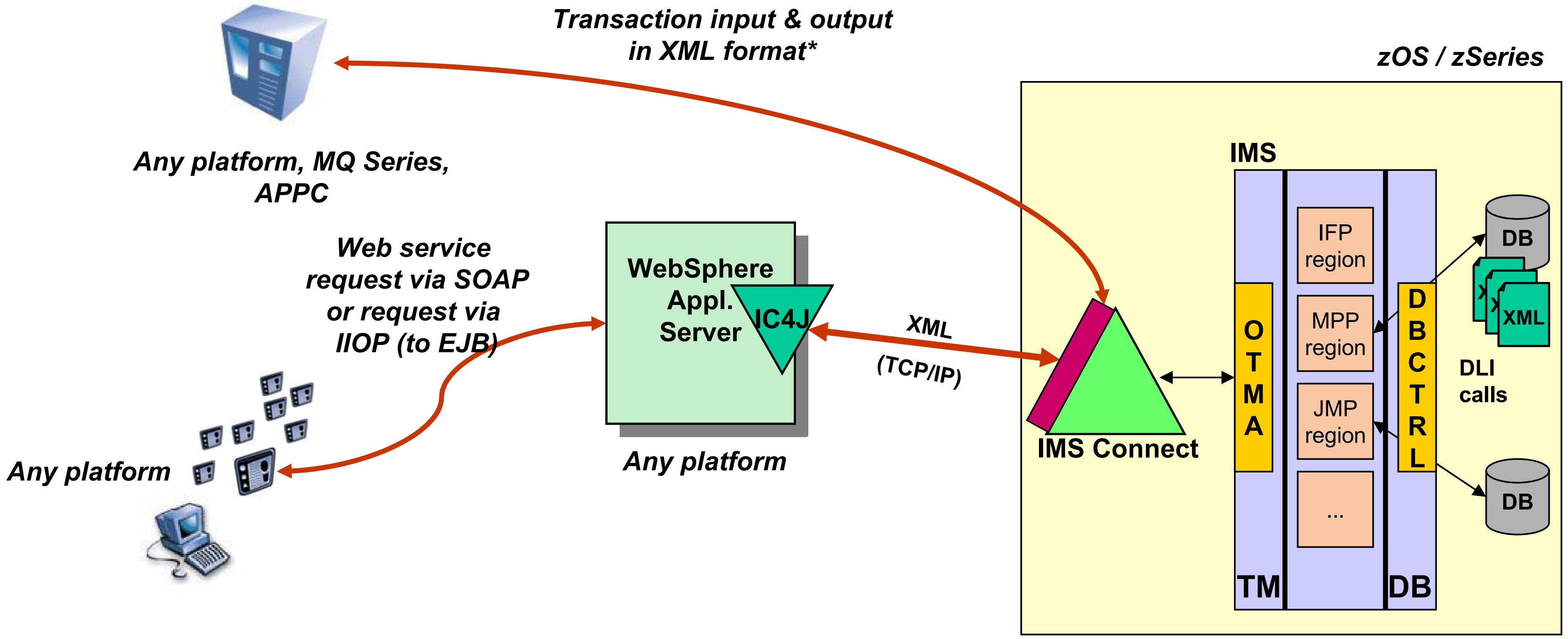
*COBOL or Java apps.

Requirement: XML adapter for COBOL in IMS Connect



XML Adapter for COBOL enables the conversion of transactional requests in XML to COBOL application data structure, and vice versa, to be done in IMS Connect

New and Existing XML Applications through XML COBOL Adaptors



*COBOL or Java apps.

IMS SOAP Gateway

So far ...

