IBM Confidential

WebSphere software

IBM

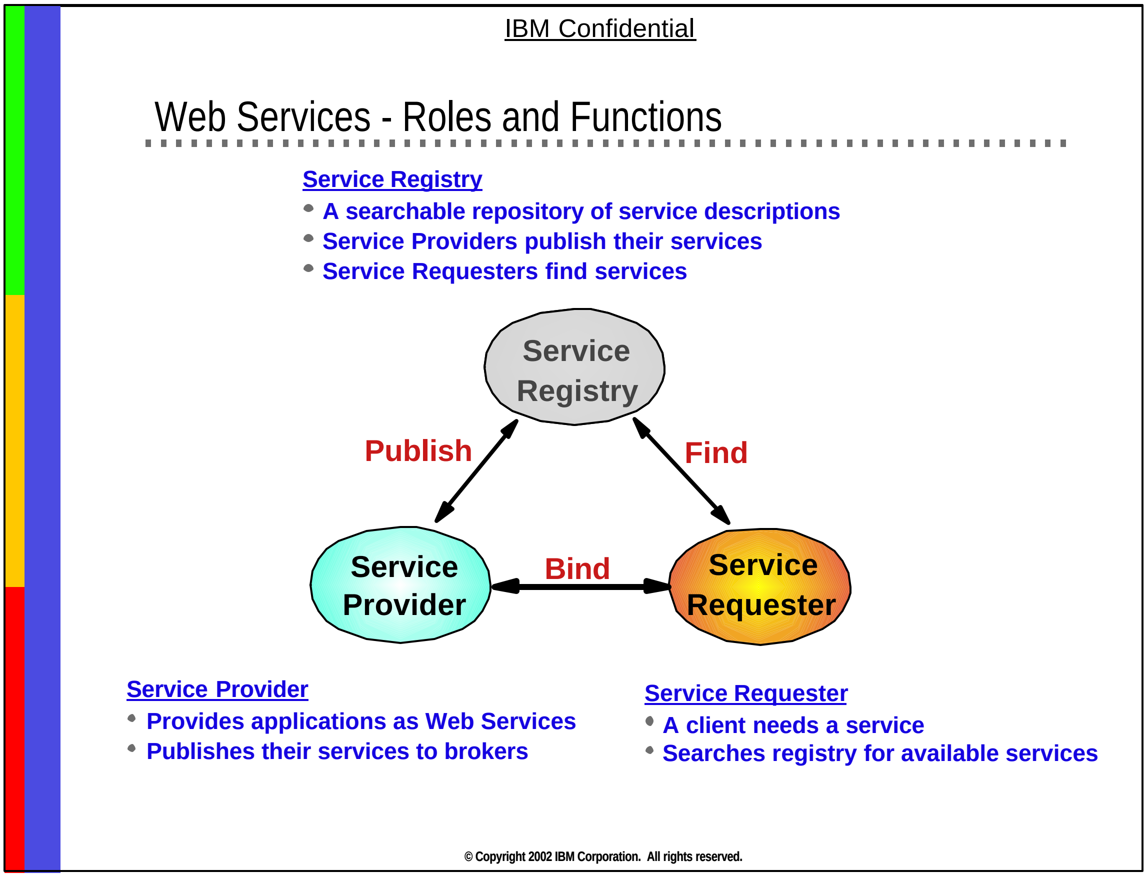
business software

Web Services Review

IBM Software Group

Objectives

- Web Services Overview
- Brief look of Service-Oriented Architectures
- Interface, Service, and Binding
- WSDL and WSIF
- The Enterprise Service Bus



IBM Confidential

Web Services - Roles and Functions

Service Registry
- **A searchable repository of service descriptions**
- **Service Providers publish their services**
- **Service Requesters find services**

Service Provider
- **Provides applications as Web Services**
- **Publishes their services to brokers**

Service Requester
- **A client needs a service**
- **Searches registry for available services**

