Services and Contracts

Dominic Duggan
Stevens Institute of Technology

1

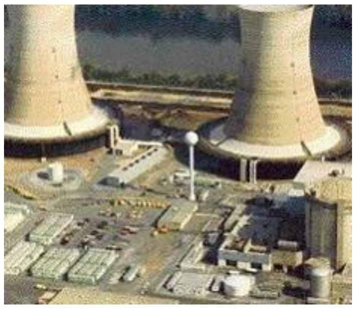
SERVICE-ORIENTED ARCHITECTURE

2

Software as a Service

Ex: SalesForce.com

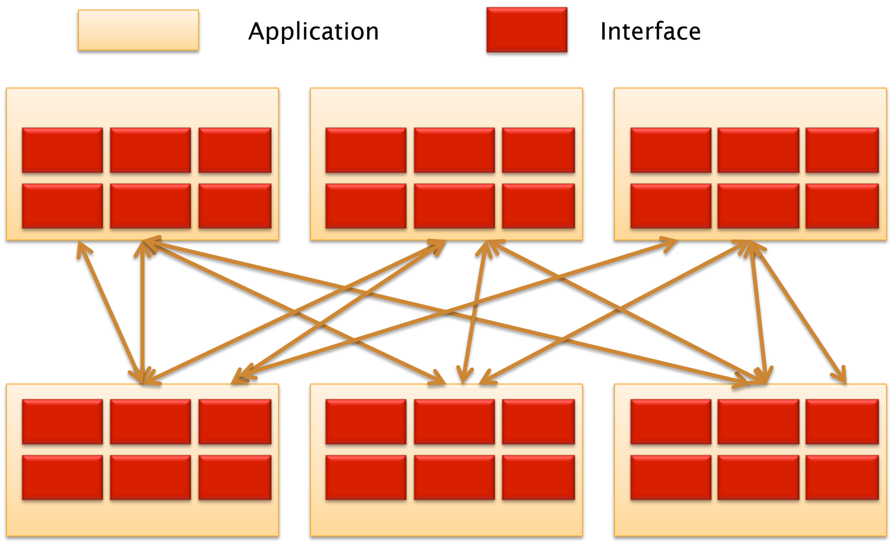
Traditional Software | **On-Demand Utility**



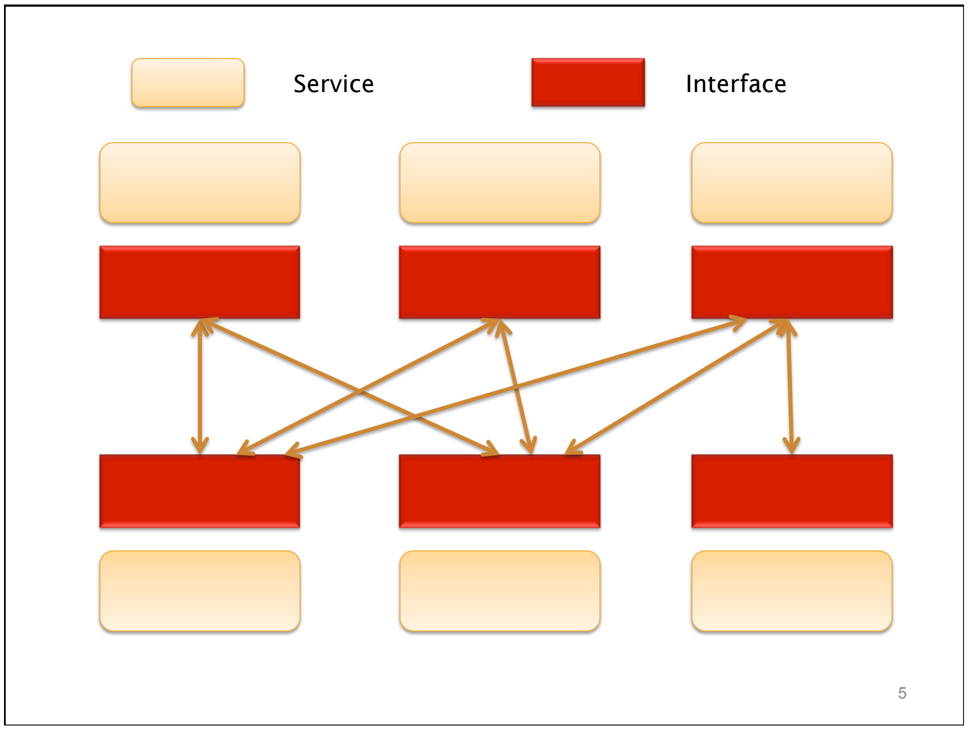
Build Your Own

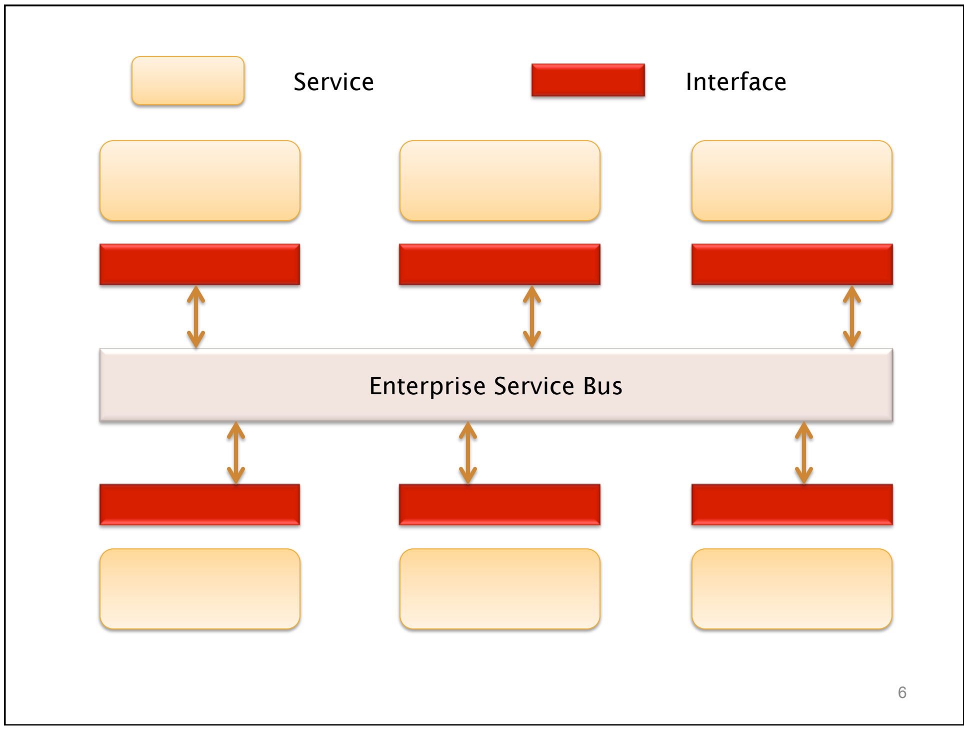
Plug In, Subscribe
Pay-per-Use

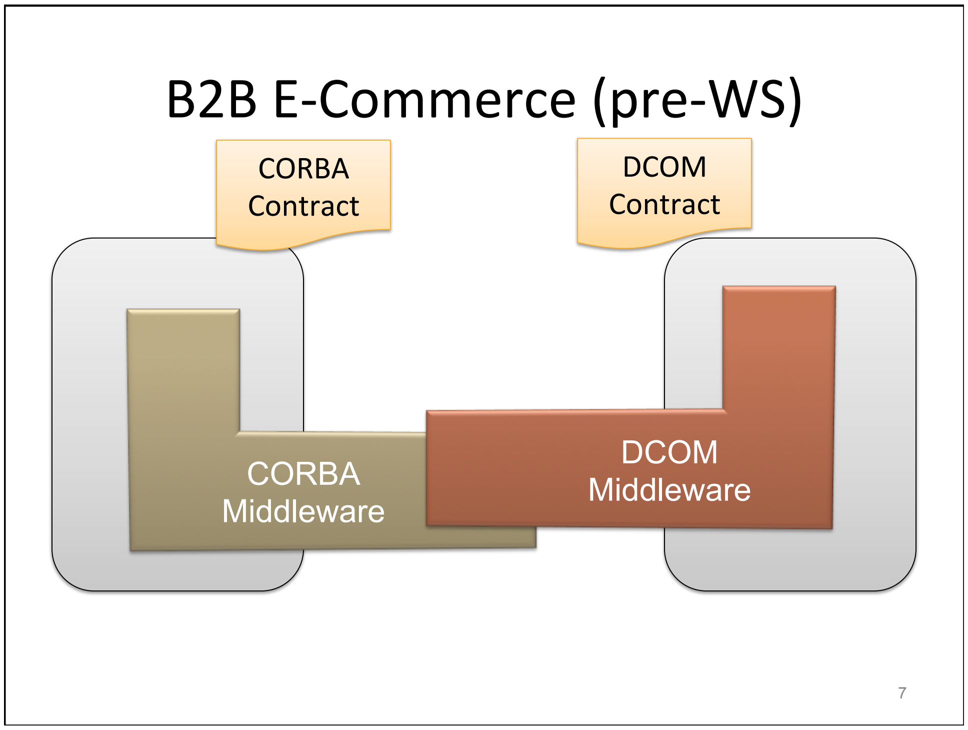
3



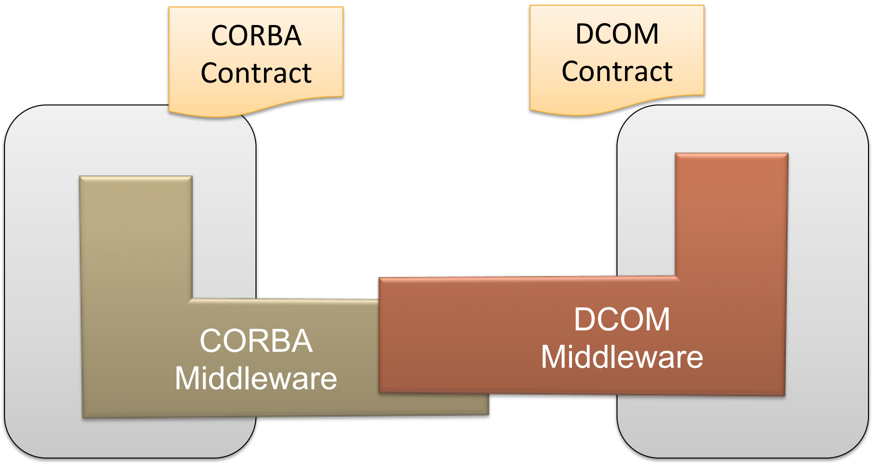
4


Service
Interface
5

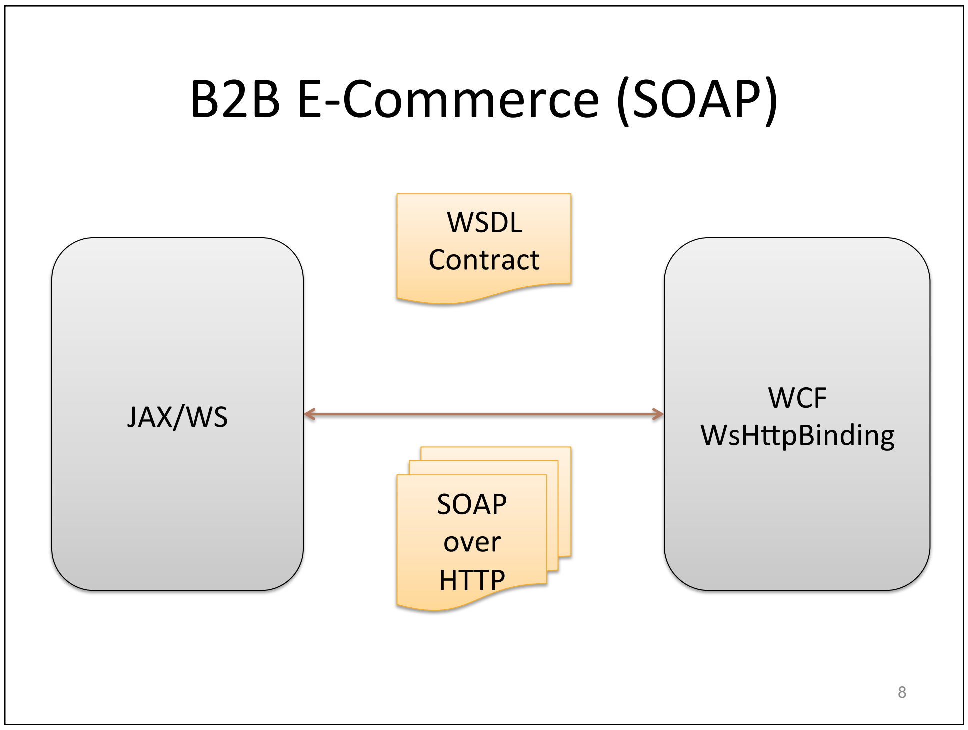

Service
Interface
Enterprise Service Bus
6



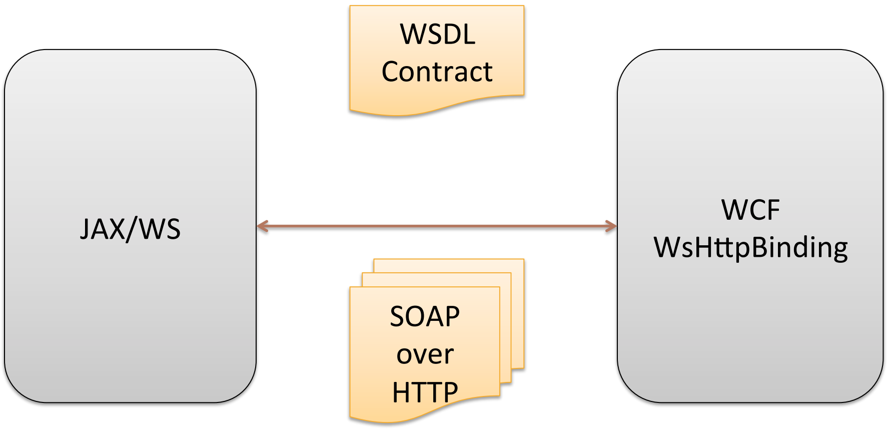
B2B E-Commerce (pre-WS)



