

Axis2, WS-Specifications and Synapse

How to use Axis2 and synapse to implement Enterprise Integration Patterns

Agenda

- Introduction 2 min.
- **Axis2 overview** 10 min.
- WS-* support from Axis2 20 min.
- EIP with Synapse 13 min.
- Questions 5 min.





What is Axis2

- Apache Soap → Axis 1.x → Axis2
- Started in Aug. 2004, 1.0 in May 2006
- Reasons Axis2 was started:
 - Performance
 - Axis2 is 4-5 times more performant
 - Asynchronicity
 - Axis was based on request/reply
 - Architecture
 - Better way to handle extensions
 - Simple deployment model for service



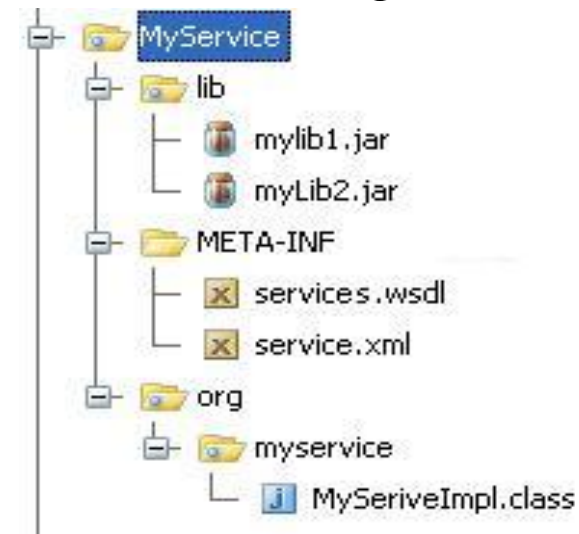
Axis2 Features

- Based on Axiom (StAX based)
 - lazy parsing
- Asynchronous / message based model
- Easy to extend (modules and phases)
- Web service versioning, hot deployment
- Flexible databinding:
 - XMLBeans, JAXB, JibX, ADB
- MTOM support
- REST support
- Support for many WS-*



Axis2 Concepts

- Repository: contains all the resources
 - Directory on the FS / can also be an URL
 - Contains modules and services
- Modules: used to extend Axis2
 - WS-Addressing, WS-Security, WS-Eventing etc.
 - Not hot-deployable
- Services: contains your web services
 - Deployment descriptor
 - Service class and libs
 - Can be “hot deployed”