- The functional enhancements and the commitment to open standards that characterize the WebSphere V5.0 release go well beyond mere support of J2EE 1.3. In fact, IBM believes that the market of information technologies is mature enough to move on to the next step in terms of distributed, Internet-enabled applications. That's where Web Services play a key role.
- The Service Registry role owns a directory of all of the services available. The Service Registry represents a new, potentially very lucrative business model. The Registry owner could charge a fee for the use of their directory. Service Providers lists (or advertise) the Service offering in the registry. Service Requesters on the other hand query the Service Registry about the services available. Once Service Registry provides the binding information to the requestor, it is no longer involved in the communications between the provider and requestor.
- The Service Provider has developed services that they make available as Web Service. These services will be hosted on their Application Server. A service is invoked by a requestor through an XML message. These XML messages are generally carried across the Internet through a network-neutral standard protocol called Simple Access Object Protocol (SOAP). The Service Provider describes the service they are making available with a standard encoding called Web Services Description Language (WSDL).
- The Service Requester is the business that requires a certain business function to be fulfilled. From an architectural perspective, this is the application that is looking for (queries the Service Registry) and then bnids to and invokes the service. The requester has to find the service before invoking it - this process of discovery involves accessing a directory where the services are published. The access occurs through a set of standard APIs defined in the UDDI standard. UDDI stands for Universal Description, Discovery and Integration specification.

Web Services - Overview

- Web Services are all about interfaces
  - Only Interface information exposed to client (service requester)
    - Interfaces information separate from implementation
    - Underlying implementation details hidden from client
  - Existing resources and logic reused

- Interface decriptions based on Industry Open-Standards
  - Offers compatibility and acceptance
  - Loose coupling between service provider and service requester possible as interface details are "public"

- Interfaces focus on sending and receiving messages
  - Send a message to perform an operation; Receive message with result
  - Messages based on Industry standards

- Web Services based on Service-Oriented Architecture (SOA)
  - SOAP (Simple Object Access Protocol) not only way of calling Web Services

Service Oriented Architectures (SOA)

IBM Software Group

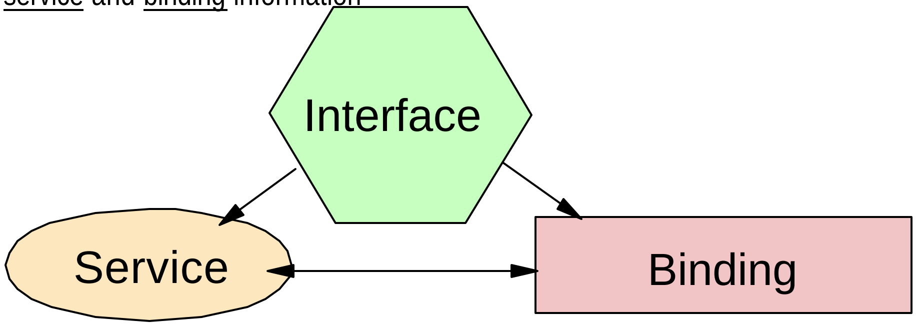
Services in Applications

- "From the beginning..." applications have been designed for process, component and code reuse with business process applications
  - Procedural (COBOL)
  - Object-Oriented (C++)
- Limited amount of reuse
  - Only within large enterprise applications
  - Other applications and languages incompatible
  - Business process lost in trying to plan for reuse
- Solution: Design components as services
  - Each component completes a certain piece of the business process
  - Reuse comes at the service
- Different problems surfaced
  - Services where tightly coupled
  - Calls were complex and inconsistent between different languages
  - Calls remained "local"
  - Hard to change when the business process changed or the underlying technology

The Next Step: Service-Oriented Achitecture

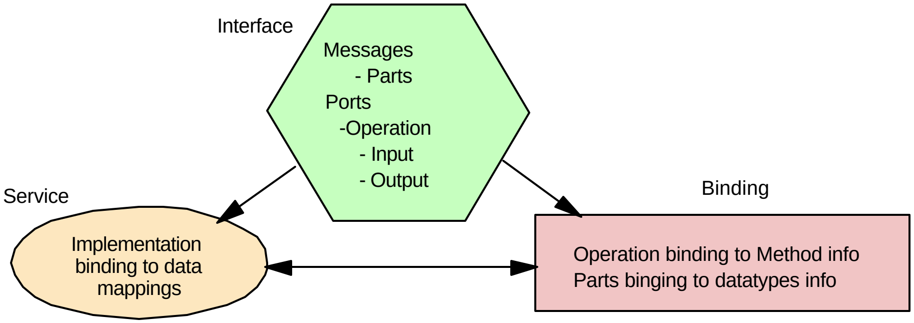
- A Service-Oriented Achitecture (SOA) provides a standard way in which services can be **described** and **called**
  - Allow for different technologies to easily interact as services
  - Support remote service calls in a distributed application
  - Change implementation (services) easily and quickly as the business process changes
    - Client programming model remains the same

- The Service-Oriented Achitecture describe the <u>interface</u> separate from the <u>service</u> and <u>binding</u> information