7

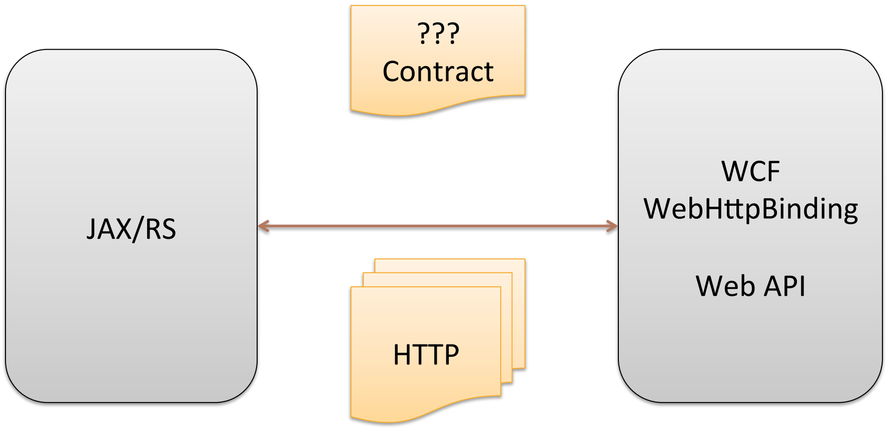


B2B E-Commerce (SOAP)



8

B2B E-Commerce (REST)



9

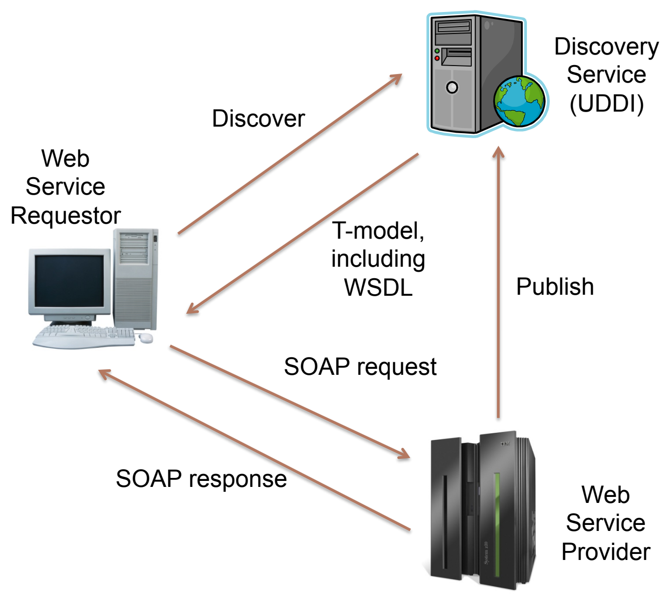
SOAP-BASED WEB SERVICES

10

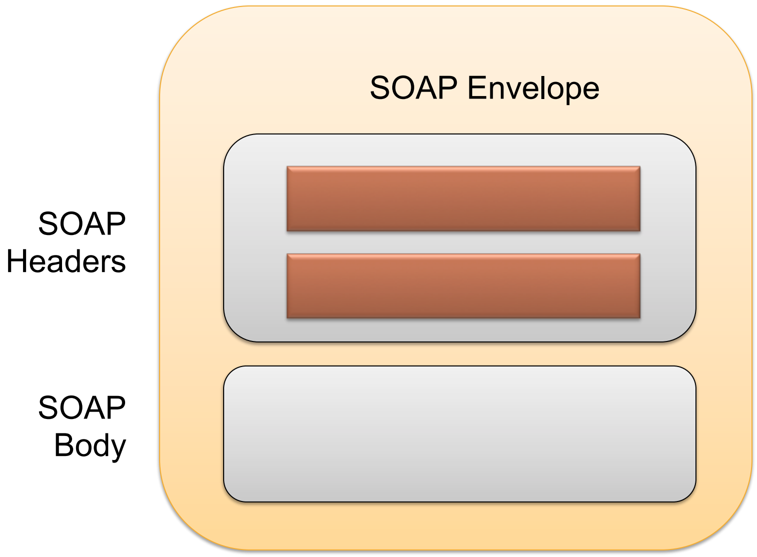
Simple Object Access Protocol (SOAP)

- Stateless one-way communication
  - Sync/async: depends on transport
  - No error-handling
  - Standard fault-signalling
- Message format
- Rules for processing messages
- How SOAP message should be transported
  - HTTP, SMTP, etc

11

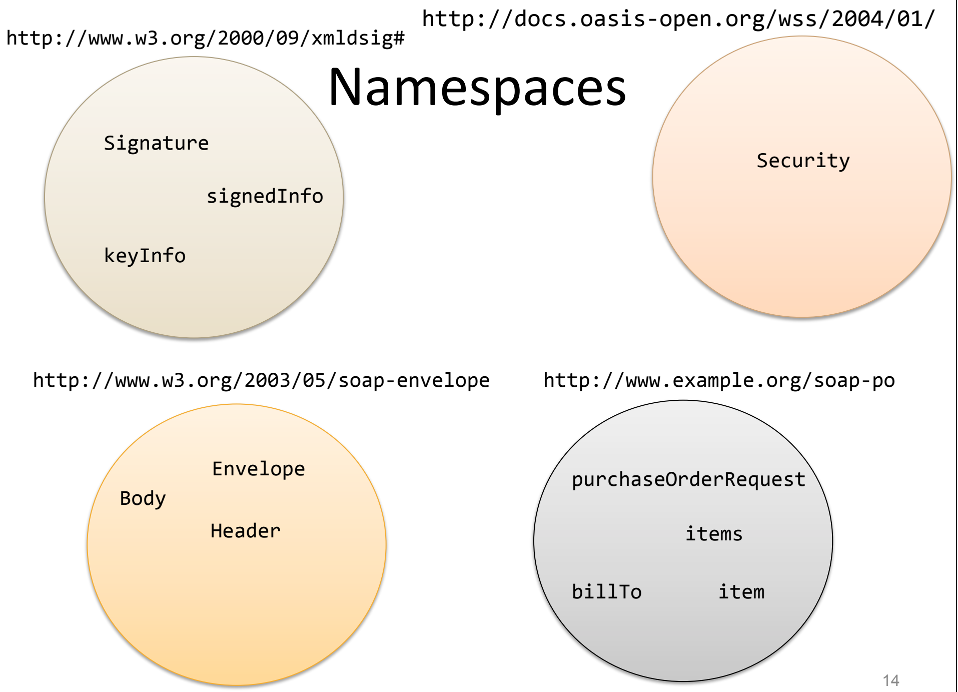


12

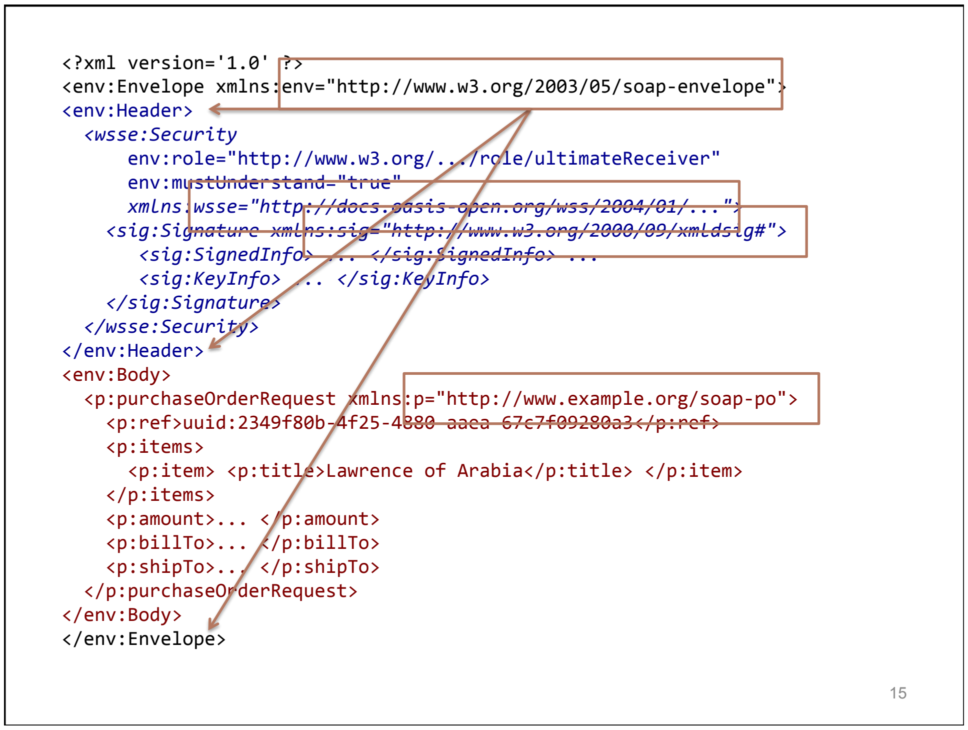


13

Namespaces


http://www.w3.org/2000/09/xmldsig#
Signature
signedInfo
keyInfo
http://docs.oasis-open.org/wss/2004/01/
Security
http://www.w3.org/2003/05/soap-envelope
Envelope
Body
Header
http://www.example.org/soap-po
purchaseOrderRequest
items
billTo
item

14



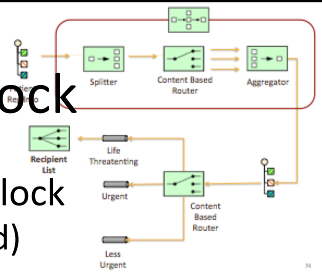
```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
<env:Header>
  <wsse:Security
      env:role="http://www.w3.org/.../role/ultimateReceiver"
      env:mustUnderstand="true"
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/..." >
    <sig:Signature xmlns:sig="http://www.w3.org/2000/09/xmldsig#">
       <sig:SignedInfo> ... </sig:SignedInfo> ...
       <sig:KeyInfo> ... </sig:KeyInfo>
    </sig:Signature>
  </wsse:Security>
</env:Header>
<env:Body>
  <p:purchaseOrderRequest xmlns:p="http://www.example.org/soap-po">
    <p:ref>uuid:2349f80b-4f25-4880-aaea-67c7f09280a3</p:ref>
    <p:items>
      <p:item> <p:title>Lawrence of Arabia</p:title> </p:item>
    </p:items>
    <p:amount>... </p:amount>
    <p:billTo>... </p:billTo>
    <p:shipTo>... </p:shipTo>
  </p:purchaseOrderRequest>
</env:Body>
</env:Envelope>
```

15

```
<?xml version='1.0' ?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope">
<env:Header>
  <wsse:Security
    env:role="…/role/ultimateReceiver"
    env:mustUnderstand="true"
    xmlns:wsse="http://docs.oasis-open.org/wss/…">
    <sig:Signature
       xmlns:sig="http://www.w3.org/2000/09/xmldsig#">
       <sig:SignedInfo> ... </sig:SignedInfo> ...
       <sig:KeyInfo> ... </sig:KeyInfo>
    </sig:Signature>
  </wsse:Security>
</env:Header>
<env:Body>
   ...
</env:Body>
</env:Envelope>
```

16

SOAP Header Block



- `role`: who may process header block
  - `ultimateReceiver` (end-to-end)
  - `next`
  - `none`
  - application-defined
- `mustUnderstand`: mandate processing
  - Fault back to sender o/w
- `relay`: forward header if not processed

- Ex: `role="next", mustUnderstand="false", relay="true"`

17

SOAP: Document/literal style

```
<?xml version='1.0' ?>
<env:Envelope
   xmlns:env="http://www.w3.org/2003/05/soap-envelope">
<env:Header>
  ...
</env:Header>
<env:Body>
  <p:purchaseOrderRequest
        xmlns:p="http://www.example.org/soap-po">
    <p:ref>uuid:2349f80b-4f25-4880-aaea-67c7f09280a3</p:ref>
    <p:items>
      <p:item>
        <p:title>Lawrence of Arabia</p:title>
      </p:item>
    </p:items>
    <p:amount>... </p:amount>
    <p:billTo>... </p:billTo>
    <p:shipTo>... </p:shipTo>
  </p:purchaseOrderRequest>
</env:Body>
</env:Envelope>
```

