




J-Spring 2007

Web Service mediation with Synapse

Tijs Rademakers



Apache Synapse



Whoami

- 🚀 Tijs Rademakers
- 🚀 Software Architect at Atos Origin
- 🚀 Focus on J2EE and integration
 - ☪ J2EE architecture in general
 - ☪ Web Services – WS-*
 - ☪ ESB and SOA technology – WebSphere Process Server, Mule, ServiceMix, Synapse
- 🚀 Committer of the Apache Synapse project
- 🚀 For questions and remarks: tijs@apache.org



Agenda

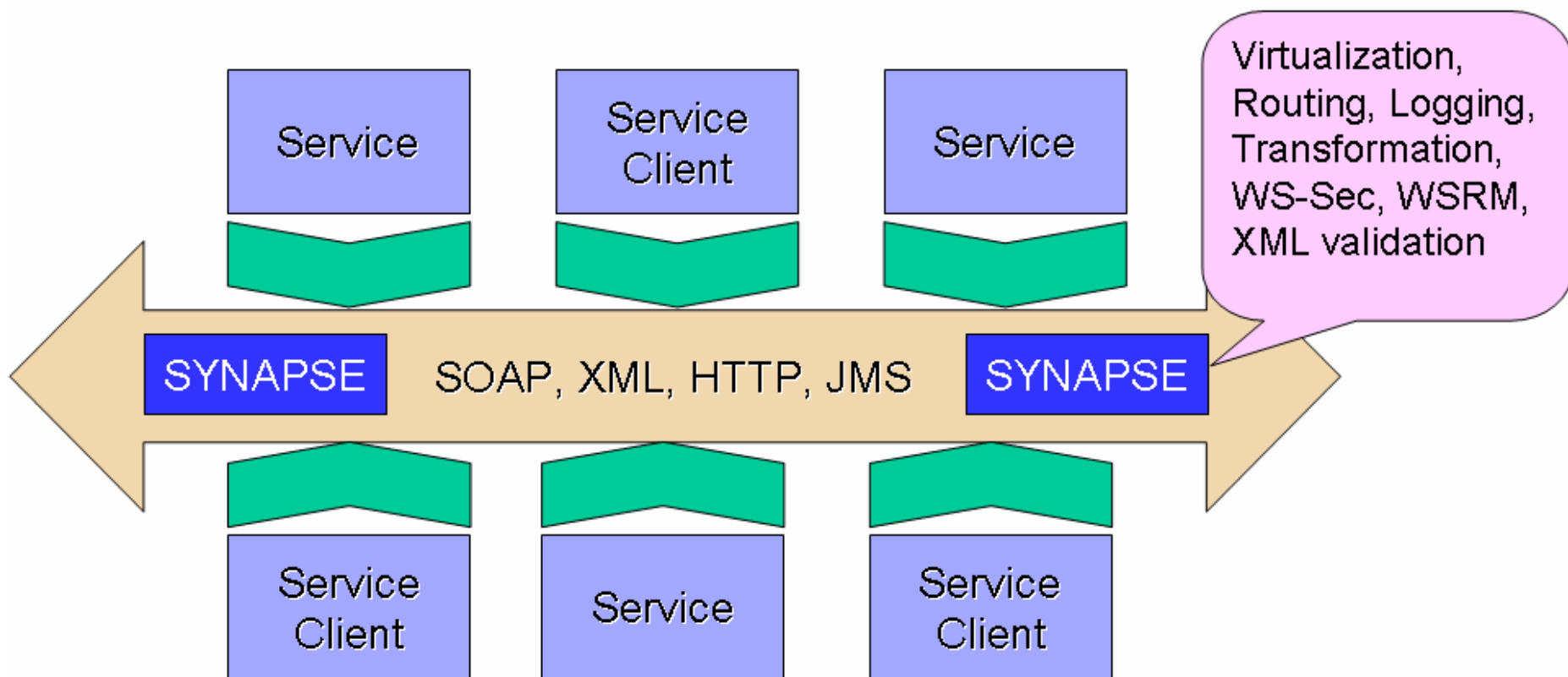
- 🐱 Introduction into Synapse
 - ☛ What is Synapse about?
 - ☛ Synapse architecture
 - ☛ Synapse Configuration Language
- 🐱 Demos with Synapse
 - ☛ Content-based routing demo
 - ☛ Message validation demo
 - ☛ Transport switching (JMS to HTTP) demo
 - ☛ Reliable Messaging (JMS to WS-RM) demo
- 🐱 Further investigation + Questions