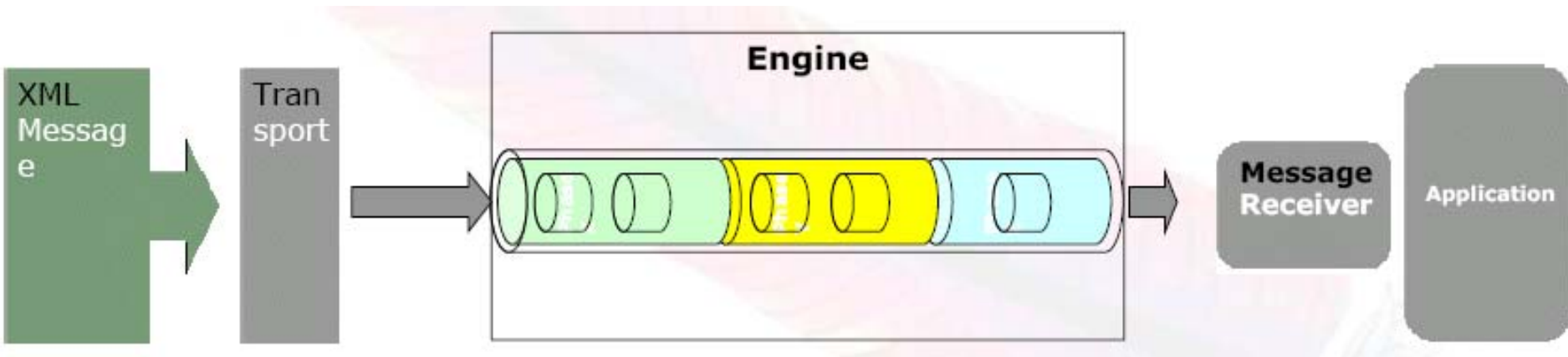


Axis2: services.xml

```
<!-- An example webservice -->  
<service name="exampleService">  
  <parameter name="ServiceClass">  
    nljug.demo1.ExampleService  
  </parameter>  
  <operation name="calculate">  
    <messageReceiver  
class="org.apache.axis2.receivers.RawXMLINOutMessageReceiver" />  
    <actionMapping>urn:doSomething</actionMapping>  
  </operation>  
</service>
```

- Things to note:
 - messageReceiver class
 - actionMapping

Axis2 Architecture



- Three main phases:
 - Transport receiver: transport related processing
 - Dispatching: finding service and operation
 - Message Receiver: invoking the service

Axis2 Demo

- Raw XML The 'preferred' way
- Bottom up: Java2WSDL
- Top down: WSDL2Java
 - Uses the wsdl from step 2

DEMO

Agenda

- Introduction 2 min.
- Axis2 overview 10 min.
- **WS-* support from Axis2** 20 min.
- EIP with Synapse 13 min.
- Questions 5 min.



WS-Specification overview

Interoperability Issues

Basic Profile
1.1 WS-2002-02
Final Specification

Attachments Profile
1.1 WS-2002-02
Final Specification

Simple SOAP Binding Profile
1.1 WS-2002-02
Final Specification

Basic Security Profile
1.0 Working Draft

BEL Token Profile
1.0 Working Draft

SAML Token Profile
1.0 Working Draft

Conformance Claim Attachment Mechanism (CCAM)
1.0 WS-2002-02
Final Specification

Reliable Anonymous Messaging Profile (RAM)
1.0 WS-2002-02
Final Specification

Business Process Specifications

Business Process Execution Language for Web Services (BPEL4WS)
1.1 WS-2003-03
March 2003
Final Specification

Business Process Markup Language (BPML)
1.0 WS-2002-02
Final Draft

Web Service Choreography Model Overview
1.0 WS-2002-02
Working Draft

Web Service Choreography Description Language (WSDL-CD)
1.0 WS-2002-02
Working Draft

Web Service Choreography Interface (WSCI)
1.0 WS-2002-02
Final Specification

Management Specifications

Web Services Management Framework
1.0 WS-2002-02
Final Specification

WS-Management
1.0 WS-2002-02
Final Specification

WS-Events
1.0 WS-2002-02
Working Draft

Management Using Web Services (MUM)
1.0 WS-2002-02
Working Draft

Web Services Management
1.0 WS-2002-02
Working Draft

Management of Web Services (MOWS)
1.0 WS-2002-02
Working Draft

Metadata Specifications

WS-Policy
1.0 WS-2002-02
Final Specification

WS-PolicyAssertions
1.0 WS-2002-02
Final Specification

WS-PolicyAttachment
1.0 WS-2002-02
Final Specification

WS-Discovery
1.0 WS-2002-02
Final Specification

WS-MetadataExchange
1.0 WS-2002-02
Final Specification

Universal Description, Discovery and Invocation (UDDI)
2.0 WS-2002-02
Final Specification