18

SOAP Interaction Styles

- Document/literal style
  - Left to application
- RPC/literal style
  - Method signatures
  - Difficult to validate
- RPC/encoded style

SOAP: RPC/encoded style

```
<?xml version='1.0' ?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope">
<env:Header> ... </env:Header>
<env:Body>
  <purchaseOrderRequest
      xmlns:p="http://www.example.org/soap-po">
    <ref>uuid:2349f80b-4f25-4880-aaea-67c7f09280a3</ref>
    <items>
      <p:item
        env:encodingStyle=
          "http://www.w3.org/2003/05/soap-encoding">
        <p:title>Lawrence of Arabia</p:title>
      </p:item>
    </items>
    <amount> ... </amount>
    <billTo> ... </billTo>
    <shipTo> ... </shipTo>
  </purchaseOrderRequest>
</env:Body>
</env:Envelope>
```

SOAP Response

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-
envelope">
<env:Header> ... </env:Header>
<env:Body>
  <p:purchaseOrderResponse
      xmlns:p="http://www.example.org/soap-po">
    <p:ref>
      uuid:2349f80b-4f25-4880-aaea-67c7f09280a3
    </p:ref>
    <p:ourRef> ... </p:ourRef>
    <p:dateReceived> 2011-02-14 </p:dateReceived>
  </p:purchaseOrderResponse>
</env:Body>
</env:Envelope>
```

SOAP Fault

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-
envelope">
<env:Header> </env:Header>
<env:Body>
  <env:Fault
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/...">
    <env:Code>
      <env:Value>env:Sender</env:Value>
    </env:Code>
    <env:Subcode>
      <env:Value>wsse:FailedCheck</env:Value>
    </env:Subcode>
    <env:Reason>
      <env:Text xml:lang="en-US”>
        The signature or decryption was invalid
      </env:Text>
    </env:Reason>
  </env:Fault>
</env:Body>
</env:Envelope>
```

SOAP Binding

- Binding to transport protocol
  - Ex: RPC request as HTTP POST
  - Response as part of HTTP POST response

  - Ex: RPC request as email (SMTP)
  - Response as reply email

Addressing

- HTTP request header

```
POST /finance HTTP/1.1
Host: www.example.com
```

- WS-Addressing

```
<env:Header xmlns:wsa="...">
  <wsa:To>
    http://www.example.com/finance
  </wsa:To>
  <wsa:Action>
    http://www.example.com/ConfirmAuthorization
  </wsa:Action>
</env:Header>
```

Message Exchange Patterns

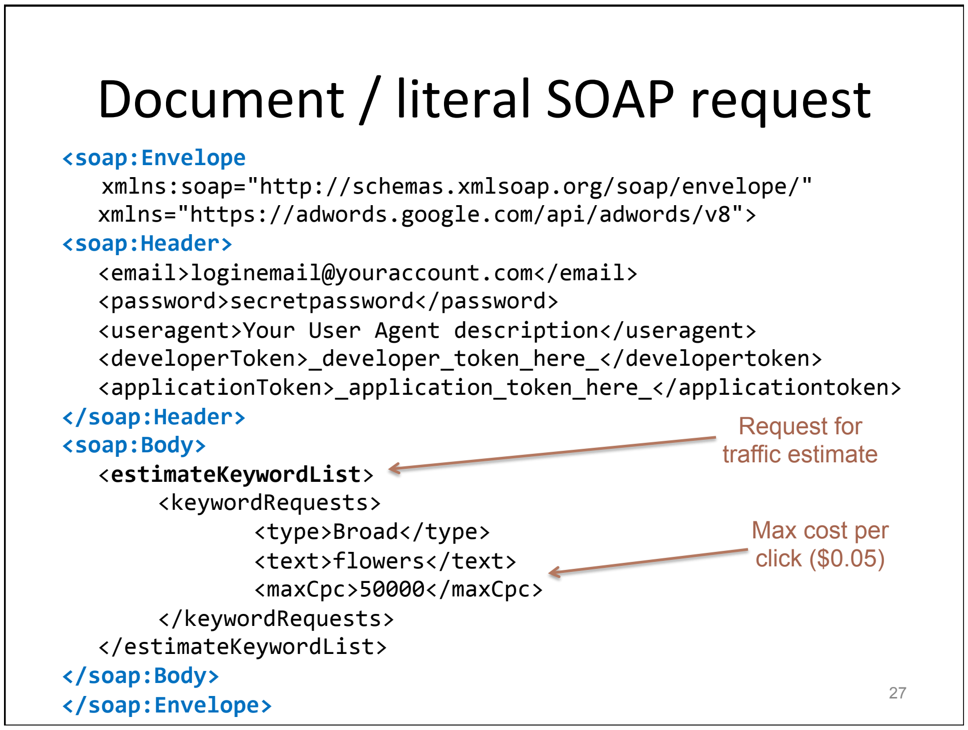
- SOAP-defined MEPs
  - SOAP *request-response* MEP
  - SOAP *response* MEP (non-SOAP request)
- SOAP HTTP binding
  - Request-response → POST
  - Response → GET

25

Example: Google Adwords

- Idea: Display relevant ads with search results
- Relevance based on *keywords*
  - Specified by customer
  - Charged based on click-through
- Customer manages ad placement
  - Monitor performance
  - Set budget for max number of clicks
  - Change keywords

26



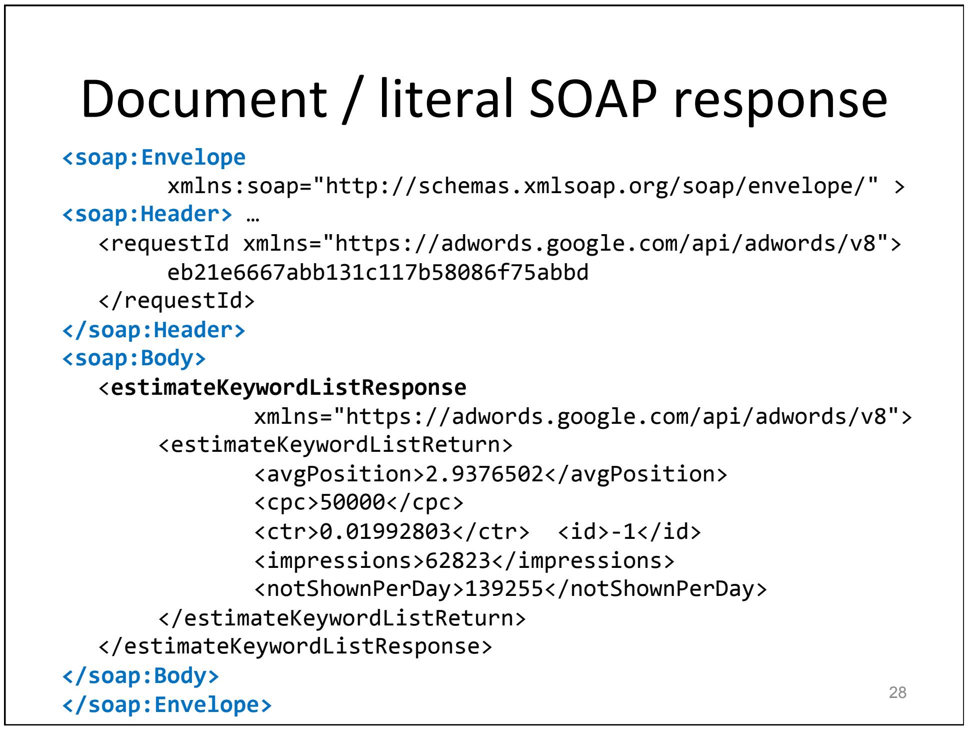
Document / literal SOAP request

```
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns="https://adwords.google.com/api/adwords/v8">
<soap:Header>
   <email>loginemail@youraccount.com</email>
   <password>secretpassword</password>
   <useragent>Your User Agent description</useragent>
   <developerToken>_developer_token_here_</developertoken>
   <applicationToken>_application_token_here_</applicationtoken>
</soap:Header>
<soap:Body>
   <estimateKeywordList>
        <keywordRequests>
                <type>Broad</type>
                <text>flowers</text>
                <maxCpc>50000</maxCpc>
        </keywordRequests>
   </estimateKeywordList>
</soap:Body>
</soap:Envelope>
```

Request for traffic estimate

Max cost per click ($0.05)

27



Document / literal SOAP response

```
<soap:Envelope
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
<soap:Header> …
   <requestId xmlns="https://adwords.google.com/api/adwords/v8">
        eb21e6667abb131c117b58086f75abbd
   </requestId>
</soap:Header>
<soap:Body>
   <estimateKeywordListResponse
                xmlns="https://adwords.google.com/api/adwords/v8">
        <estimateKeywordListReturn>
                <avgPosition>2.9376502</avgPosition>
                <cpc>50000</cpc>
                <ctr>0.01992803</ctr>  <id>-1</id>
                <impressions>62823</impressions>
                <notShownPerDay>139255</notShownPerDay>
        </estimateKeywordListReturn>
   </estimateKeywordListResponse>
</soap:Body>
</soap:Envelope>
```

28

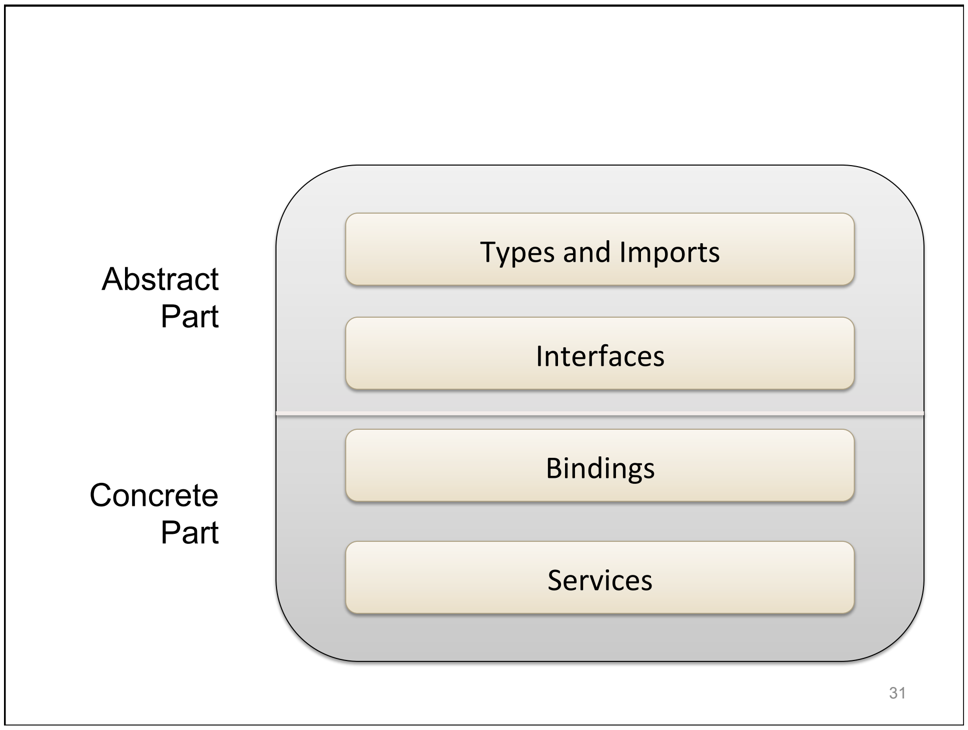
WEB SERVICES DESCRIPTION LANGUAGE (WSDL)

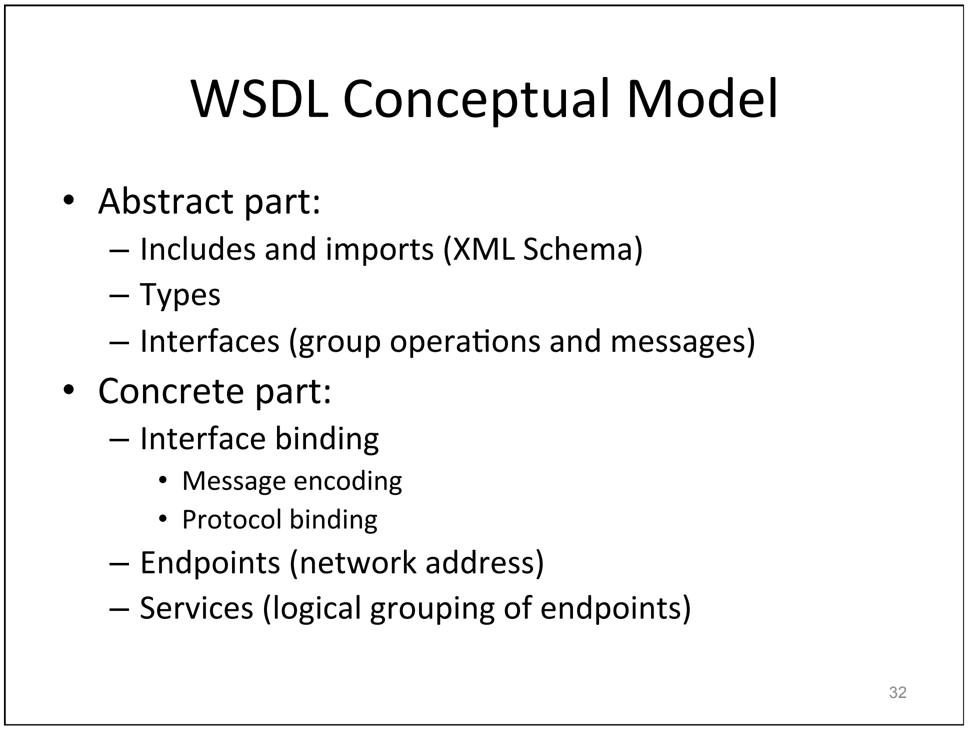
29

Web Services Description Language (WSDL)