Definitions

- **Interface** describes the generic service
  - Different ports
  - Operations (methods) which are available on a particular port
  - Messages (parameter list) for the different operations
  - Parts to the Messages (individual parameters)
- **Binding** describes the binding of service to specific implementation
  - Binds Operations to actual methods
  - Binds Parts to actual datatypes
- **Service** describes binding of communication information for calling service
  - Binds service name to implementation specifics


Interface
Messages
- Parts
Ports
-Operation
- Input
- Output
Service
Implementation binding to data mappings
Binding
Operation binding to Method info
Parts binging to datatypes info

IBM Confidential

WebSphere software

IBM

e business software

Web Service Description Language (WSDL)

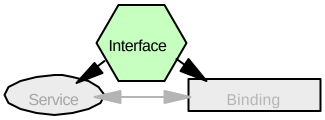
IBM Software Group

Web Services Description Language (WSDL)

- To describes services for calling, the definition must be readable and compatible with different technologies to ensure true services
- Web Services Description Language (WSDL) provides an industry standard way of **describing** services in Service Oriented Architectures (we will talk about calling later...)
- Formatted in XML allowing for easy reading
- Designed to be flexible and extendable for describing different services without making the service definition incomprehensible

WSDL - Inteface information

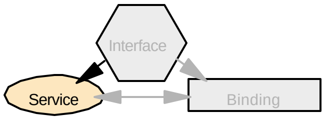
- Service information developer programs against



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="StockQuote"
    targetNamespace="http://example.com.wsdl/stockquote/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://example.com.wsdl/stockquote/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <message name="getQuoteRequest">
        <part name="ticker" type="xsd:string"/>
    </message>
    <message name="getQuoteResponse">
        <part name="result" type="xsd:float"/>
    </message>
    <portType name="StockQuote">
        <operation name="getQuote" parameterOrder="ticker">
            <input message="tns:getQuoteRequest" name="getQuoteRequest"/>
            <output message="tns:getQuoteResponse" name="getQuoteResponse"/>
        </operation>
    </portType>
</definitions>
```

WSDL - Service information

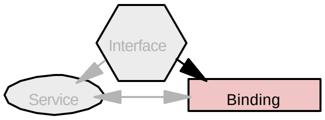
- Describes location of Service



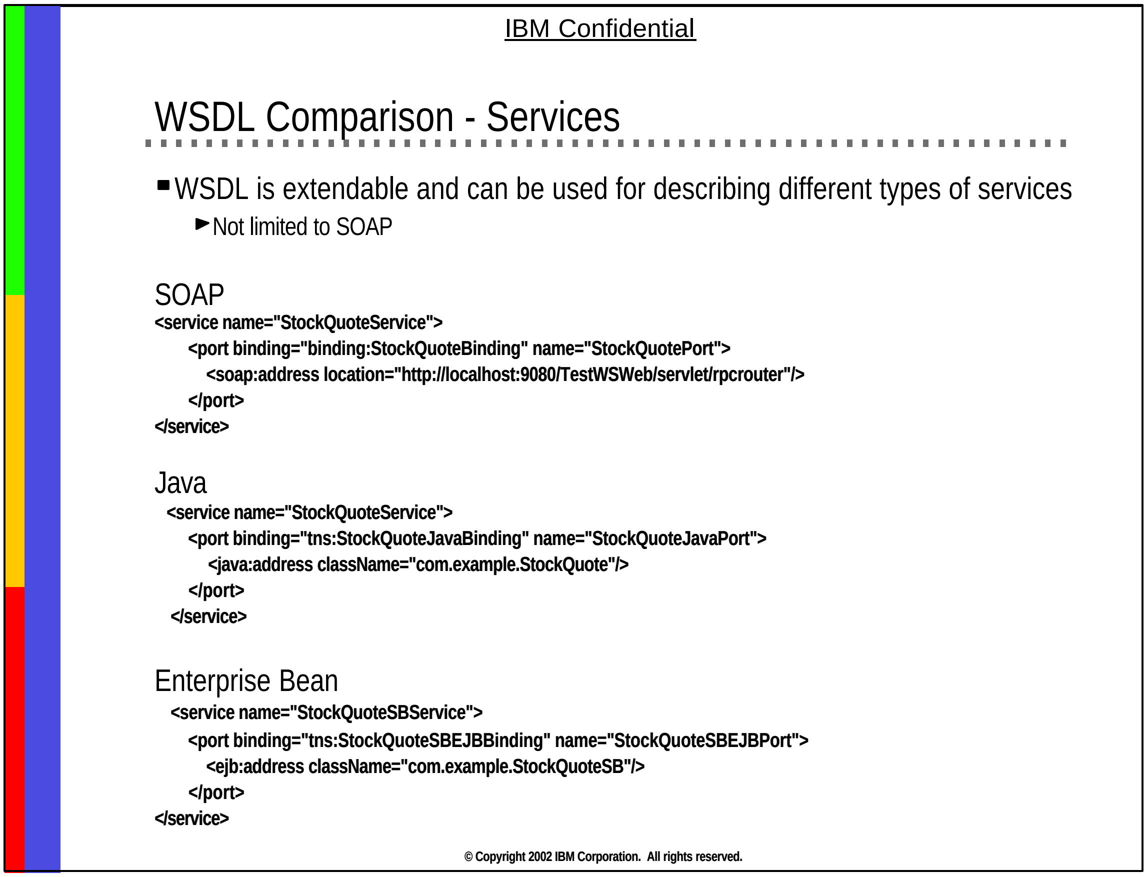
```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="StockQuoteService"
            targetNamespace="http://example.com.wsdl/stockquoteservice/"
            xmlns:tns="http://example.com.wsdl/stockquoteservice/"
            xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
            xmlns:binding="http://example.com.wsdl/stockquotebinding/"
            xmlns="http://schemas.xmlsoap.org/wsdl/">
  <import namespace="http://example.com.wsdl/stockquotebinding/"
        location="http://localhost:9080/TestWSWeb/wsdl/com/example/StockQuoteBinding.wsdl"/>
  <service name="StockQuoteService">
    <port name="StockQuotePort" binding="binding:StockQuoteBinding">
      <soap:address location="http://localhost:9080/TestWSWeb/servlet/rpcrouter"/>
    </port>
  </service>
</definitions>
```

WSDL - Binding information

- Binds generic method and data service information to actual implementation



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions ......>
   <import location="StockQuote.wsdl" namespace="http://example.com.wsdl/stockquote/"/>
   <binding name="StockQuoteBinding" type="interface:StockQuote">
      <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="getQuote">
         <soap:operation soapAction="" style="rpc"/>
         <input name="getQuoteRequest">
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="http://tempuri.org/com.example.StockQuote"
               parts="ticker" use="encoded"/>
         </input>
         <output name="getQuoteResponse">
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="http://tempuri.org/com.example.StockQuote" use="encoded"/>
         </output>
      </operation>
   </binding>
</definitions>
```