Web services problem area

- 🚀 Client dependent on Web Service endpoint
- 🚀 How to deal with integration challenges?
 - ☕ Content-based routing
 - ☕ Message validation
- 🚀 How to support protocol translation?
 - ☕ JMS to Web service
- 🚀 How to implement web services reliability?
 - ☕ WS-Reliable Messaging

What is Apache Synapse?



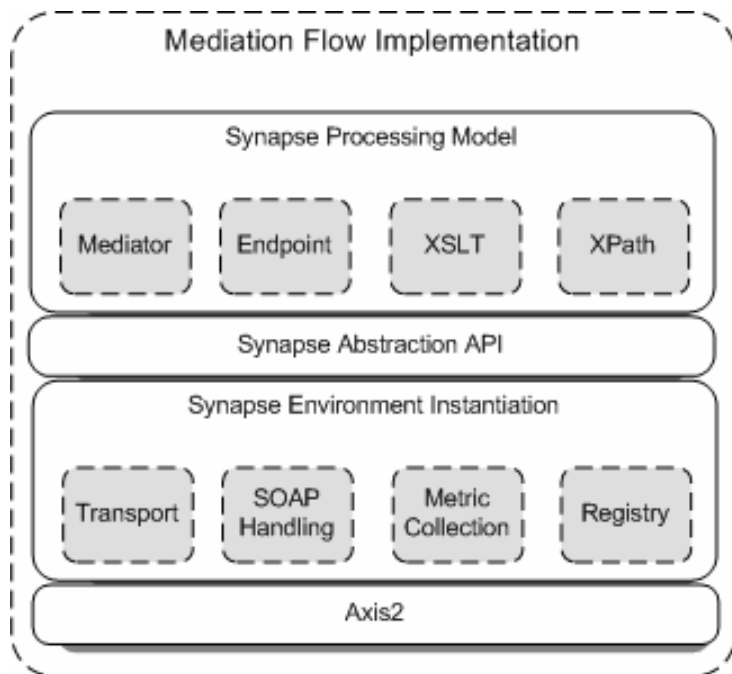


Overview of the Apache Synapse project

- 🐾 Part of the Apache Web Services project
 - ☛ Open source donation → X-Broker product (Infravio)
 - ☛ Promoted out of incubation in January 2007
 - ☛ 1.0 release June 11, 2007
 - ☛ <http://ws.apache.org/synapse>
- 🐾 Web services mediation framework (ESB-like)
 - ☛ Routing
 - ☛ Transformation
 - ☛ Validation
 - ☛ Registry / Management