- Operation: message exchange
- Interface (PortType): group related operations
- Binding: concrete details about connection
  - Message transfer protocol, e.g., HTTP
  - Endpoint URL

30

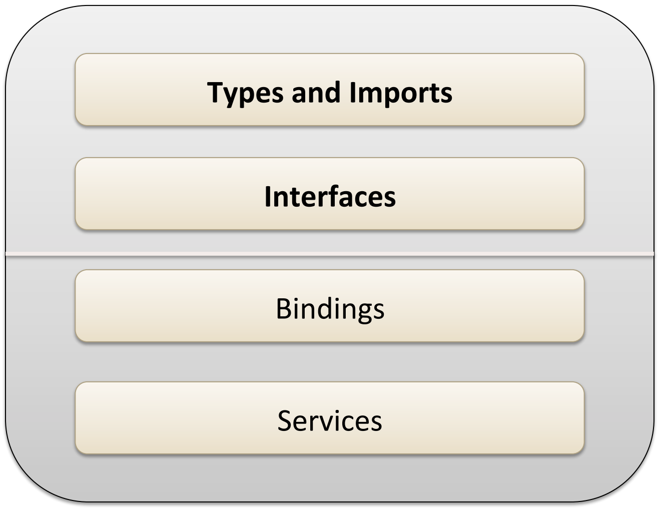




WSDL Conceptual Model

- Abstract part:
  - Includes and imports (XML Schema)
  - Types
  - Interfaces (group operations and messages)
- Concrete part:
  - Interface binding
    - Message encoding
    - Protocol binding
  - Endpoints (network address)
  - Services (logical grouping of endpoints)

32



33

Google Adwords WSDL: Keyword Request Message Type

```
<complexType name="KeywordRequest">
   <sequence>
        <element name="id" minOccurs="0" type="xsd:long"/>
        <element name="maxCpc" minOccurs="0" type="xsd:long"/>
        <element name="negative" minOccurs="0" type="xsd:boolean"/>
        <element name="text" minOccurs="0" type="xsd:string"/>
        <element name="type" minOccurs="0" type="impl:KeywordType"/>
   </sequence>
</complexType>
```

34

Google Adwords WSDL: Keyword Request Message Type

```
<complexType name="KeywordRequest">
   <sequence>
        <element name="id" minOccurs="0" type="xsd:long"/>
        <element name="maxCpc" minOccurs="0" type="xsd:long"/>
        <element name="negative" minOccurs="0" type="xsd:boolean"/>
        <element name="text" minOccurs="0" type="xsd:string"/>
        <element name="type" minOccurs="0" type="impl:KeywordType"/>
   </sequence>
</complexType>
<element name="estimateKeywordList">
   <complexType> <sequence>
        <element name="keywordRequests" maxOccurs="unbounded"
                type="data:KeywordRequest"/>
   </sequence> </complexType>
</element>
```

35

Google Adwords WSDL: Keyword Request Message Type

```
<complexType name="KeywordRequest">
   <sequence>
        <element name="id" minOccurs="0" type="xsd:long"/>
        <element name="maxCpc" minOccurs="0" type="xsd:long"/>
        <element name="negative" minOccurs="0" type="xsd:boolean"/>
        <element name="text" minOccurs="0" type="xsd:string"/>
        <element name="type" minOccurs="0" type="impl:KeywordType"/>
   </sequence>
</complexType>
<element name="estimateKeywordList">
   <complexType> <sequence>
        <element name="keywordRequests" maxOccurs="unbounded"
                type="data:KeywordRequest"/>
   </sequence> </complexType>
</element>
<wsdl:message name="estimateKeywordListRequest">
   <wsdl:part element="data:estimateKeywordList" name="parameters"/>
</wsdl:message>
```

36

Google Adwords WSDL: Traffic Estimator Interface

```
<wsdl:portType name="TrafficEstimatorInterface">

   <wsdl:operation name="checkKeywordTraffic">
        …
   </wsdl:operation>

   <wsdl:operation name="estimateKeywordList">
        <wsdl:input message="svc:estimateKeywordListRequest"
               name="estimateKeywordListRequest"/>
        <wsdl:output message="svc:estimateKeywordListResponse"
               name="estimateKeywordListResponse"/>
        <wsdl:fault message="svc:ApiException"/>
   </wsdl:operation>

</wsdl:portType>
```

37

Data vs Service Namespace

```
<definitions
    xmlns="http://schemas.xmlsoap.org/wsdl"
    xmlns:svc="http://example.org/Shopping"
    targetNamespace="http://example.org/Shopping">
  <types
     xmlns:data="http://example.org/Product"
     targetNamespace="http://example.org/Product">
    <element name="RequestData"> …
  </types>
  <message name="CheckoutRequest">
    <part element="data:RequestData" .../>
  </message>
  <portType>
    <operation name="Checkout">
      <input message="svc:CheckoutRequest" ...>
    </operation>
  </portType>
</definitions>
```

38

WSDL Message Exchange Patterns

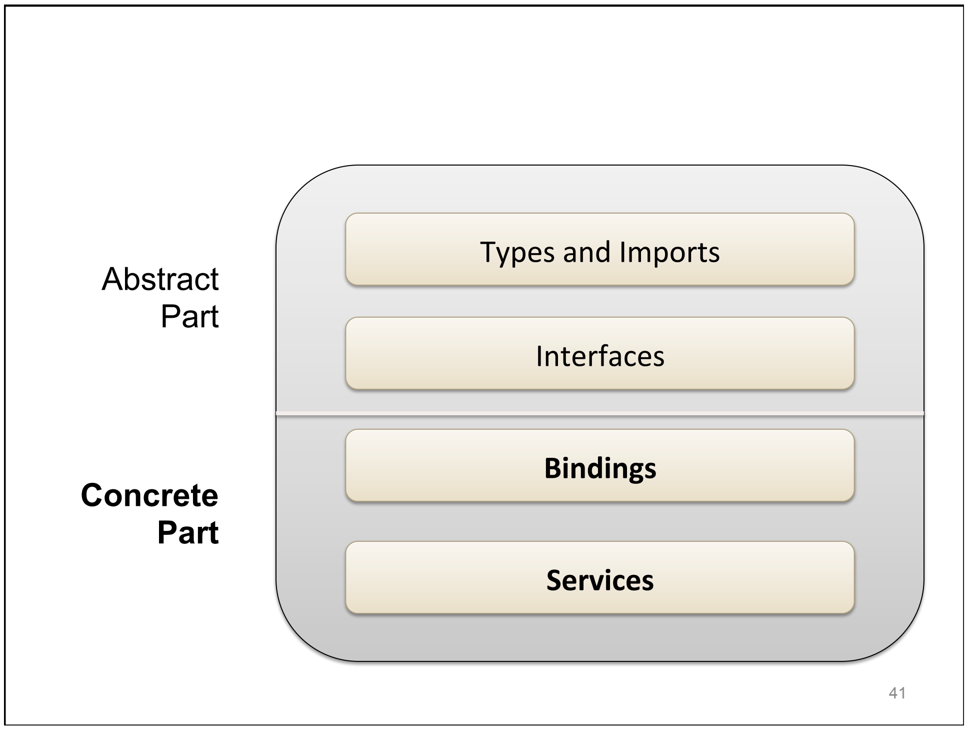
- Standard (WSDL 2.0)
  - In-Only
  - Robust In-Only
    - Fault may *trigger* message
    - Ex: “No such recipient”
    - *Must* be delivered back to sender
  - In-Out
    - Not necessarily synchronous
    - Fault message may *replace* result message
    - Ex: Authentication failure

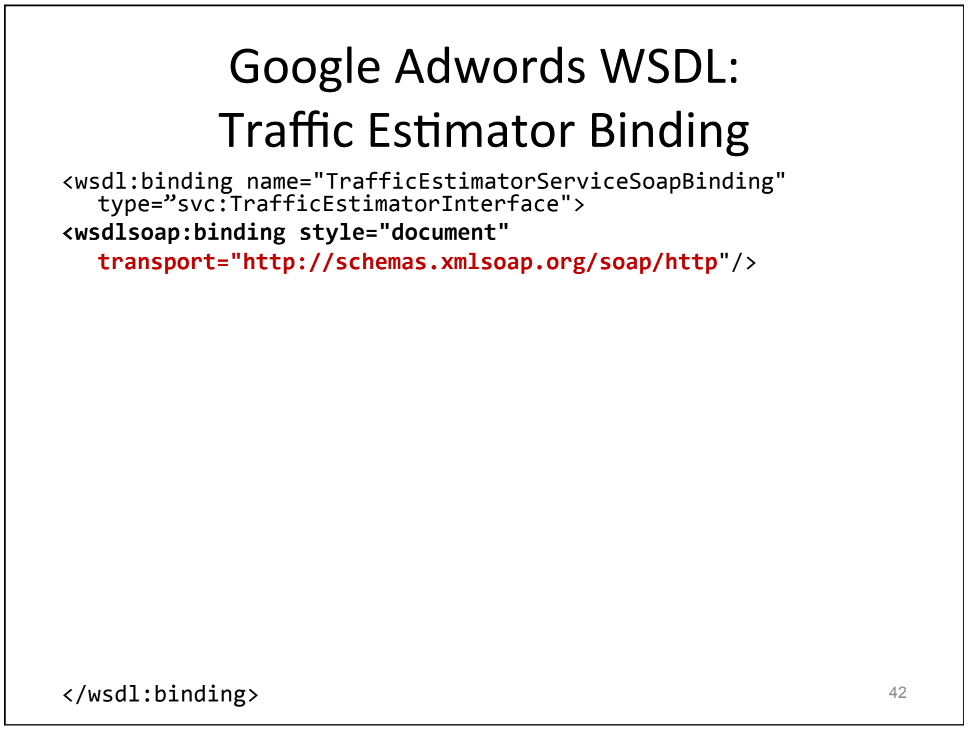
39

WSDL Message Exchange Patterns

- Non-normative
  - In-Optional-Out
  - Out-Only
    - Ex: WCF duplex MEP
  - Robust Out-Only
  - Out-In
  - Out-Optional-In

40





Google Adwords WSDL: Traffic Estimator Binding

```
<wsdl:binding name="TrafficEstimatorServiceSoapBinding"
   type="svc:TrafficEstimatorInterface">
<wsdlsoap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http"/>









</wsdl:binding>
```

42

Google Adwords WSDL: Traffic Estimator Binding

```
<wsdl:binding name="TrafficEstimatorServiceSoapBinding"
   type=”svc:TrafficEstimatorInterface">
<wsdlsoap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="estimateKeywordList">









   </wsdl:operation>
</wsdl:binding>
```

Google Adwords WSDL: Traffic Estimator Binding

```
<wsdl:binding name="TrafficEstimatorServiceSoapBinding"
   type="svc:TrafficEstimatorInterface”>
<wsdlsoap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="estimateKeywordList">
        <wsdl:input name="estimateKeywordListRequest">
            <wsdlsoap:header message=”data:useragent"
               part="useragent“ use="literal"/>
            <wsdlsoap:header message=”data:password"
               part="password" use="literal"/>
            <wsdlsoap:header message=”data:developerToken"
               part="developerToken" use="literal"/>

        </wsdl:input>



   </wsdl:operation>
</wsdl:binding>
```

Google Adwords WSDL: Traffic Estimator Binding

```
<wsdl:binding name="TrafficEstimatorServiceSoapBinding"
   type=”svc:TrafficEstimatorInterface">
<wsdlsoap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="estimateKeywordList">
        <wsdl:input name="estimateKeywordListRequest">
            <wsdlsoap:header message=”data:useragent"
               part="useragent“ use="literal"/>
            <wsdlsoap:header message=”data:password"
               part="password" use="literal"/>
            <wsdlsoap:header message=”data:developerToken"
               part="developerToken" use="literal"/>
            <wsdlsoap:body use="literal"/>
        </wsdl:input>



   </wsdl:operation>
</wsdl:binding>
```