WSDL Comparison - Services

- WSDL is extendable and can be used for describing different types of services
  - Not limited to SOAP

SOAP

```
<service name="StockQuoteService">
      <port binding="binding:StockQuoteBinding" name="StockQuotePort">
         <soap:address location="http://localhost:9080/TestWSWeb/servlet/rpcrouter"/>
      </port>
</service>
```

Java

```
  <service name="StockQuoteService">
      <port binding="tns:StockQuoteJavaBinding" name="StockQuoteJavaPort">
         <java:address className="com.example.StockQuote"/>
      </port>
   </service>
```

Enterprise Bean

```
   <service name="StockQuoteSBService">
      <port binding="tns:StockQuoteSBEJBBinding" name="StockQuoteSBEJBPort">
         <ejb:address className="com.example.StockQuoteSB"/>
      </port>
</service>
```

WSDL Comparison - Bindings

- Binding information specific to implementation

Java

```
<binding name="StockQuoteJavaBinding" type="interface1:StockQuote">
      <java:binding/>
       <format:typeMapping encoding="Java " style="Java">
          <format:typeMap formatType="float" typeName="xsd:float"/>
          <format:typeMap formatType="java.lang.String" typeName="xsd:string"/>
      </format:typeMapping>
      <operation name="getQuote">
         <java:operation methodName="getQuote" parameterOrder="ticker" returnPart="result"/>
         <input name="getQuoteRequest"/>
          <output name="getQuoteResponse"/>
      </operation>
   </binding>
```