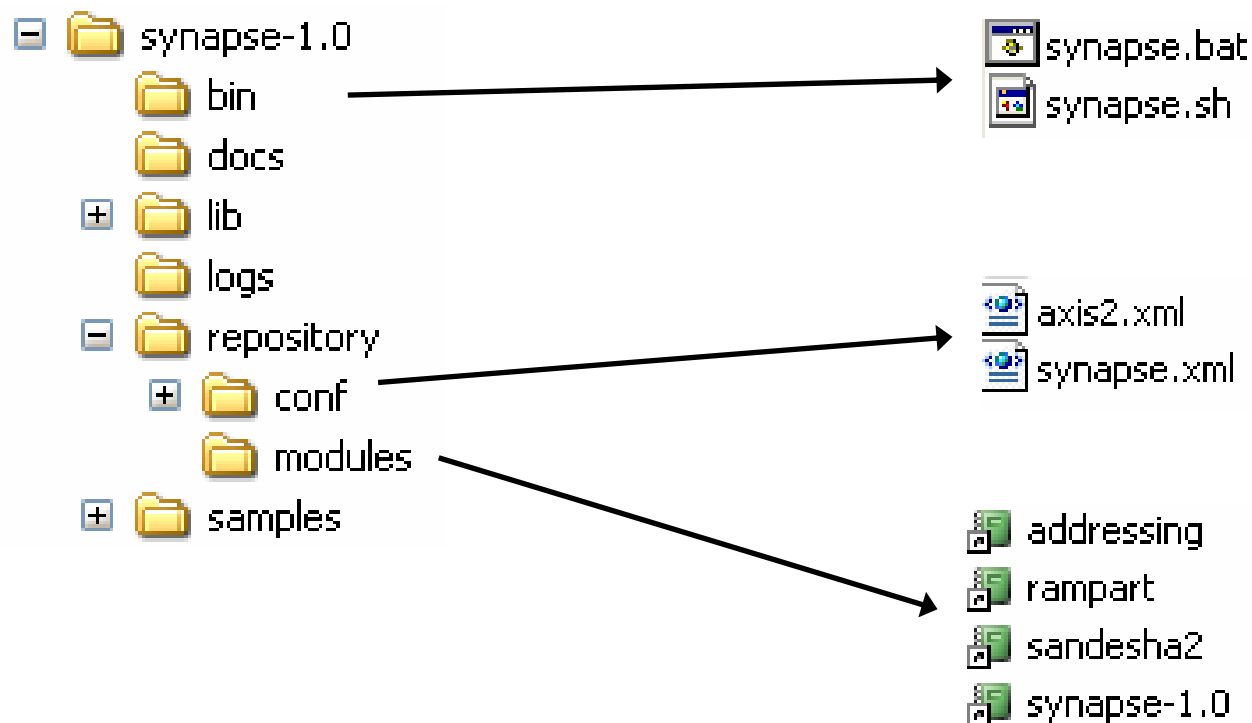
Apache Synapse architecture



- 🦋 Built on Axis2:
 - ☛ AXIOM model
 - ☛ Standards support:
 - SOAP, WSDL, XSD
 - ☛ Wide variety of WS-* support:
 - WS-Addressing
 - WS-Security
 - WS-Reliable Messaging
- 🦋 Flexible model with mediators
- 🦋 XML based configuration



Synapse distribution





Synapse Configuration Language

```
<definitions xmlns="http://ws.apache.org/ns/synapse">
```

```
  <registry provider="provider">...</registry>
```

```
  <sequence name="string">...</sequence>
```

```
  <endpoint name="string">...</endpoint>
```

























```
  <proxy name="string" ...>...</proxy>
```

```
  mediator*
```