45

Google Adwords WSDL: Traffic Estimator Binding

```
<wsdl:binding name="TrafficEstimatorServiceSoapBinding"
   type=”svc:TrafficEstimatorInterface">
<wsdlsoap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="estimateKeywordList">
        <wsdl:input name="estimateKeywordListRequest">
            <wsdlsoap:header message=”data:useragent"
               part="useragent“ use="literal"/>
            <wsdlsoap:header message=”data:password"
               part="password" use="literal"/>
            <wsdlsoap:header message=”data:developerToken"
               part="developerToken" use="literal"/>
            <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="estimateKeywordListResponse">
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
   </wsdl:operation>
</wsdl:binding>
```

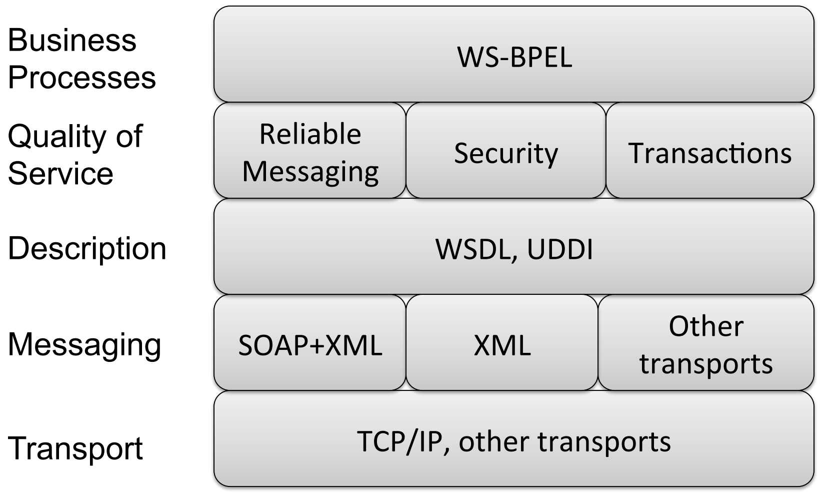
46

Google Adwords WSDL: Traffic Estimator Service

```
<wsdl:service name="TrafficEstimatorService">
<wsdl:port
   binding=”svc:TrafficEstimatorServiceSoapBinding"
   name="TrafficEstimatorService">

<wsdlsoap:address
   location="https://adwords.google.com/api/adwords/v13/
   TrafficEstimatorService"/>

</wsdl:port>
</wsdl:service>
```

47

SOAP Web Services Stack



48

SOAP Web Services Stack

- WS-Addressing
- WS-Transaction
- WS-Reliable-Messaging
- WS-Security
- WS-Policy
- WS-Trust
- WS-Discovery
- WS-BPEL

49

WS-POLICY

50

WS-Policy

- Issue: SOA governance
  - Quality of service
  - Data consistency
  - Data security
- WSDL: contracts for services
- WS-Policy: enforcement of policies
  - Language for defining policy languages
  - Automatic run-time enforcement
  - Policy vs mechanism

51

Example: WS-Addressing

wsa: WS-Addressing

- SOAP message

```
<soap:Envelope>
  <soap:Header>
    <wsa:To>http://www.example.org/finance</wsa:To>
    <wsa:Action>
      http://www.example.org/ConfirmAuthorization
    </wsa:Action>
  </soap:Header>
  <soap:Body>...</soap:Body>
</soap:Envelope>
```

52

Example: WS-Addressing Metadata

wsam: WS-Addressing Metadata

- Policy assertion

```
<wsdl:binding
    name="FinanceBindingWithWSA"
    type="tns:FinanceInterface">
  <Policy>
    <wsam:Addressing>...</wsam:Addressing>
  </Policy>
  ...
</wsdl:binding>
```

53

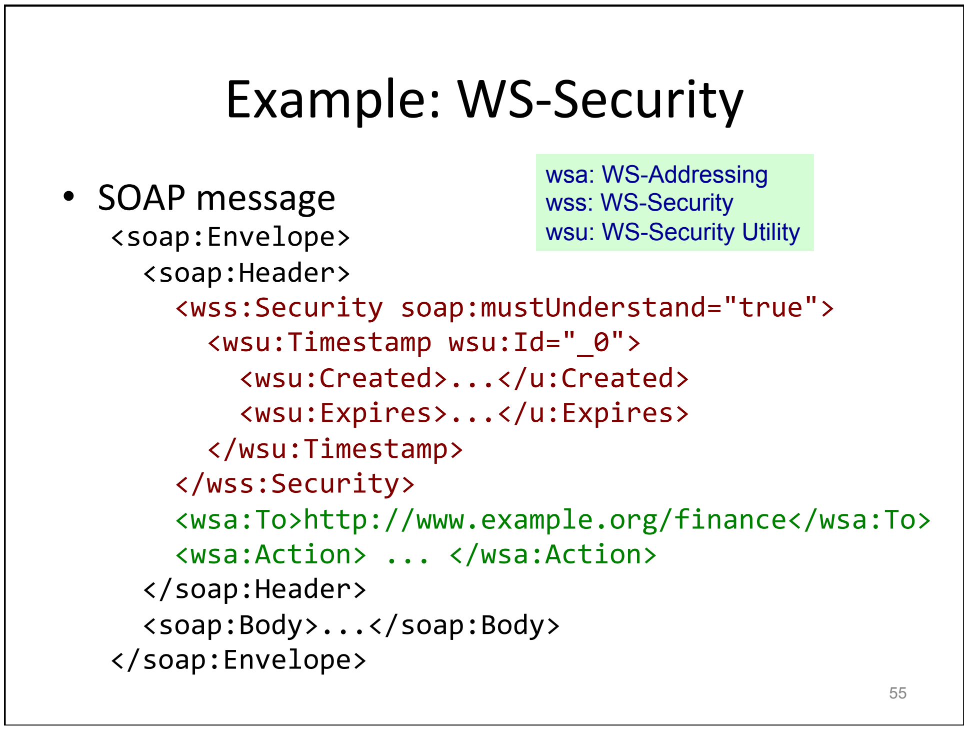
Example: WS-SecurityPolicy

wsam: WS-Addressing Metadata
sp: WS-SecurityPolicy

- Policy assertion

```
<wsdl:binding
    name="FinanceBindingWithWSA"
    type="tns:FinanceInterface">
  <Policy>
    <All>
      <wsam:Addressing>...</wsam:Addressing>
      <sp:TransportBinding>...</sp:TransportBinding>
    </All>
  </Policy>
  ...
</wsdl:binding>
```

54



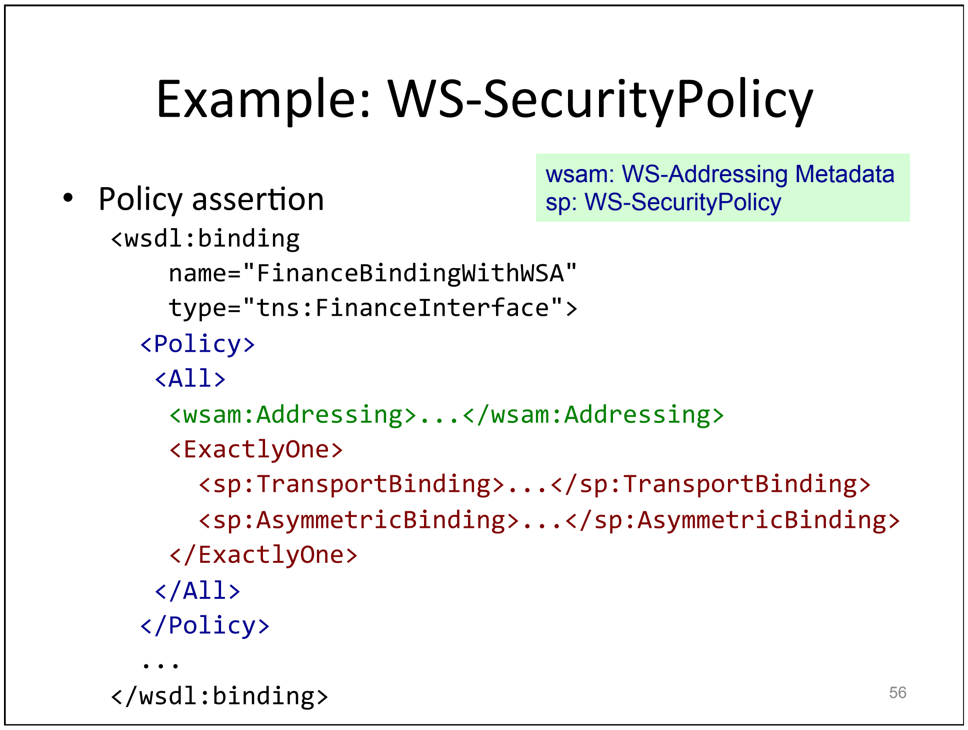
Example: WS-Security

wsa: WS-Addressing
wss: WS-Security
wsu: WS-Security Utility

- SOAP message

```
<soap:Envelope>
  <soap:Header>
    <wss:Security soap:mustUnderstand="true">
      <wsu:Timestamp wsu:Id="_0">
        <wsu:Created>...</u:Created>
        <wsu:Expires>...</u:Expires>
      </wsu:Timestamp>
    </wss:Security>
    <wsa:To>http://www.example.org/finance</wsa:To>
    <wsa:Action> ... </wsa:Action>
  </soap:Header>
  <soap:Body>...</soap:Body>
</soap:Envelope>
```

55



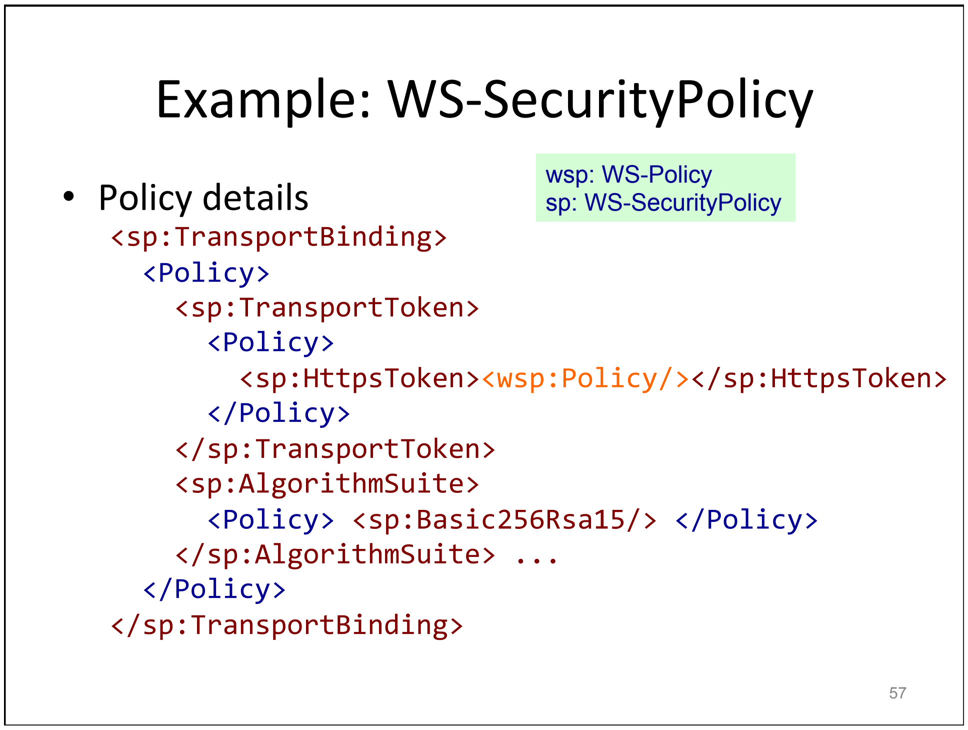
Example: WS-SecurityPolicy

wsam: WS-Addressing Metadata
sp: WS-SecurityPolicy

- Policy assertion

```
<wsdl:binding
    name="FinanceBindingWithWSA"
    type="tns:FinanceInterface">
  <Policy>
   <All>
    <wsam:Addressing>...</wsam:Addressing>
    <ExactlyOne>
      <sp:TransportBinding>...</sp:TransportBinding>
      <sp:AsymmetricBinding>...</sp:AsymmetricBinding>
    </ExactlyOne>
   </All>
  </Policy>
  ...
</wsdl:binding>
```