Web Service Description Language (WSDL)
2.0 WS-2002-02
Final Specification

Web Service Description Language (WSDL)
2.0 WS-2002-02
Final Specification

Reliability Specifications

WS-ReliableMessaging
1.0 WS-2002-02
Final Specification

WS-Reliability
1.0 WS-2002-02
Final Specification

Security Specifications

WS-Security
1.0 WS-2002-02
Final Specification

WS-Security: SOAP Message Security
1.0 WS-2002-02
Final Specification

WS-Security: Kerberos Binding
1.0 WS-2002-02
Final Specification

WS-Security: SAML Token Profile
1.0 WS-2002-02
Final Specification

WS-Security: X.509 Certificate Token Profile
1.0 WS-2002-02
Final Specification

WS-Security: Username Token Profile
1.0 WS-2002-02
Final Specification

WS-SecurityPolicy
1.0 WS-2002-02
Final Specification

WS-Trust
1.0 WS-2002-02
Final Specification

WS-Federation
1.0 WS-2002-02
Final Specification

WS-Security Conversation
1.0 WS-2002-02
Final Specification

Transaction Specifications

WS-Business Activity
1.0 WS-2002-02
Final Specification

WS-Atomic Transaction
1.0 WS-2002-02
Final Specification

WS-Coordination
1.0 WS-2002-02
Final Specification

WS-Composable Applications Framework
1.0 WS-2002-02
Final Specification

WS-Context
1.0 WS-2002-02
Final Specification

WS-Coordination Framework
1.0 WS-2002-02
Final Specification

WS-Transaction Management
1.0 WS-2002-02
Final Specification

Resource Specifications

Web Services Resource Framework
1.0 WS-2002-02
Final Specification

WS-QueueFaults
1.0 WS-2002-02
Final Specification

WS-ServiceGroup
1.0 WS-2002-02
Final Specification

WS-ResourceProperties