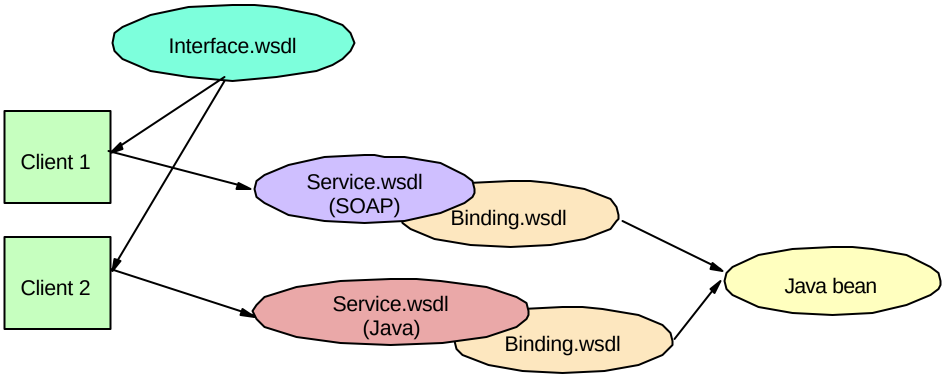
Enterprise Bean

```
<binding name="StockQuoteSBEJBBinding" type="interface1:StockQuoteSB">
      <ejb:binding/>
       <format:typeMapping encoding="EJB" style="Java">
          <format:typeMap formatType="float" typeName="xsd:float"/>
          <format:typeMap formatType="java.lang.String" typeName="xsd:string"/>
      </format:typeMapping>
      <operation name="getQuote">
         <ejb:operation methodName="getQuote" parameterOrder="ticker" returnPart="result"/>
         <input name="getQuoteRequest"/>
          <output name="getQuoteResponse"/>
      </operation>
   </binding>
```

WSDL in Summary

- WSDL provides a standard way of **defining** services

- WSDL leads to a consistent client programming model
  - Proxy used by client access specific service
  - Offers option of specific interface for a specific service and situation



IBM Confidential

WebSphere software

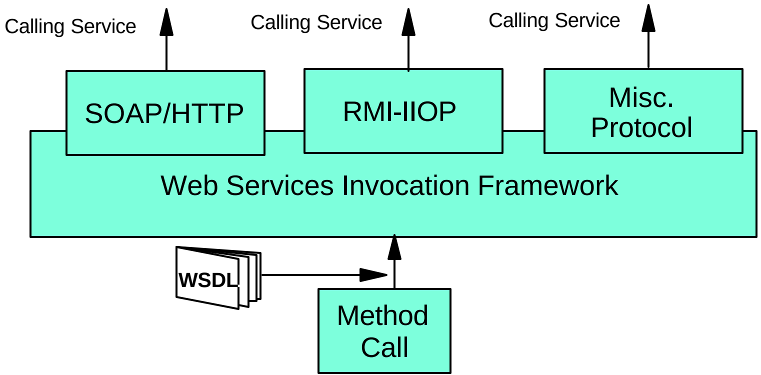
business software

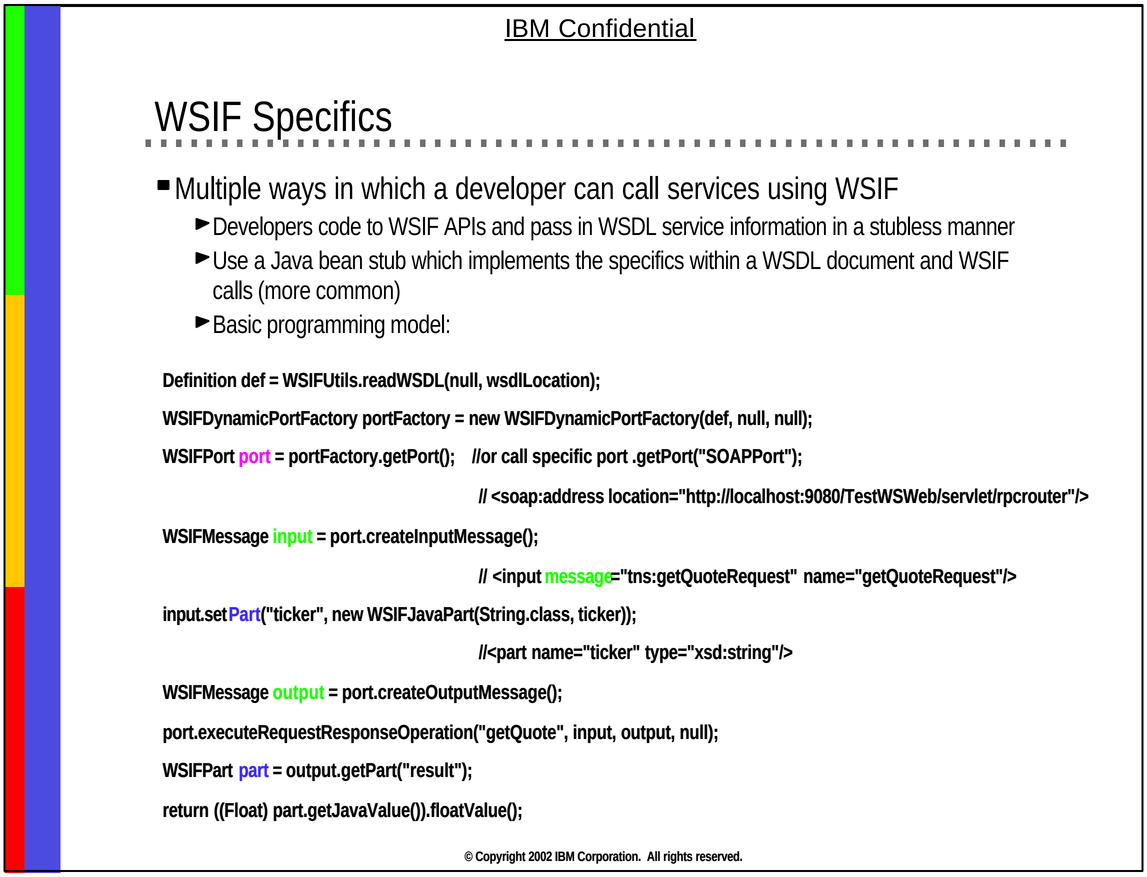
Web Service Invocation Framework (WSIF)