56



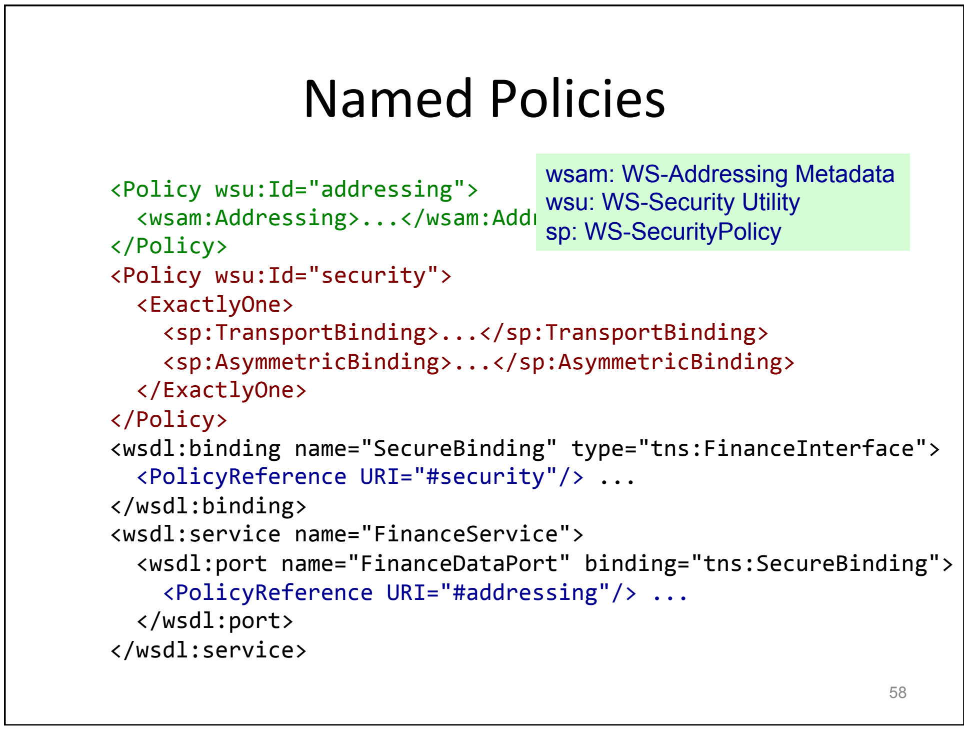
Example: WS-SecurityPolicy

- Policy details

wsp: WS-Policy
sp: WS-SecurityPolicy

```
<sp:TransportBinding>
  <Policy>
    <sp:TransportToken>
      <Policy>
        <sp:HttpsToken><wsp:Policy/></sp:HttpsToken>
      </Policy>
    </sp:TransportToken>
    <sp:AlgorithmSuite>
      <Policy> <sp:Basic256Rsa15/> </Policy>
    </sp:AlgorithmSuite> ...
  </Policy>
</sp:TransportBinding>
```

57



Named Policies

wsam: WS-Addressing Metadata
wsu: WS-Security Utility
sp: WS-SecurityPolicy

```
<Policy wsu:Id="addressing">
  <wsam:Addressing>...</wsam:Addr
</Policy>
<Policy wsu:Id="security">
  <ExactlyOne>
    <sp:TransportBinding>...</sp:TransportBinding>
    <sp:AsymmetricBinding>...</sp:AsymmetricBinding>
  </ExactlyOne>
</Policy>
<wsdl:binding name="SecureBinding" type="tns:FinanceInterface">
  <PolicyReference URI="#security"/> ...
</wsdl:binding>
<wsdl:service name="FinanceService">
  <wsdl:port name="FinanceDataPort" binding="tns:SecureBinding">
    <PolicyReference URI="#addressing"/> ...
  </wsdl:port>
</wsdl:service>
```

58

WINDOWS COMMUNICATION FOUNDATION (WCF)

59

WCF Overview

- Framework for B2B e-commerce
- Interoperability via SOAP/WSDL

- Separate parts of a service
  - Contract
  - Implementation
  - Binding

60

WCF Contracts

- Service contract
  - Operations
  - Message exchange patterns
  - Instancing
- Data contract
- Fault contract
- Message contract (optional)

61

WCF Common Bindings

- `WsHttpBinding`
  - SOAP Web services
- `NetTcpBinding`
  - Optimized for WCF-to-WCF
- `NetNamedPipeBinding`
  - IPC communication (same-machine)
- `WebHttpBinding`
  - REST Web services
- `BasicHttpBinding`
  - Legacy ASMX Web services
- `NetMsmqBinding`

62

WCF HOSTING

63

WCF Hosting

- IIS Web Server
- Self-Hosting
- Windows Activation Service (WAS)
- AppFabric

64

Hosting in Web Server

- Service description in .svc file

```
<%@ ServiceHost
  Language = “C#”
  CodeBehind = “~/App_Code/MyService.cs”
  Service = “MyService”
%>
```

- web.config file

```
<system.serviceModel>
  <services>
    <service name=“MyNameSpace.MyService”>
      ...
    </service>
  </services>
<system.serviceModel>
```

65

Hosting in Web Server

- Description in web.config file

```
<system.serviceModel>
  <serviceHostingEnvironment>
    <serviceActivations>
      <add relativeAddress = “MyService.svc”
           service = “MyNameSpace.MyService” />
    </serviceActivations>
  </serviceHostingEnvironment>
  <services>
    <service name=“MyNameSpace.MyService”>
      ...
    </service>
  </services>
<system.serviceModel>
```

66

Self-Hosting

- app.config file

```
<services>
  <service name = “MyNamespace.Service”>
    ...
  </service>
</services>
```

- Main program

```
static void Main() {
  ServiceHost host =
          new ServiceHost(typeof(MyService));
  host.Open();
  Appication.Run(new MyForm());
  host.Close();
}
```

67

Self-Hosting

- app.config file

```
<services>
  <service name = “MyNamespace.Service”>
  </service>
</services>
```

- Main program

```
Uri tcpBaseAddress =
    new Uri(“net.tcp://localhost:8001/”);
Uri httpBaseAddress =
    new Uri(“http://localhost:8002/”);
ServiceHost host =
    new ServiceHost(typeof(MyService),
                    tcpBaseAddress,
                    httpBaseAddress);
```

68

Hosting in WAS

- Web Server hosting
  - Limited to HTTP
  - Complex, unstable
- Windows Activation Service (WAS)
  - Configuration as in IIS
  - All transports available
  - Runtime support: Application pooling, etc

69

Custom Hosting in WAS

- Service description in .svc file

```
<%@ ServiceHost
   ... Factory = “MyServiceFactory”
%>
```

- web.config file

```
class MyServiceFactory : ServiceHostFactory {
  protected override ServiceHost
    CreateServiceHost(Type serviceType,
                      Uri[] baseAddresses) {
    ServiceHost host =
        new ServiceHost(serviceType, baseAddresses);
    ...
    return host;
} }
```

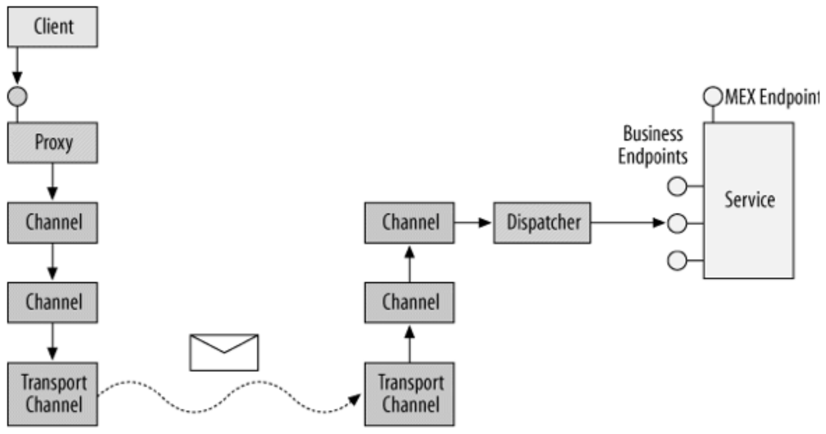
70

Hosting in AppFabric

- WAS
  - General purpose hosting engine (IIS and WCF)
- AppFabric
  - Extension to WAS, specific to WCF
  - Additional configuration options
  - Monitoring, instrumentation and event tracking
  - Auto-start services without client requests

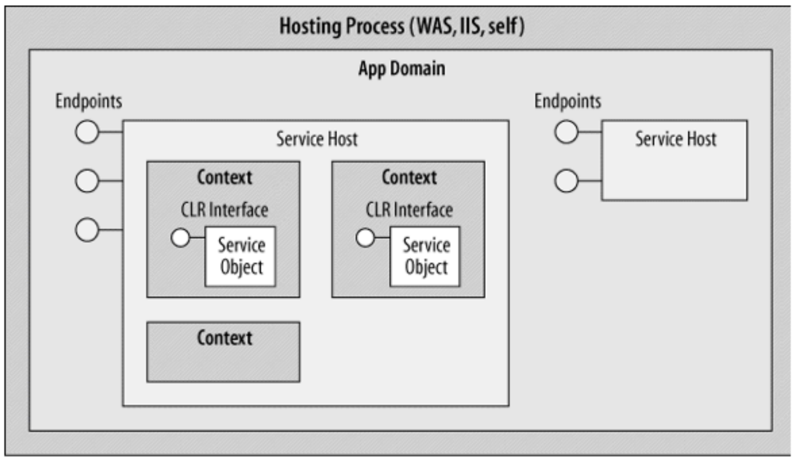
71

WCF Architecture



72

Hosting Architecture



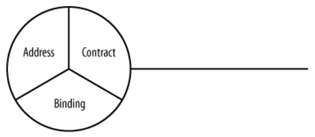
73

ENDPOINTS

74

Endpoint

```
namespace MyNamespace
{
  [ServiceContract]
  interface IMyContract {...}
  class MyService : IMyContract {...}
}

<system.serviceModel>
  <services>
    <service name="MyNamespace.MyService”>
      <endpoint address = "http://localhost:8000/MyService"
                binding = "wsHttpBinding”
                contract = "MyNamespace.IMyContract" />
    </service>
  </services>
</system.serviceModel>
```



75

Multiple Endpoints

```
<services>
 <service name = “MyService”>
  <endpoint address = “http://localhost:8000/MySvc”
            binding = “wsHttpBinding”
            contract = “IMyContract” />
  <endpoint address = “net.tcp://localhost:8000/MySvc”
            binding = “netTcpBinding”
            contract = “IMyContract” />
  <endpoint address = “net.tcp://localhost:8001/MySvc2”
            binding = “netTcpBinding”
            contract = “IMyOtherContract” />
 </service>
</services>
```

76

Programmatically

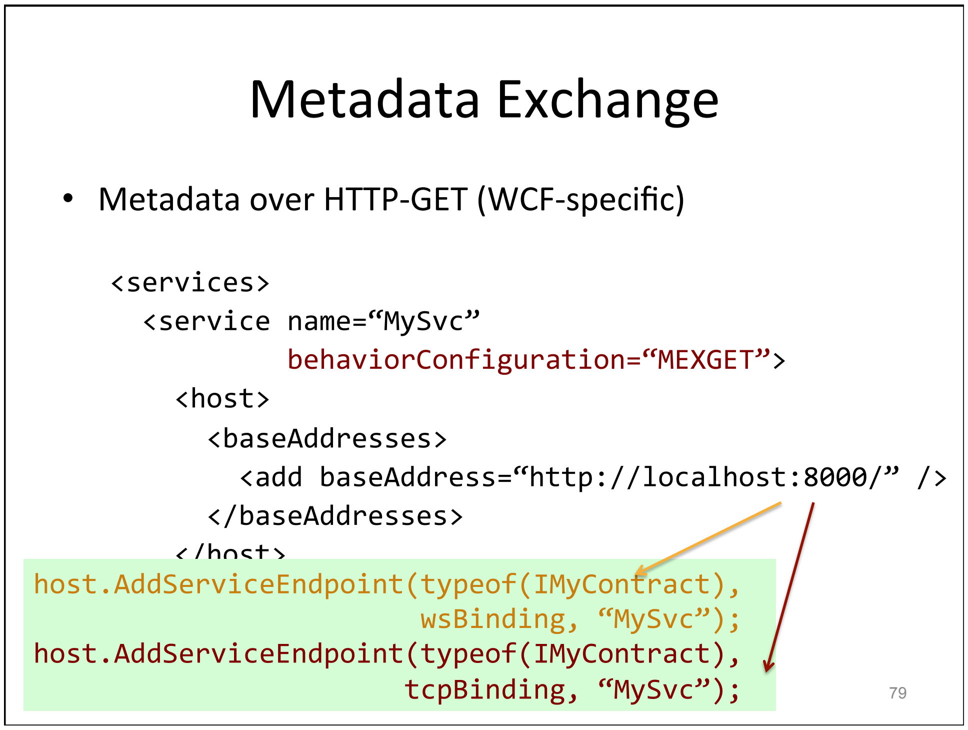
```
ServiceHost host = new ServiceHost(typeof(MyService));

Binding wsBinding = new WSHttpBinding();
Binding tcpBinding = new NetTcpBinding();

host.AddServiceEndpoint(typeof(IMyContract),
    wsBinding, “http://localhost:8000/MySvc”);
host.AddServiceEndpoint(typeof(IMyContract),
    tcpBinding, “net.tcp://localhost:8000/MySvc”);
host.AddServiceEndpoint(typeof(IMyOtherContract),
    tcpBinding, “net.tcp://localhost:8001/MySvc2”);

host.Open();
```