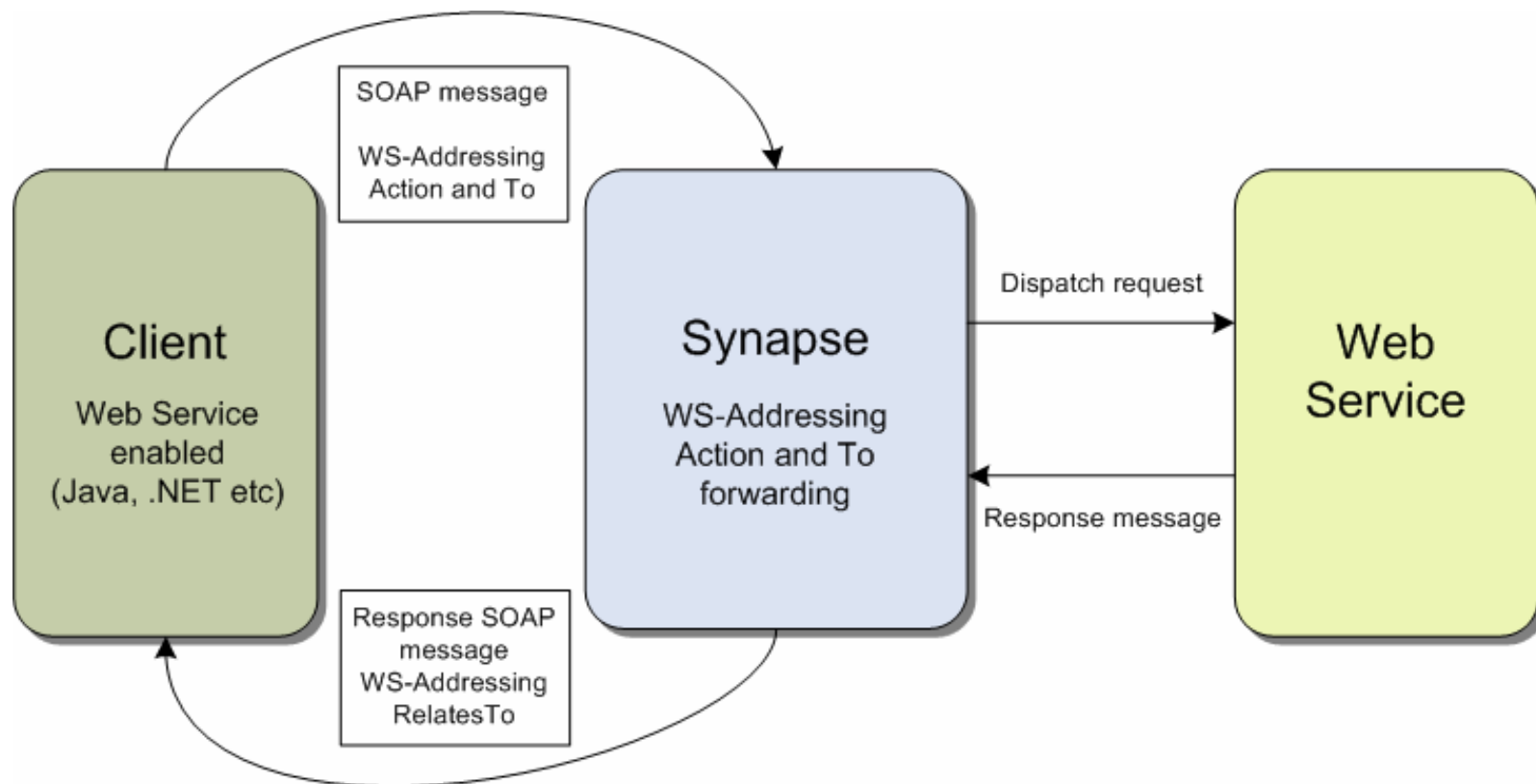
```
</definitions>
```



Overview of Synapse mediators

-  In:
 -  *Incoming messages*
-  Out:
 -  *Outgoing messages*
-  Send:
 -  *Send on*
-  Drop:
 -  *Ditch the message*
-  Log:
 -  *Log the message*
-  Validate:
 -  *XSD validation*
-  Transform:
 -  *XSLT*
-  Makefault:
 -  *Sends a fault on or back*
-  Header:
 -  *Sets a header in the message*
-  Filter:
 -  *Conditional processing (if)*
-  Switch:
 -  *Conditional processing (case)*
-  Class:
 -  *Your own mediator*

Synapse simple example (graphically)





Synapse simple example

```
<definitions xmlns="http://ws.apache.org/ns/synapse">  
  <in>  
    <!-- Log all messages passing through -->  
    <log level="full"/>  
    <!-- Send the messages to implicit "To" endpoint -->  
    <send/>  
  </in>  
  <out>  
    <send/>  
  </out>  
</definitions>
```



Typical WS-Addressing usage

Request to TestService

```
<wsa:MessageID>urn:uuid:8B3F32C2ABD</wsa:MessageID>
```

```
<wsa:To>http://localhost:8080/test/TestService</wsa:To>
```

```
<wsa:ReplyTo>http://localhost:9090/reply/ReplyService</wsa:ReplyTo>
```

```
<wsa:Action>http://test</wsa:Action>
```