*… IMS provides services for IMS applications using **(a)** IMS OTMA **(b)** IMS Connect, **(c)** IMS Connector for Java, and **(d)** WebSphere Application Server:*

- ✓ *transform existing IMS transactions into services* by using e.g. WSADIE, WSED, RAD to create service definitions for IMS transactions
- ✓ deploy these service definitions to WAS to make the IMS services available as an *Enterprise Java Bean (EJB) services* or *Simple Object Access Protocol (SOAP) web services*

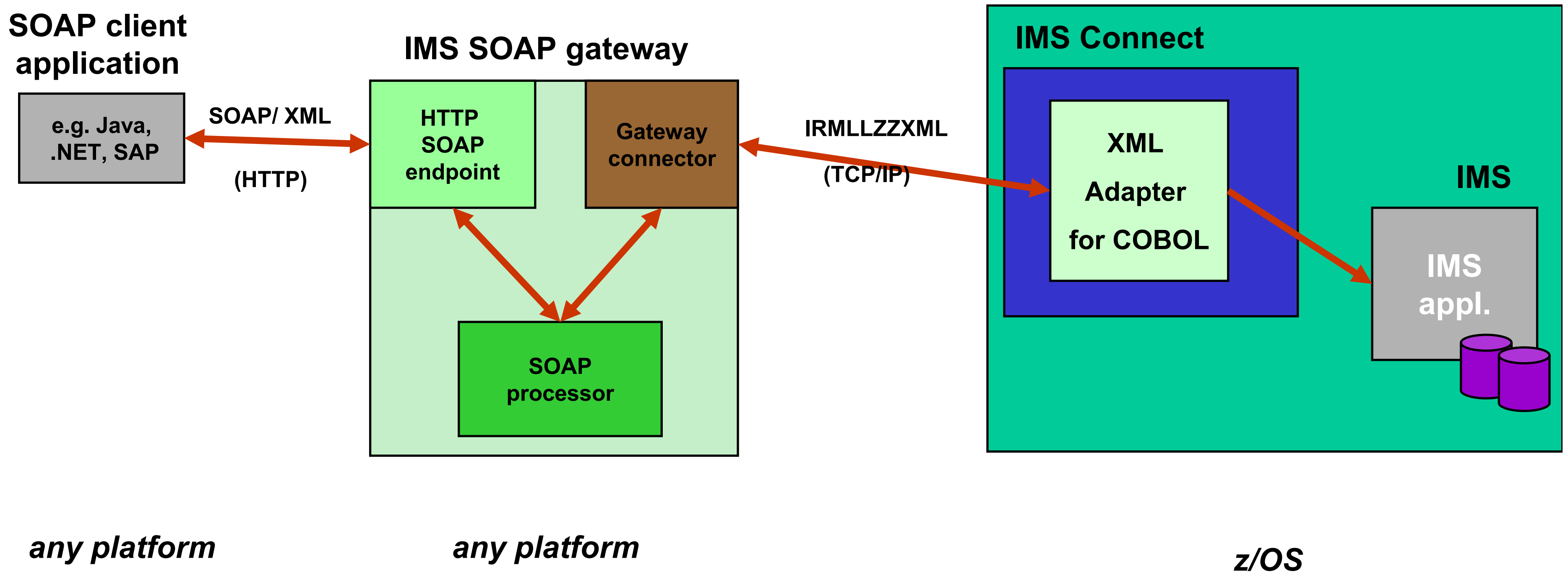
***Requirement:* The IMS SOAP Gateway …**

… is an XML based connectivity solution that enables existing/ new IMS applications to:

- using SOAP to provide and request services

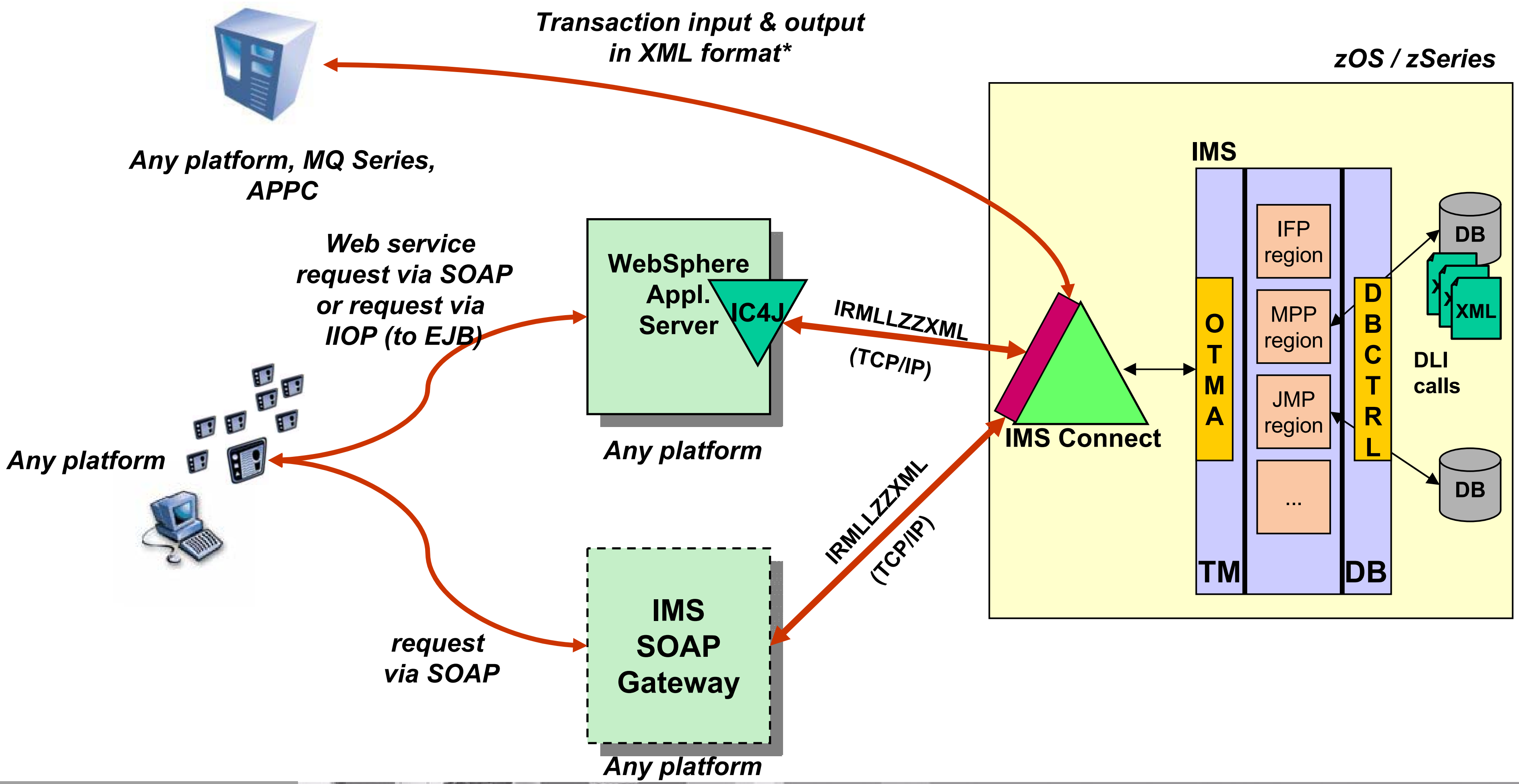
- independently of platform, environment, application language, or programming model

Requirement: IMS SOAP Gateway



http://www.ibm.com/ims (*then look for SOAP for IMS*)