77

Binding Configuration

```
<services>
 <service name = “MyService”>
  <endpoint address = “net.tcp://localhost:8000/MySvc”
            bindingConfiguration = “TransTCP”
            binding = “netTcpBinding”
            contract = “IMyContract” />
  <endpoint address = “net.tcp://localhost:8001/MySvc2”
            bindingConfiguration = “TransTCP”
            binding = “netTcpBinding”
            contract = “IMyOtherContract” />
 </service>
</services>
<bindings>
  <netTcpBinding>
    <binding name = “TransTCP” transactionFlow = “true” />
  </netTcpBinding>
</bindings>
```

78

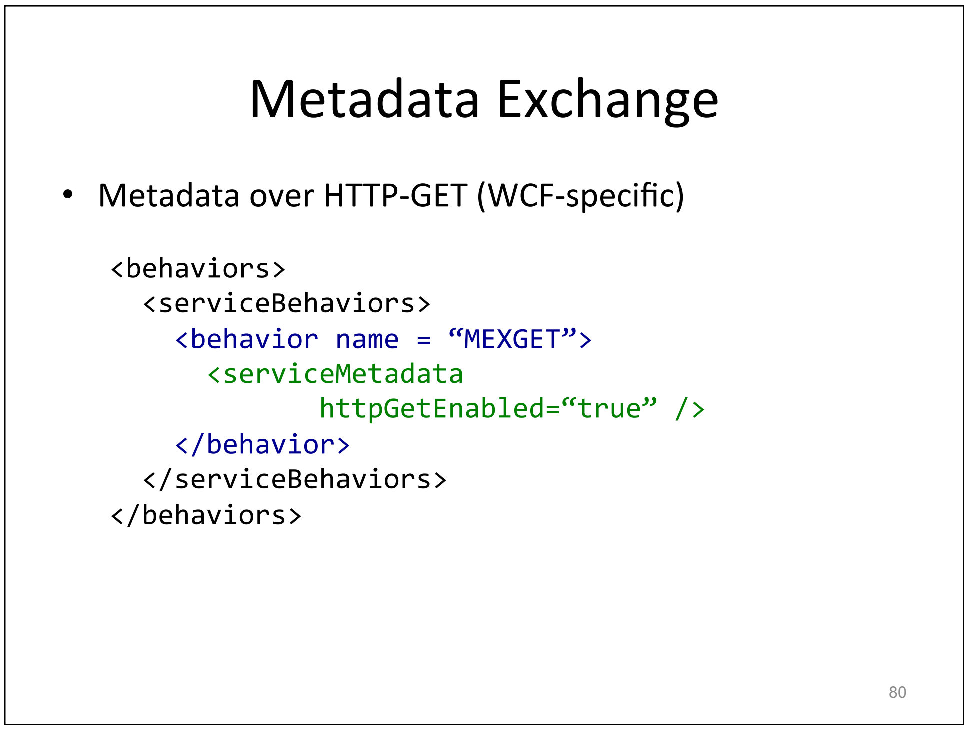


Metadata Exchange

- Metadata over HTTP-GET (WCF-specific)

```
<services>
  <service name="MySvc"
           behaviorConfiguration="MEXGET">
    <host>
      <baseAddresses>
        <add baseAddress="http://localhost:8000/" />
      </baseAddresses>
    </host>
host.AddServiceEndpoint(typeof(IMyContract),
                        wsBinding, "MySvc");
host.AddServiceEndpoint(typeof(IMyContract),
                       tcpBinding, "MySvc");
```

79



Metadata Exchange

- Metadata over HTTP-GET (WCF-specific)

```
<behaviors>
  <serviceBehaviors>
    <behavior name = "MEXGET">
      <serviceMetadata
             httpGetEnabled="true" />
    </behavior>
  </serviceBehaviors>
</behaviors>
```

80

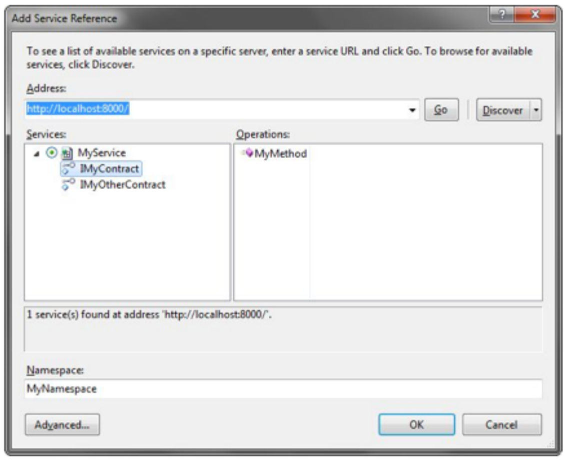
Metadata Exchange Endpoint

- Industry standard MEX endpoint

```
<behaviors>
  <serviceBehaviors>
    <behavior name = “MEXGET”>
      <serviceMetadata
                                 />
    </behavior>
  </serviceBehaviors>
</behaviors>
```

81

Generating a Client Proxy



Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://localhost:8000/

Services: MyService, IMyContract, IMyOtherContract

Operations: MyMethod

1 service(s) found at address 'http://localhost:8000/'.

Namespace: MyNamespace

82

Programmatic Client Generation

```
WSHttpBinding wsBinding = new WSHttpBinding();

wsBinding.SendTimeout = TimeSpan.FromMinutes(5);
wsBinding.TransactionFlow = true;

EndpointAddress addr = new EndpointAddress(...);

MyContractClient client =
  new MyContractClient(wsBinding,addr);

client.MyMethod(...);

client.Close();
```

83

WCF DATA CONTRACT

84

Serialization

- Serialized attribute

```
[Serialized]
class MyClassA {
   [NonSerialized]
   MyClassB b;
}
```

- .NET formatters
  - BinaryFormatter
  - SoapFormatter
- WCF formatter (no type info)
  - DataContractSerializer

85

Data Contract

Data transfer object (DTO)

```
[DataContract]
public class Product {
   [DataMember]
   public string Description;

   [DataMember]
   private int ProductID;

   private int inventoryCode;

   private float price;

   [DataMember]
   public float Price{
      get { return price; }
   }
}
```

86

Data Contract

```
[DataContract(
   Namespace=“http://www.example.org/Product”)]
public class Product {
   [DataMember]
   public string Description;

   [DataMember]
   private int ProductID;

   private int inventoryCode;

   private float price;

   [DataMember]
   public float Price{
      get { return price; }
   } }
```

Service Contract

```
[ServiceContract(
  Namespace = "http://www.example.org/Shopping")]
public interface IShoppingCart {
   [OperationContract]
   public float ComputeTax (String state);

   [OperationContract(IsOneWay=true)]
   void Add (Product product, int quantity);

   [OperationContract]
   Uri Checkout (int purchOrder);
}
```

Collection Data Contract

- Collection definition

```
[CollectionDataContract(Name = "MyCollectionOf{0}")]
public class MyCollection<T> : IEnumerable<T> {...}
```

- Service-side contract definition

```
[ServiceContract]
interface IShoppingCart {
  [OperationContract]
  void Add(Product product);

  [OperationContract]
  MyCollection<Product> GetCart();
}
```

89

Collection Data Contract

- Client-side (from metadata):

```
[CollectionDataContract]
public class MyCollectionOfProduct : List<T> { ... }
```

- Client-side contract definition

```
[ServiceContract]
interface IShoppingCart {
  [OperationContract]
  void Add(Product product);

  [OperationContract]
  MyCollectionOfProduct GetCart();
}
```

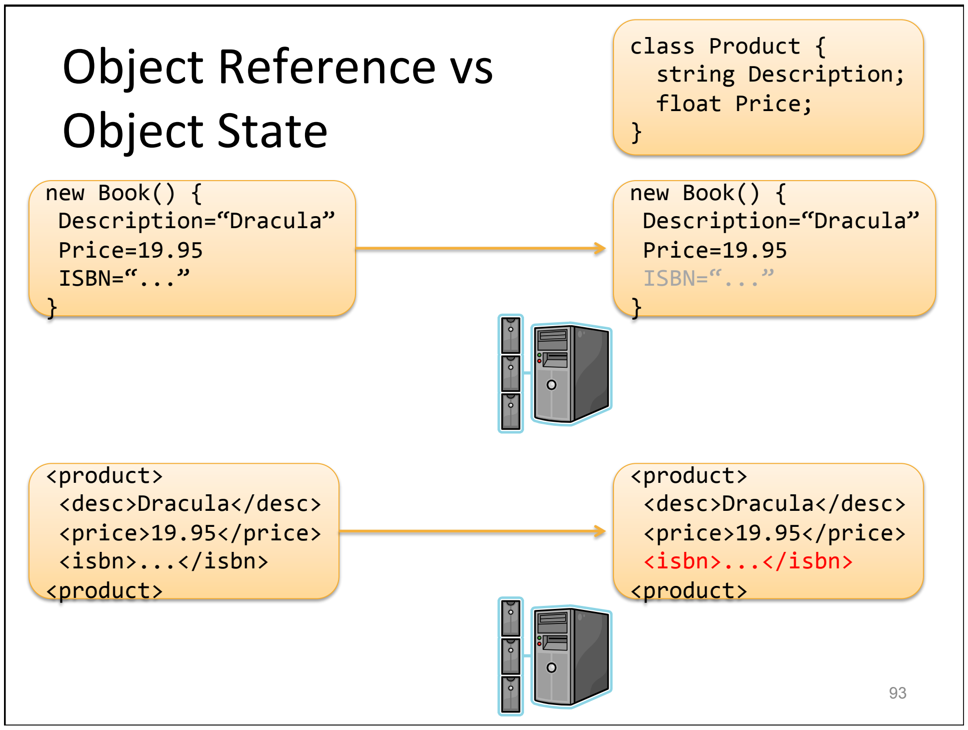
90

SUBCLASSING AND KNOWN TYPES

Contracts and Subclassing

```
[DataContract]
public class Product { ... }

[DataContract]
public class Book : Product {
   [DataMember]
   public string ISBN;
}

Product p = new Product();
client.Add(p,3);    // Okay

Book b = new Book();
client.Add(b,3);    // Fail!
```

Object Reference vs Object State



```
class Product {
  string Description;
  float Price;
}
```

```
new Book() {
 Description="Dracula"
 Price=19.95
 ISBN="..."
}
```

```
new Book() {
 Description="Dracula"
 Price=19.95
 ISBN="..."
}
```

```
<product>
 <desc>Dracula</desc>
 <price>19.95</price>
 <isbn>...</isbn>
<product>
```

```
<product>
 <desc>Dracula</desc>
 <price>19.95</price>
 <isbn>...</isbn>
<product>
```

93

Known Types

```
[DataContract]
[KnownType(typeof(Book))]
public class Product { ... }

[DataContract]
Public class Book : Product {
   [DataMember]
   public string ISBN;
}

Product p = new Product();
Add(p,3);    // Okay

Book b = new Book();
Add(b,3);    // Okay
```

94

Known Types

```
[DataContract]
[KnownType(typeof(Book))]
[KnownType(typeof(DVD))]
public class Product { ... }

[DataContract]
Public class Book : Product {
   [DataMember]
   public string ISBN;
}

[DataContract]
Public class DVD: Product {
   [DataMember]
   public string Format;
}
```

95

Service Known Types

```
[ServiceContract(
  Namespace = "http://www.example.org/Shopping")]
public interface IShoppingCart {
  [OperationContract]
  public float ComputeTax (String state);

  [OperationContract(IsOneWay=true)]
  [ServiceKnownType(typeof(Book))]
  [ServiceKnownType(typeof(DVD))]
  void Add (Product product, int quantity);

  [OperationContract]
  Uri Checkout (int purchOrder);
}
```

96

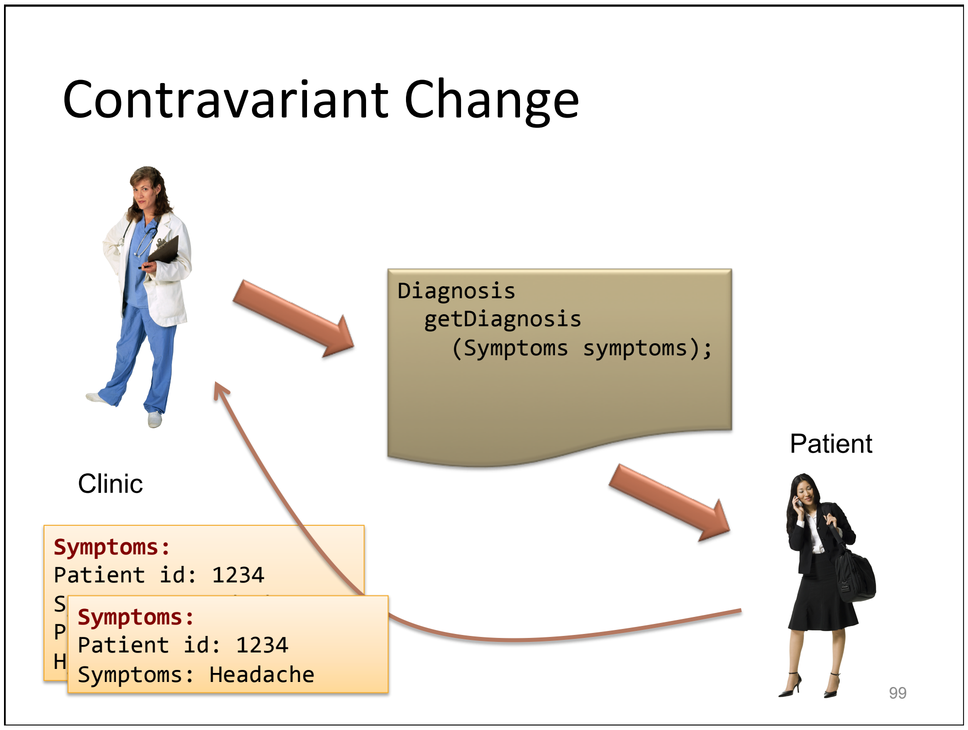
Known Types in Config File

```
<system.runtime.serialization>
  <dataContractSerializer>
    <declaredTypes>
      <add type =
           “Product,MyClassLibrary”>
        <knownType type =
                   “Book,MyOtherClassLibrary”/>
      </add>
    </declaredTypes>
  </dataContractSerializer>
</system.runtime.serialization>
```