Response to ReplyService

```
<wsa:MessageID>urn:uuid:CDC1D02A252</wsa:MessageID>
```

```
<wsa:RelatesTo>urn:uuid:8B3F32C2ABD</wsa:RelatesTo>
```

```
<wsa:Action>http://test</wsa:Action>
```



Built-in support for WS-Addressing

- 🐎 MessageID header is automatically generated
- 🐎 To and Action headers used for actual service request
 - 🍎 To header used for endpoint address
 - 🍎 Action header to determine which web service method to call
- 🐎 ReplyTo is used as response address
- 🐎 RelatesTo is automatically generated
- 🐎 Support for header manipulation

```
<header name="Action" value="urn:youraction"/>
```

```
<header name="To" value="get-property('FaultTo')"/>
```



Agenda

- 🐱 Introduction into Synapse
 - ☛ What is Synapse about?
 - ☛ Synapse architecture
 - ☛ Synapse Configuration Language
- 🐱 Demos with Synapse
 - ☛ Content-based routing demo
 - ☛ Message validation demo
 - ☛ Transport switching (JMS to HTTP) demo
 - ☛ Reliable Messaging (JMS to WS-RM) demo
- 🐱 Further investigation + Questions



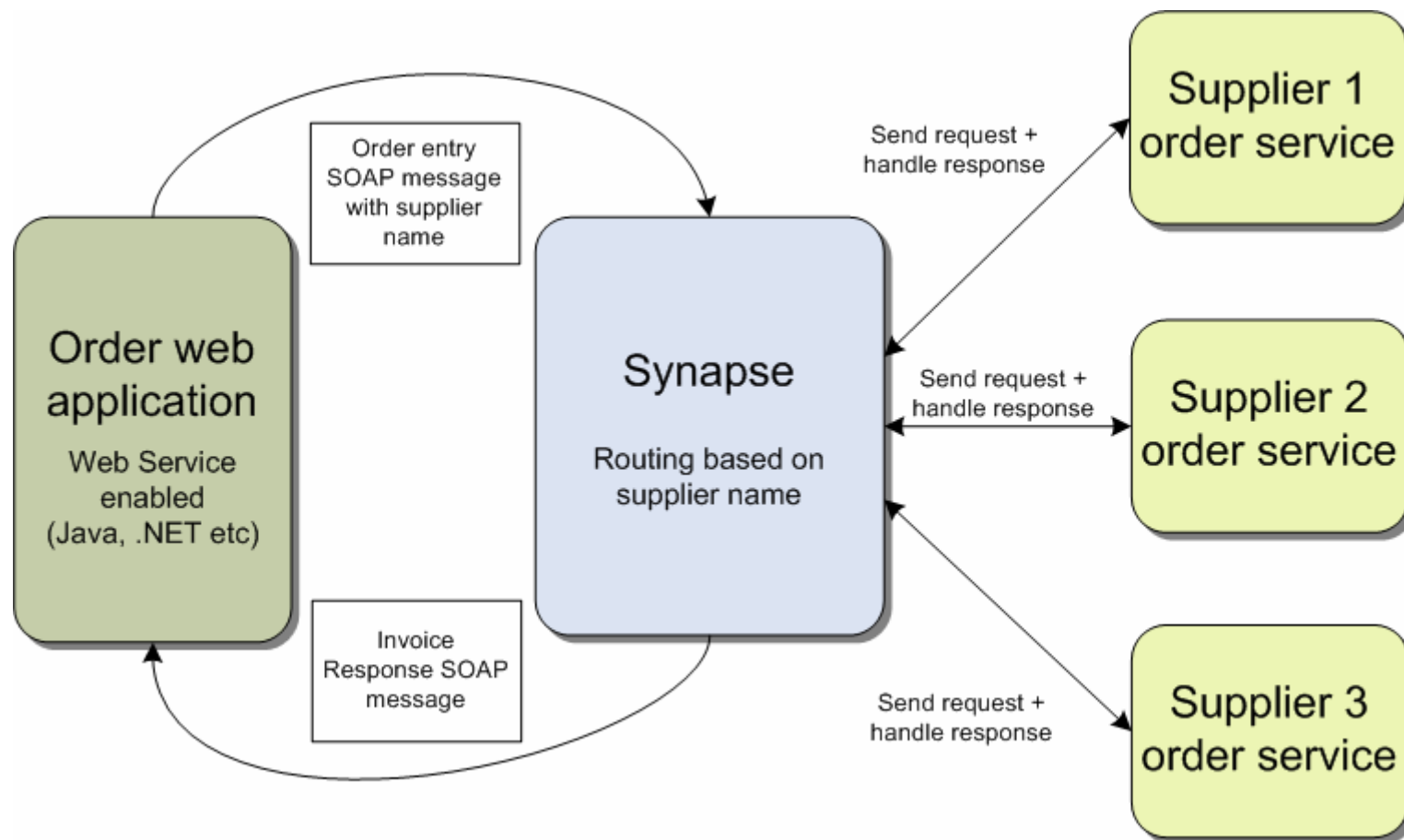
Content-based routing

- 🐎 Determine destination based on message content
- 🐎 Not limited to the SOAP message itself
- 🐎 Two options

```
<filter xpath="xpath"> <!--Only apply mediator if filter matches -->  
  <mediator..>  
</filter>
```

```
<switch source="xpath"> <!-- only one will execute -->  
  <case regex="string">...</case>  
  <default>...</default>  
</switch>
```

Content-based routing with Synapse





Order entry message

```
<order xmlns="http://atosorigin.com/nljug" orderId="923809">  
  <clientName>clientA</clientName>  