IBM Software Group

Calling Services

- Web Services Invocation Framework (WSIF) used to **call** the WSDL described services
- Framework based on APIs which are not tied to a specific protocol
- WSIF allows for protocols to be plugged in and supported for different services
- IBM technology soon to be an Apache open-source project





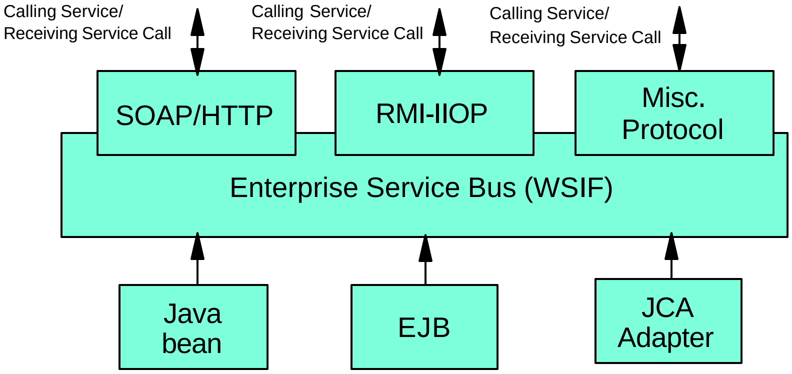
WSIF Specifics

- Multiple ways in which a developer can call services using WSIF
  - Developers code to WSIF APIs and pass in WSDL service information in a stubless manner
  - Use a Java bean stub which implements the specifics within a WSDL document and WSIF calls (more common)
  - Basic programming model:

```
Definition def = WSIFUtils.readWSDL(null, wsdlLocation);
WSIFDynamicPortFactory portFactory = new WSIFDynamicPortFactory(def, null, null);
WSIFPort port = portFactory.getPort();    //or call specific port .getPort("SOAPPort");
                                          // <soap:address location="http://localhost:9080/TestWSWeb/servlet/rpcrouter"/>
WSIFMessage input = port.createInputMessage();
                                          // <input message="tns:getQuoteRequest"  name="getQuoteRequest"/>
input.setPart("ticker", new WSIFJavaPart(String.class, ticker));
                                          //<part name="ticker" type="xsd:string"/>
WSIFMessage output = port.createOutputMessage();
port.executeRequestResponseOperation("getQuote", input, output, null);
WSIFPart part = output.getPart("result");
return ((Float) part.getJavaValue()).floatValue();
```

WSIF leads to the Enterprise Service Bus

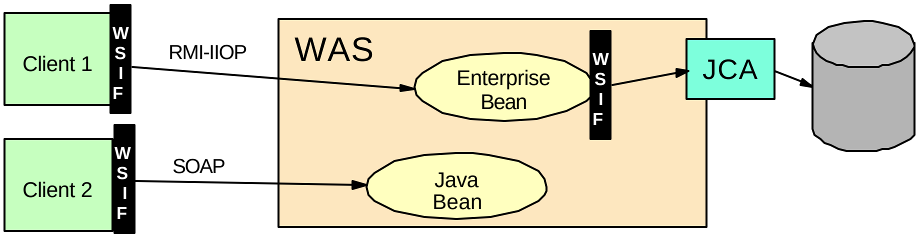
- WSIF can be used for calling services **and** receiving service calls
- Many Java artifacts and components can become services
- Integration point or Enterprise Service Bus emerges where services can call and be called
- Individual Services can be combined (process) and executed to complete business processes