97

DATA CONTRACT VERSIONING

98



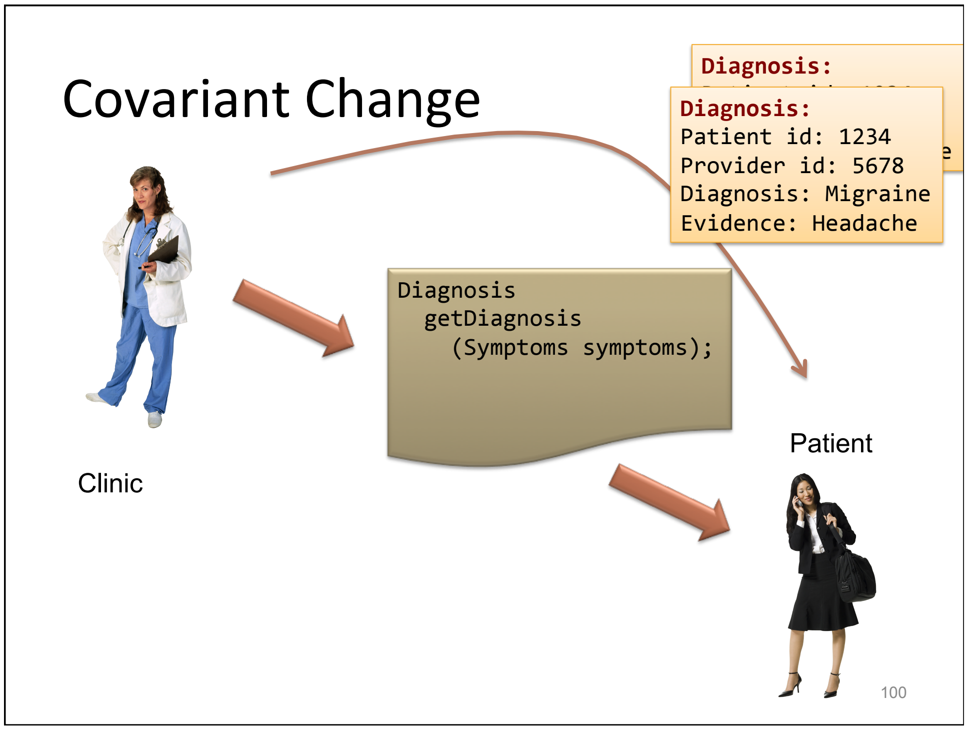
Contravariant Change

```
Diagnosis
  getDiagnosis
    (Symptoms symptoms);
```

Symptoms:
Patient id: 1234
S
P
H

Symptoms:
Patient id: 1234
Symptoms: Headache

99



Covariant Change

```
Diagnosis
  getDiagnosis
    (Symptoms symptoms);
```

Diagnosis:
e

Diagnosis:
Patient id: 1234
Provider id: 5678
Diagnosis: Migraine
Evidence: Headache

100

Data Contract: Compatible Change

- Compatible **output** data model changes (*covariant*)
  - Field extension
- Compatible **input** data model changes (*contravariant*)

- *Forward-compatible* data contracts
  - Implicitly add field to DTO for extra data
  - Extra data reconstituted on return of DTO

101

Contract Versioning

- Strict vs lax versioning
- Lax:
  - Ignore additional fields
  - Missing fields
    - Default values
    - Compensating action
    - Required

102

Missing Members

```
[DataContract(
   Namespace=“http://www.example.org/schemas/Product”)]
public class Product {
   [DataMember]
   public string Description;

   [DataMember]
   private string ASIN;

   [OnDeserializing]
   void OnDeserializing(StreamingContext context) {
      ASIN = “Unspecified ASIN”;
   }
}
```

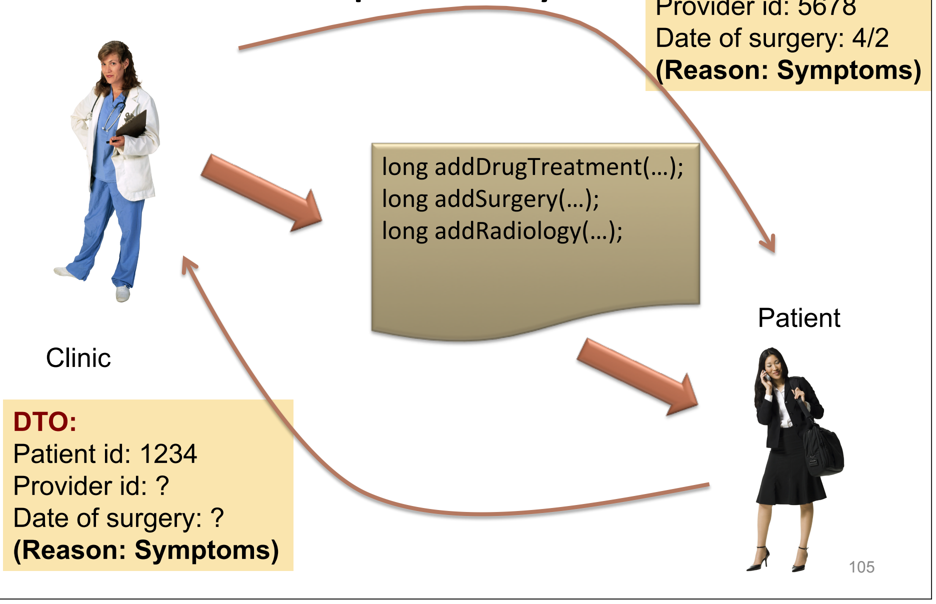
103

Missing Members

```
[DataContract(
   Namespace=“http://www.example.org/schemas/Product”)]
public class Product {
   [DataMember]
   public string Description;

   [DataMember(Required=“true”)]
   private string ASIN;





}
```

104

Forward Compatibility



DTO:
Patient id: 1234
Provider id: 5678
Date of surgery: 4/2
(Reason: Symptoms)

long addDrugTreatment(…);
long addSurgery(…);
long addRadiology(…);

DTO:
Patient id: 1234
Provider id: ?
Date of surgery: ?
(Reason: Symptoms)

105

Round-Trip Versioning

```
[DataContract(
   Namespace="http://www.example.org/schemas/Product")]
public class Product : IExtensibleDataObject {
   [DataMember]
   public string Description;

   ExtensionDataObject
      IExtensibleDatObject.ExtensionData {get;set;}

}
```

106

SERVICE CONTRACT

Service Interface

```
[ServiceContract(Namespace =
    "http://www.example.org/Shopping")]
public interface IShoppingCart {
  [OperationContract]
  public float ComputeTax (String state);

  [OperationContract(IsOneWay=true)]
  void Add (Product product, int amount);

  [OperationContract]
  Uri Checkout (int purchOrder);
}
```

Service Implementation

```
[ServiceBehavior(InstanceContextMode =
  InstanceContextMode.PerSession)]
public class ShoppingCartService : IShoppingCart {
  float tax;
  public float ComputeTax (String state) {
      ... return tax;
  }
  public void Add (Product product, int amount) {
      ...
  }
  public Uri Checkout (int purchOrder) {
      ... return shippingUri;
  }
}
```

Service Binding

```
<system.serviceModel>
  <services>
    <service name="ShoppingCartService">
      <endpoint
        address = "net.tcp://localhost:8090/Shopping"
        binding = "netTcpBinding"
        bindingConfiguration = ""
        contract = "IShoppingCart" />
      <endpoint
        address = "http://localhost:9000/Shopping"
        binding = "wsHttpBinding"
        bindingConfiguration = ""
        contract = "IShoppingCart" />
    </service>
  </services>
</system.serviceModel>
```

Client

```
WSHttpBinding wsBinding = new WSHttpBinding();

wsBinding.SendTimeout = TimeSpan.FromMinutes(5);
wsBinding.TransactionFlow = true;

EndpointAddress addr = new EndpointAddress(...);

ShoppingCartClient client =
  new ShoppingCartClient(wsBinding,addr);

client.Add("Enterprise Architecture",
          39.95);

client.Close();
```

111

CONTRACT METADATA

112

Contract Metadata

```
public class ServiceEndpoint {
  public EndpointAddress Address {get; set;}
  public Binding Binding {get; set;}
  public ContractDescription Contract {get;}
  ...
}

public class ContractDescription {
  public static ContractDescription
                 GetContract(Type contractType);
  public string Name {get; set;}
  public string Namespace {get; set;}
  ...
}
```

Metadata Exchange Client

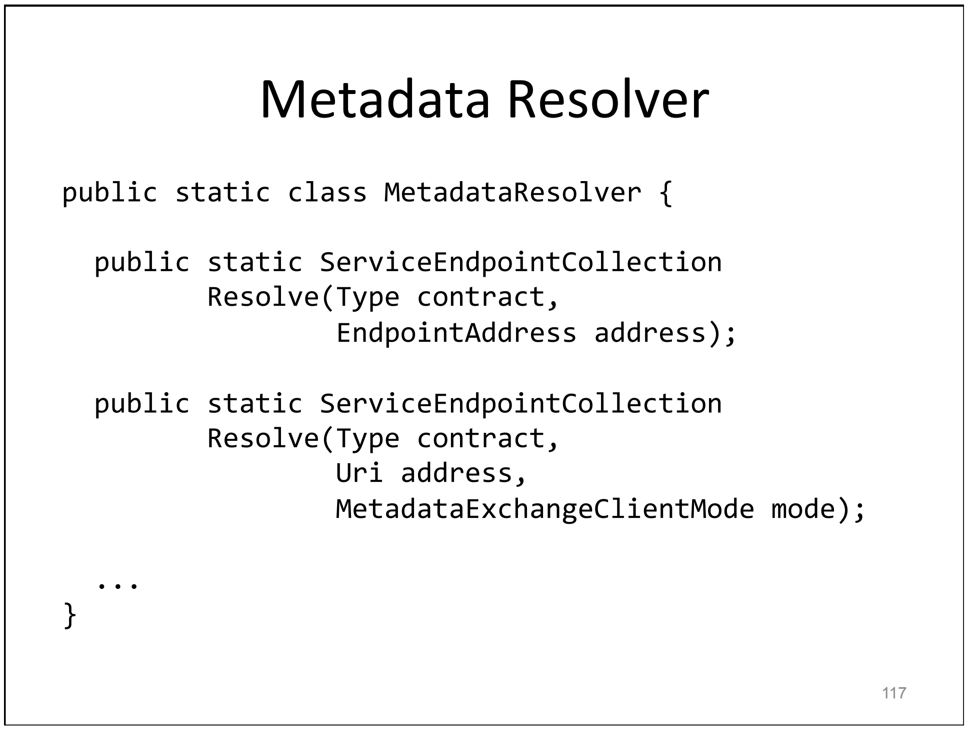
```
public class MetadataSet : ... {...}

public enum MetadataExchangeClientMode {
  MetadataExchange, HttpGet
}

public class MetadataExchangeClient {
  public MetadataExchangeClient
                  (Uri address,
                   MetadataExchangeClientMode mode);
  public MetadataSet GetMetadata();
}
```

Metadata Importer

```
public class ServiceEndpointCollection :
               Collection<ServiceEndpoint> {...}

public abstract class MetadataImporter {
  public abstract ServiceEndpointCollection
    ImportAllEndpoints();
  ...
}

public class WsdlImporter : MetadataImporter {
  public WsdlImporter(MetadataSet metadata);
  ...
}
```

Querying Metadata

```
Uri mexAddress = new Uri("...? wsdl");

MetadataExchangeClient mexClient =
  new MetadataExchangeClient(mexAddress,
                MetadataExchangeClientMode.HttpGet);

MetadataSet metadata = mexClient.GetMetadata();

MetadataImporter importer =
                       new WsdlImporter(metadata);

ServiceEndpointCollection endpoints =
                       importer.ImportAllEndpoints();
```



Metadata Resolver

```
public static class MetadataResolver {

  public static ServiceEndpointCollection
         Resolve(Type contract,
                 EndpointAddress address);

  public static ServiceEndpointCollection
         Resolve(Type contract,
                 Uri address,
                 MetadataExchangeClientMode mode);

  ...
}
```

117