  <productNumber>12345</productNumber>  
  <quantity>10</quantity>  
  <supplierName>Supplier1</supplierName>  
</order>
```

🐱 Routing based on the supplierName element !!



Synapse routing configuration

```
<definitions xmlns="http://ws.apache.org/ns/synapse">  
  <endpoint name="supplier1Endpoint">  
    <address uri="http://localhost:7070/OrderServiceSupplier1">  
      <enableAddressing/>  
    </address>  
  </endpoint>  
  <endpoint name="supplier2Endpoint">  
    <address uri="http://localhost:7070/OrderServiceSupplier2">  
      <enableAddressing/>  
    </address>  
  </endpoint>  
  .....
```



Synapse routing configuration (2)

```
<in>
  <switch source="//jug:supplierName"
    xmlns:jug="http://atosorigin.com/nljug">
    <case regex="Supplier1">
      <header name="Action"
        value="http://supplier1.com/placeorder"/>
      <send>
        <endpoint key="supplier1Endpoint"/>
      </send>
    </case>
    ...
  </switch>
</in>
.....
```



DEMO



Synapse registry support

- ✈ Support to share configurations across multiple Synapse instances
- ✈ Introduce dynamic configuration to Synapse
- ✈ URL based registry is built-in
 - ☛ File system, HTTP
- ✈ Can be used for
 - ☛ Sequence definitions
 - ☛ Endpoint definitions
 - ☛ Schemas
 - ☛ XSLTs



Synapse registry configuration

```
<definitions xmlns="http://ws.apache.org/ns/synapse">  
  
<registry  
  provider="org.apache.synapse.registry.url.SimpleURLRegistry">  
  <parameter name="root">file:./repository/conf</parameter>  
  <parameter name="cachableDuration">15000</parameter>  
</registry>
```

.....