Calling Service/ Receiving Service Call
Calling Service/ Receiving Service Call
Calling Service/ Receiving Service Call
SOAP/HTTP
RMI-IIOP
Misc. Protocol
Enterprise Service Bus (WSIF)
Java bean
EJB
JCA Adapter

WSIF in Summary

- Removes requirement of programming to service and binding APIs

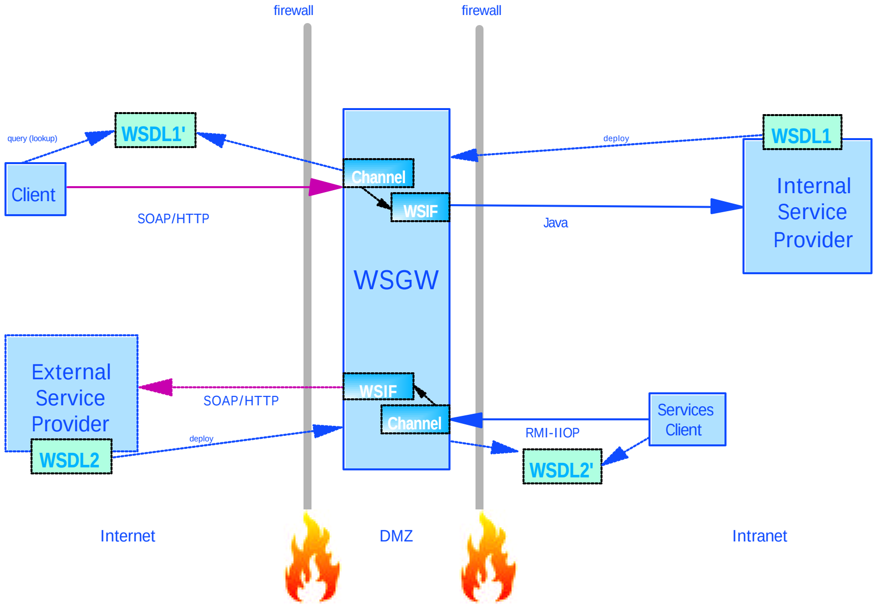
- Services can be clients to other services



Web Services Gateway - Overview

- Puts WSIF in the "middle"
  - Service Provider can offer single service for multiple Service Consumers
  - Service Consumer can program directly to protocol APIs
  - WSIF is independent of Service Provider and Service Consumer
- Web Services Gateway takes existing service information (service.wsdl) and exposes it:
  - from a controlled (secure) location
  - in a format compatible with the network (Protocol Transformation)
- Offers new intergration opportunites