1.0 WS-2002-02
Final Specification

WS-ResourceRelief
1.0 WS-2002-02
Final Specification

WS-Transfer
1.0 WS-2002-02
Final Specification

Resource SOAP Representation (RSR)
1.0 WS-2002-02
Final Specification

Messaging Specifications

WS-Messaging
1.0 WS-2002-02
Final Specification

WS-Addressing
1.0 WS-2002-02
Final Specification

WS-Eventing
1.0 WS-2002-02
Final Specification

WS-EventNotification
1.0 WS-2002-02
Final Specification

WS-Topics
1.0 WS-2002-02
Final Specification

WS-Streams
1.0 WS-2002-02
Final Specification

WS-BrokeredNotification
1.0 WS-2002-02
Final Specification

SOAP 1.2 WS-2003-05

SOAP 1.1 WS-2003-05

SOAP Message Transport Protocol (SMTP)
1.0 WS-2002-02
Final Specification

WS-Spec support from Axis2

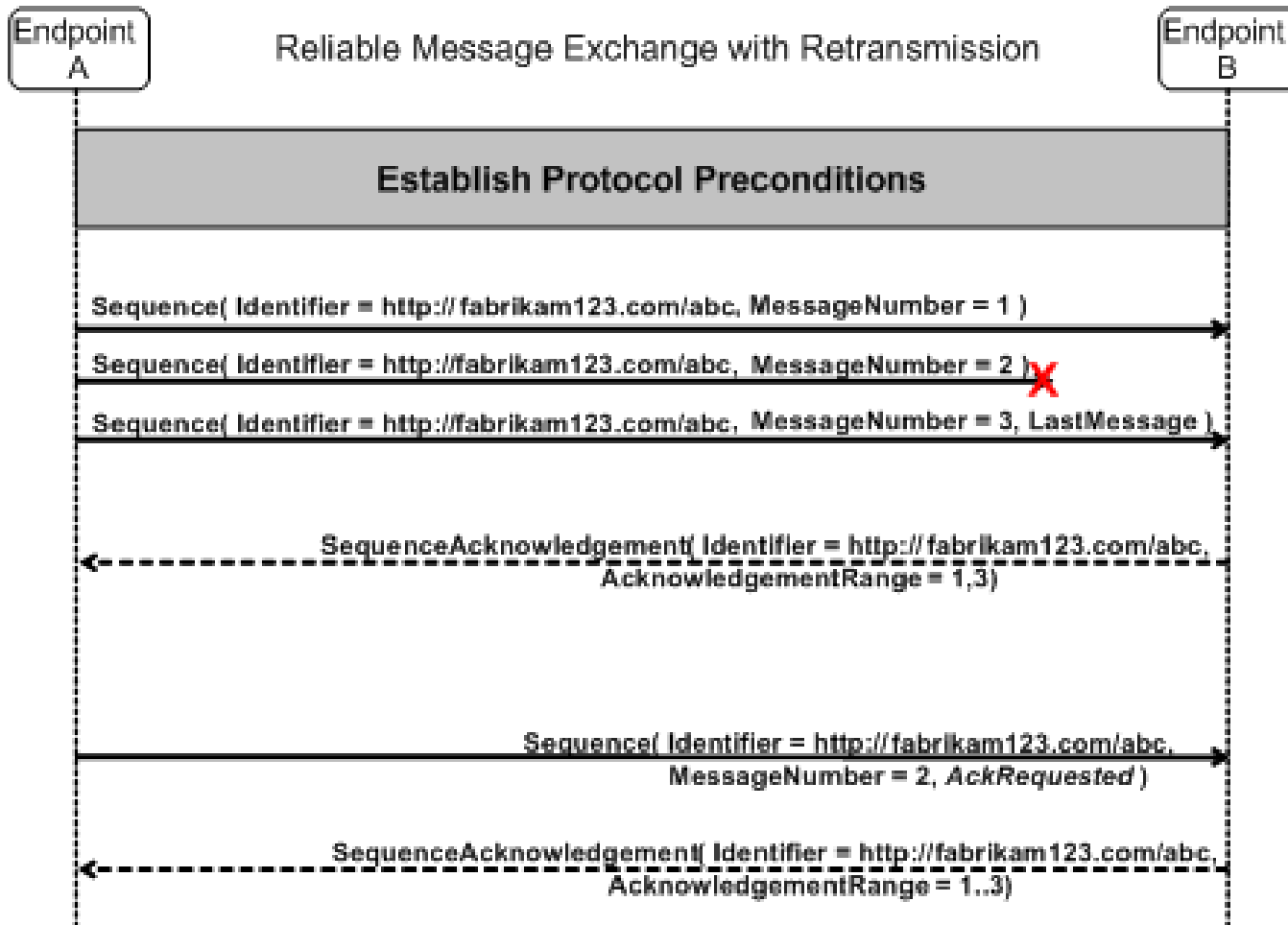
- Axis2 currently supports the following:
 - WS-Addressing
 - WS-Reliable messaging
 - WS-Coordination
 - WS-AtomicTransaction
 - WS-Security
 - WS-Eventing / WS-Notification
 - core
 - Sandesha2
 - Kandula2
 - Kandula2
 - Rampart
 - Savaan



WS-ReliableMessaging

- Request-Response is not reliable
 - You know when it fails, but where
- Use a different transport
 - e.g. use SOAP over JMS
- WS-ReliableMessaging tries to:
 - Ensure that messages are delivered to their destination.
 - “Exactly Once In Order”, most common requirement

WS-ReliableMessaging exchange



WS-ReliableMessaging Demo

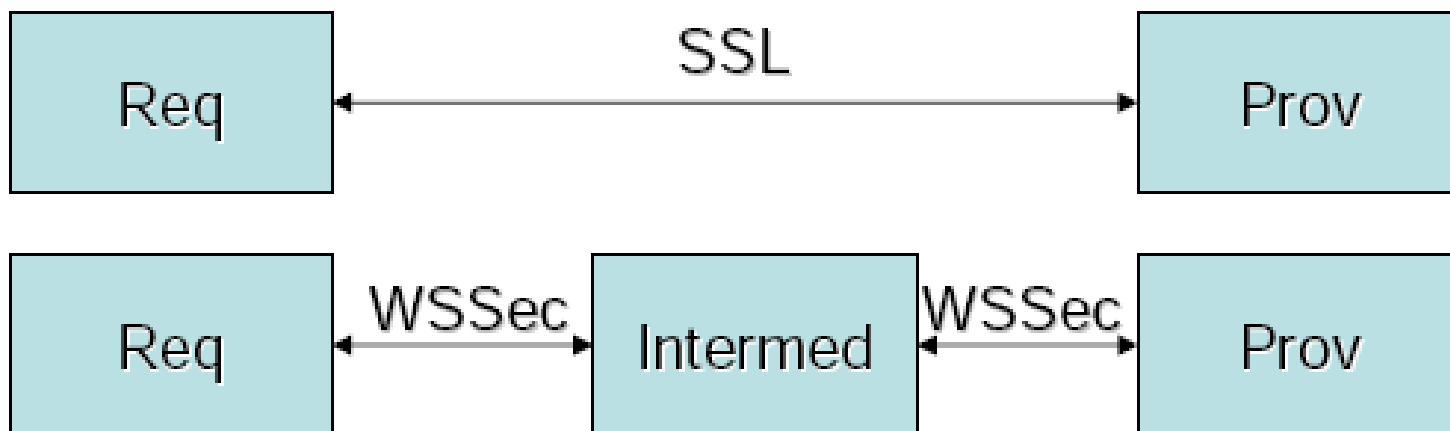
1. Send message 1 to target webservice
2. Send message 2 to target webservice
3. Send message 3 (the last message) to target webservice
4. Retrieve a Sandesha report