Validation of incoming messages

- 🐞 XML Schema validation
- 🐞 Validate incoming messages
- 🐞 Report validation error back to client

```
<xsd:element name="order" type="jug:OrderType"/>
<xsd:complexType name="OrderType">
  <xsd:sequence>
    <xsd:element name="clientName" type="xsd:string"/>
    <xsd:element name="productNumber" type="xsd:string"/>
    <xsd:element name="quantity" type="xsd:int"/>
    <xsd:element name="supplierName" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="orderId" type="xsd:string"/>
</xsd:complexType>
```



Synapse validation configuration

```
<validate>
  <schema key="orderxsd.xml" source="//jug:order"
    xmlns:jug="http://atosorigin.com/nljug"/>
  <on-fail>
    <makefault>
      <code value="tns:Receiver"
        xmlns:tns="http://www.w3.org/2003/05/soap-envelope"/>
      <reason value="Invalid order request"/>
    </makefault>
    <property name="RESPONSE" value="true"/>
    <header name="To" expression="get-property('ReplyTo')"/>
  </on-fail>
</validate>
```



DEMO



Reliable transport with Synapse

- 🐘 HTTP connection is unreliable (demo)
- 🐘 JMS can help with this
- 🐘 Define JMS proxy in Synapse configuration

```
<proxy name="OrderJMS" transports="jms">  
  <target inSequence="orderIn" outSequence="orderOut"/>  
</proxy>
```