Web Services Gateway - Big Picture



Summary

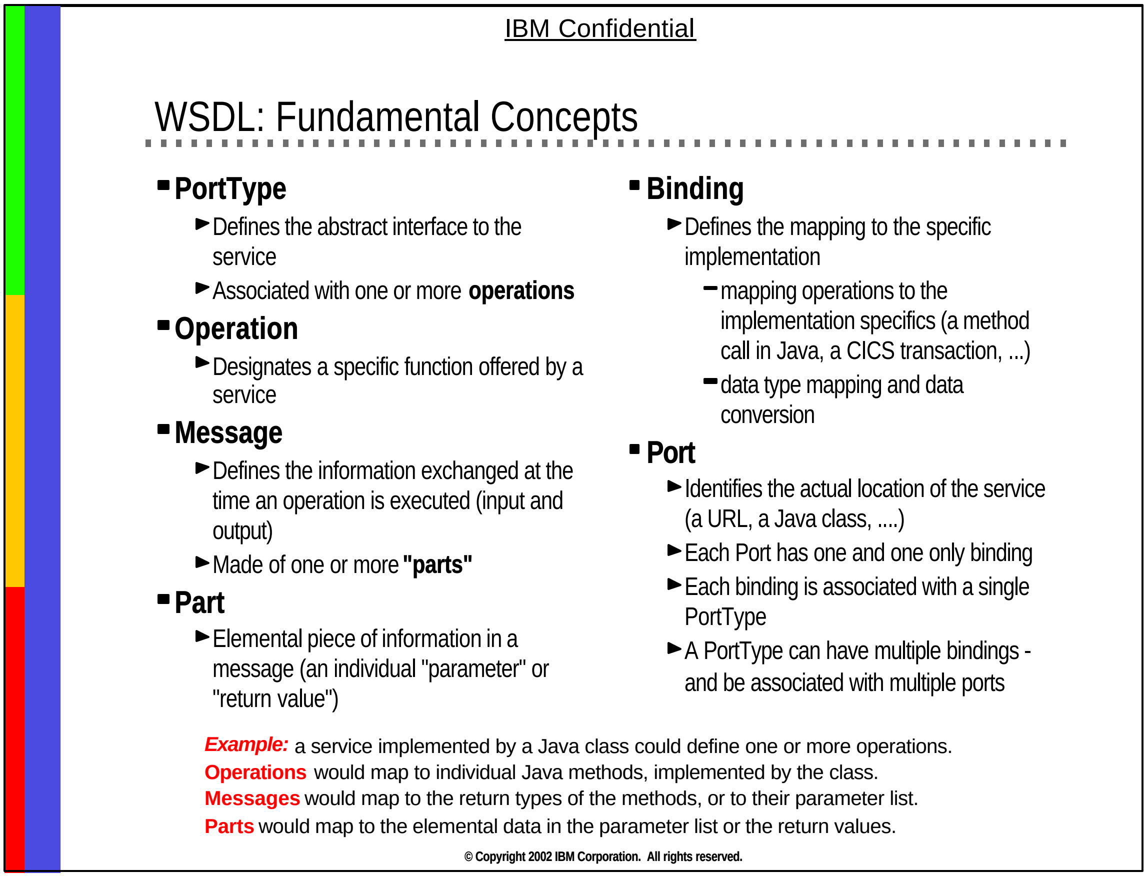
- Web Services are Interfaces
- Services-Oriented Architecture based on open standards allows for true Services independent of implementation and protocol requirements
- Services now can be reused based on what they contribute to the overall business process
- Services can be combined to complete business process
- Services can be easily changed as the business process changes, implementations change, protocols change, etc.

WebSphere software

e business software

Appendix

IBM Software Group



WSDL: Fundamental Concepts

- **PortType**
  - Defines the abstract interface to the service
  - Associated with one or more **operations**
- **Operation**
  - Designates a specific function offered by a service
- **Message**
  - Defines the information exchanged at the time an operation is executed (input and output)
  - Made of one or more **"parts"**
- **Part**
  - Elemental piece of information in a message (an individual "parameter" or "return value")
- **Binding**
  - Defines the mapping to the specific implementation
    - mapping operations to the implementation specifics (a method call in Java, a CICS transaction, ...)
    - data type mapping and data conversion
- **Port**
  - Identifies the actual location of the service (a URL, a Java class, ...)
  - Each Port has one and one only binding
  - Each binding is associated with a single PortType
  - A PortType can have multiple bindings - and be associated with multiple ports

Example: a service implemented by a Java class could define one or more operations.
Operations would map to individual Java methods, implemented by the class.
Messages would map to the return types of the methods, or to their parameter list.
Parts would map to the elemental data in the parameter list or the return values.

- This chart introduces some fundamental WSDL terminology.
- PortType is the abstract designation of the service and it is independent of the implementation of the service itself.
- A PortType is associated with one or more operations. Operations represents the individual functions provided by the service itself. Operations typically need input data and return result data. Input and output data is encapsulated by "messages".
- A message can be made of one or more parts, or elemental pieces of information. Think for example of a service that allows you to get a quote for the best fare for an airline itinerary. The service could be implemented by a Java class. It may expose an operation called getQuote that takes an input data structure that contains the departure city, the destination, and the date of travel. It may return an output data structure containing the airline code, the flight number, and the fare. The input and output data structures are the "messages" involved in the execution of the operation. Each piece of information (departure city, destination, date, fare, airline code, and flight number) are the "parts".
- The binding information specifies some details that allow mapping the abstract information encapsulated in the PortType to the specific implementation. Data mapping and translation, and mapping the operations to the implementation functions are among the issues that the binding information clarifies.
- The port identifies the "location" of the service and it is associated with a specific binding. For example, if a service is implemented by a Java bean, the Port information may point to the the java class that implements the functions. Or, if the service is to be accessed via SOAP, the port contains the urn of the SOAP service.
- Every port is associate with a single binding and the binding in turn is associated to a single PortType. But the same port type can be associated with multiple bindings. For instance, you may want to define two different bindings to your "best airline fare" search service - one binding for the clients that access the service "locally" (intranet) and a binding for Internet clients that are going to use SOAP to get to the service.