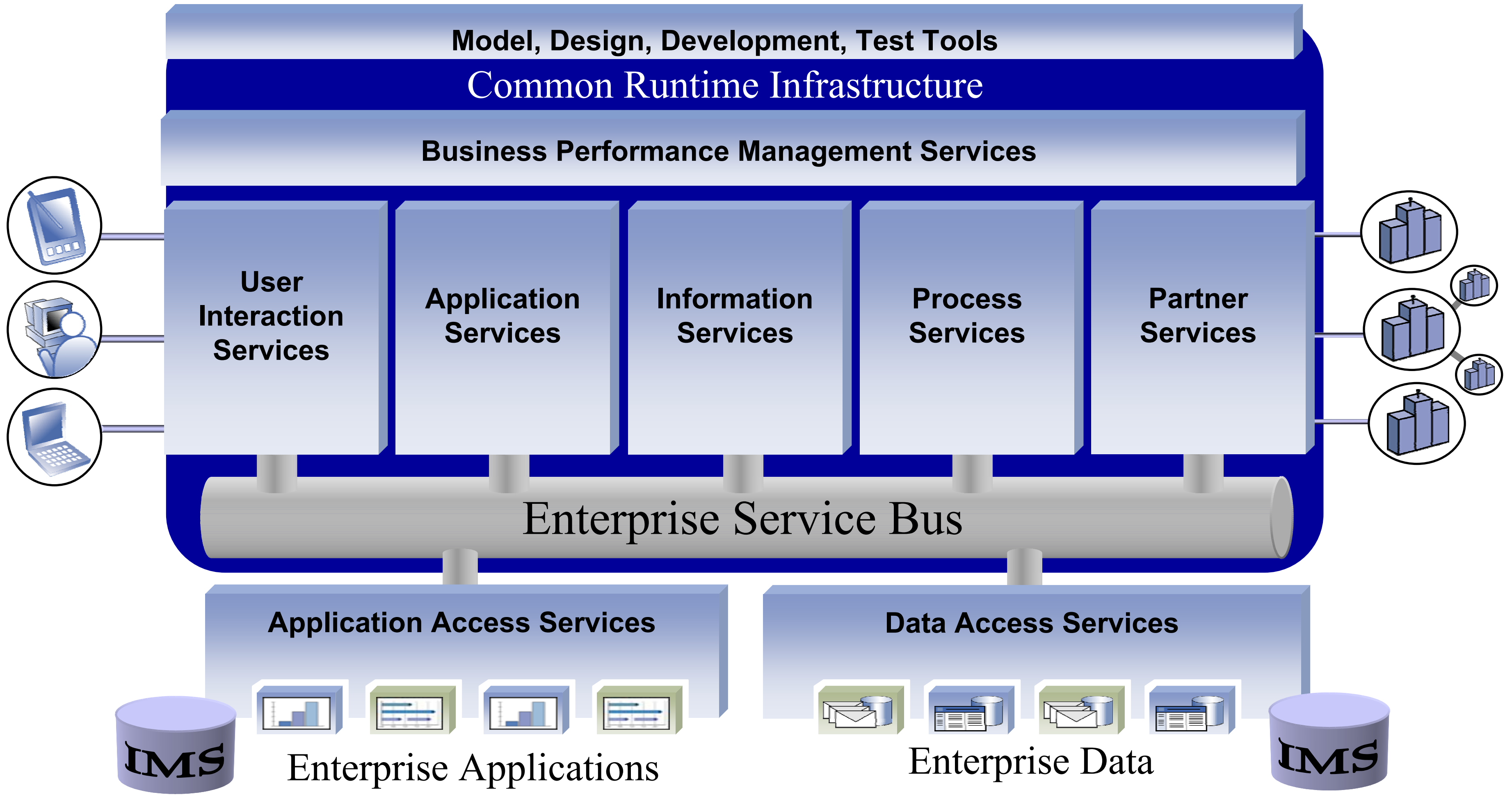
New and Existing XML Applications through XML COBOL Adaptors



*COBOL or Java apps.

IBM Service Oriented Architecture

ESB Is a Defining Element


Model, Design, Development, Test Tools
Common Runtime Infrastructure
Business Performance Management Services
User Interaction Services
Application Services
Information Services
Process Services
Partner Services
Enterprise Service Bus
Application Access Services
Data Access Services
IMS
Enterprise Applications
Enterprise Data
IMS