- 🐘 Listens on OrderJMS queue for incoming messages

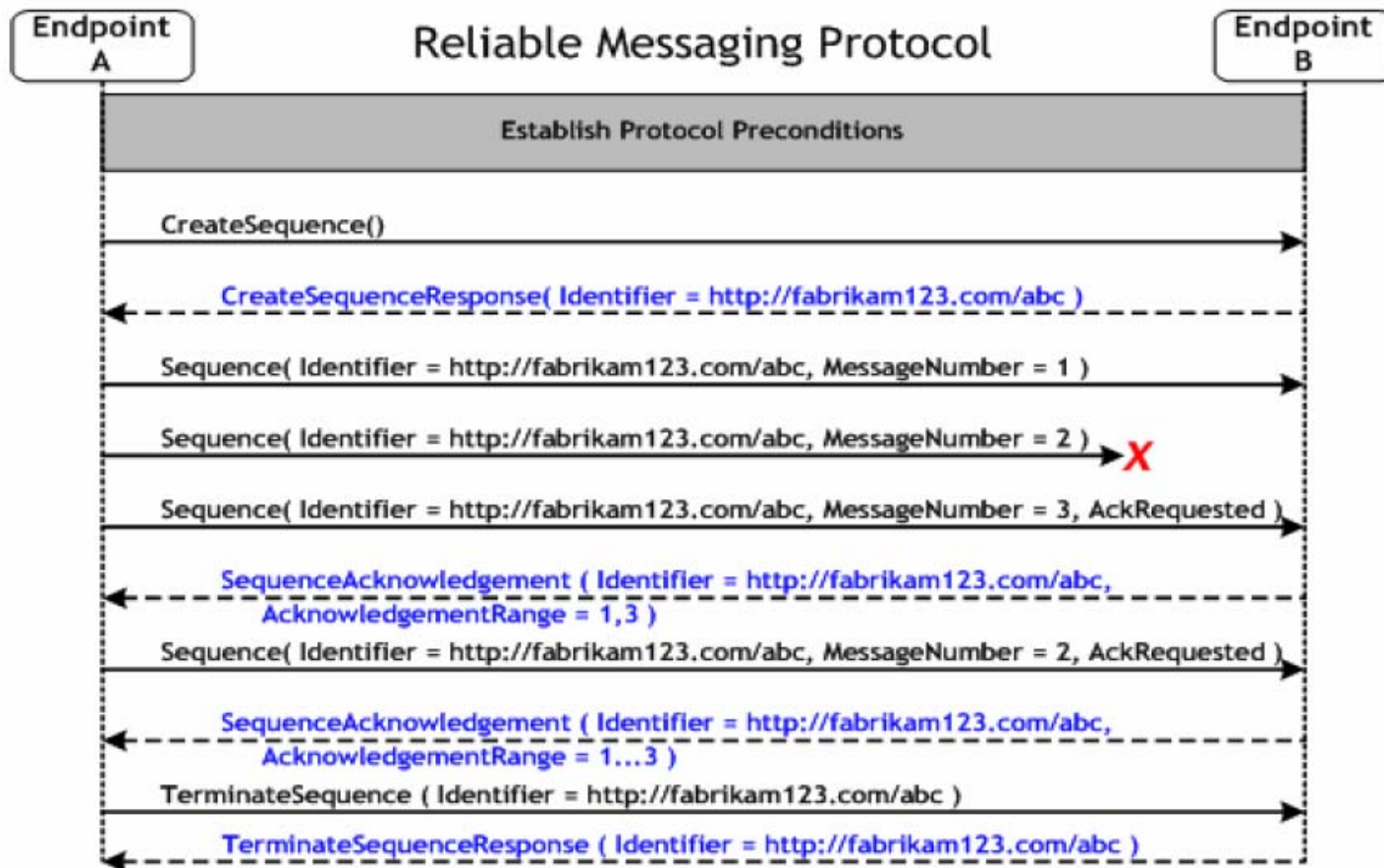


DEMO

WS-RM communication with Synapse

- ⚡ HTTP connection from Synapse to web service is also unreliable
- ⚡ We could use a JMS transport
- ⚡ But Synapse also supports WS-Reliable Messaging

WS-Reliable Messaging





Configure WS-RM in Synapse

Endpoint configuration

```
<endpoint name="?" address="?">  
  <enableRM/>  
</endpoint>
```

Extra parameters

```
<RMSequence (correlation="xpath" [last-message="xpath"]) |  
  single="true" [version="1.0|1.1"]/>
```

For the example just one message in sequence

```
<RMSequence single="true" version="1.0"/>
```



DEMO



Agenda

- 🐼 Introduction into Synapse
 - ☛ What is Synapse about?
 - ☛ Synapse architecture
 - ☛ Synapse Configuration Language
- 🐼 Demos with Synapse
 - ☛ Content-based routing demo
 - ☛ Message validation demo
 - ☛ Transport switching (JMS to HTTP) demo
 - ☛ Reliable Messaging (JMS to WS-RM) demo
- 🐼 Further investigation + Questions



Summary

- ✈ Synapse can help with web services complexity
 - ☛ Enables web service support in your application
 - ☛ WS-* support
- ✈ Flexible architecture
- ✈ Provides mediation functionality
- ✈ Please provide feedback on Synapse mailinglist



Look at the Apache Synapse samples

Running the Synapse Samples

Contents

- [Overview](#)
- [Message mediation samples](#)
 - [Sample 0: Introduction to Synapse](#)
 - [Sample 1: Content based routing \(CBR\)](#)
 - [Sample 2: CBR with the Switch-case mediator, using message properties](#)
 - [Sample 3: Local Registry entry definitions, reusable endpoints and sequences](#)
 - [Sample 4: Introduction to error handling](#)
 - [Sample 5: Creating SOAP fault messages and changing the direction of a message](#)
 - [Sample 6: Manipulating SOAP headers, and filtering incoming and outgoing messages](#)
 - [Sample 7: Introduction to local Registry entries and using Schema validation](#)
 - [Sample 8: Introduction to static and dynamic registry resources, and using XSLT transformations](#)
 - [Sample 9: Introduction to dynamic sequences with the Registry](#)
 - [Sample 10: Introduction to dynamic endpoints with the Registry](#)
 - [Sample 11: A full registry based configuration, and sharing a configuration between multiple instances](#)
- [Endpoints](#)
 - [Sample 50: Using WS-Security for outgoing messages](#)
 - [Sample 51: MTOM and SwA optimizations and request/response correlation](#)
 - [Sample 52: POX to SOAP conversion](#)
 - [Sample 53: Reliable message exchange between Synapse and the back-end server using WS-ReliableMessaging](#)
 - [Sample 54: Load balancing](#)
 - [Sample 55: Failover](#)



Interesting links

🚀 Synapse website <http://ws.apache.org/synapse>

- ☪ Download Synapse
- ☪ Look at developer discussion forum
- ☪ Participate!



🚀 WSO2 Oxygen website <http://www.wso2.org>

- ☪ Interesting articles on Axis2, WS-* and Synapse
- ☪ Open Source products based on Synapse





Questions?