A SandeshaReport include following information:

- * Sequence IDs of all the outgoing sequences.
- * Number of completed messages of each outgoing sequences.
- * Sequence IDs of all the incoming sequences.
- * No of completed messages of each incoming sequence.

WS-Security

- WS-Security provides:
 - Encryption, signatures and authentication
- Why WS-Security when we've got SOAP over SSL/TLS?
 - Doesn't work for multi-hop



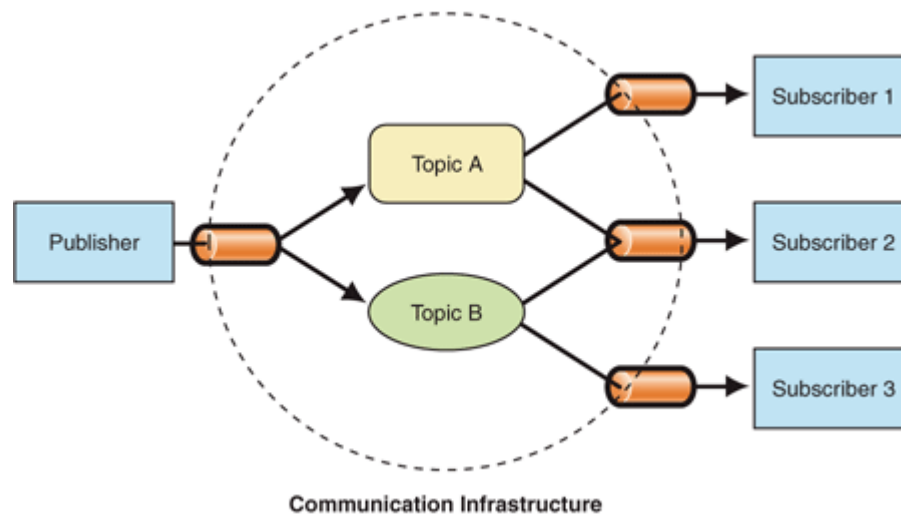
WS-Security Demo

1. Configure WS-Security to sign the message and add a timestamp
2. Send a signed message
3. Send a signed message that will create an error
4. Route the response to a separate web service
5. And show that hot-deploy / hot-update works

Step 4 makes use of one of WS-Addressing headers: `wsa:FaultTo`. If a SOAP fault occurs it's send to the specified address.

WS-Eventing

- Allows webservice to subscribe themselves to receive certain events.
- Comparable to the JMS topic.
- Allows for filters
- Specify subscription timeout



WS-Eventing Demo

1. Create a web service that receives and manages the WS-Eventing requests.
2. Register three webservicess as subscriber to the previous one.
3. Send a message to the webservice created in the first request.
4. Specify that the response of the webservice calls are send back to the client.

Agenda

- Introduction 2 min.
- Axis2 overview 10 min.
- WS-* support from Axis2 20 min.
- **EIP with Synapse** 13 min.
- Questions 5 min.



What is Synapse?

- Synapse is an ESB for webservices.
- Synapse is a webservices mediation framework

“Any to any data connectivity and transformation (including Web Services) built on an advanced, proven, reliable middleware infrastructure”

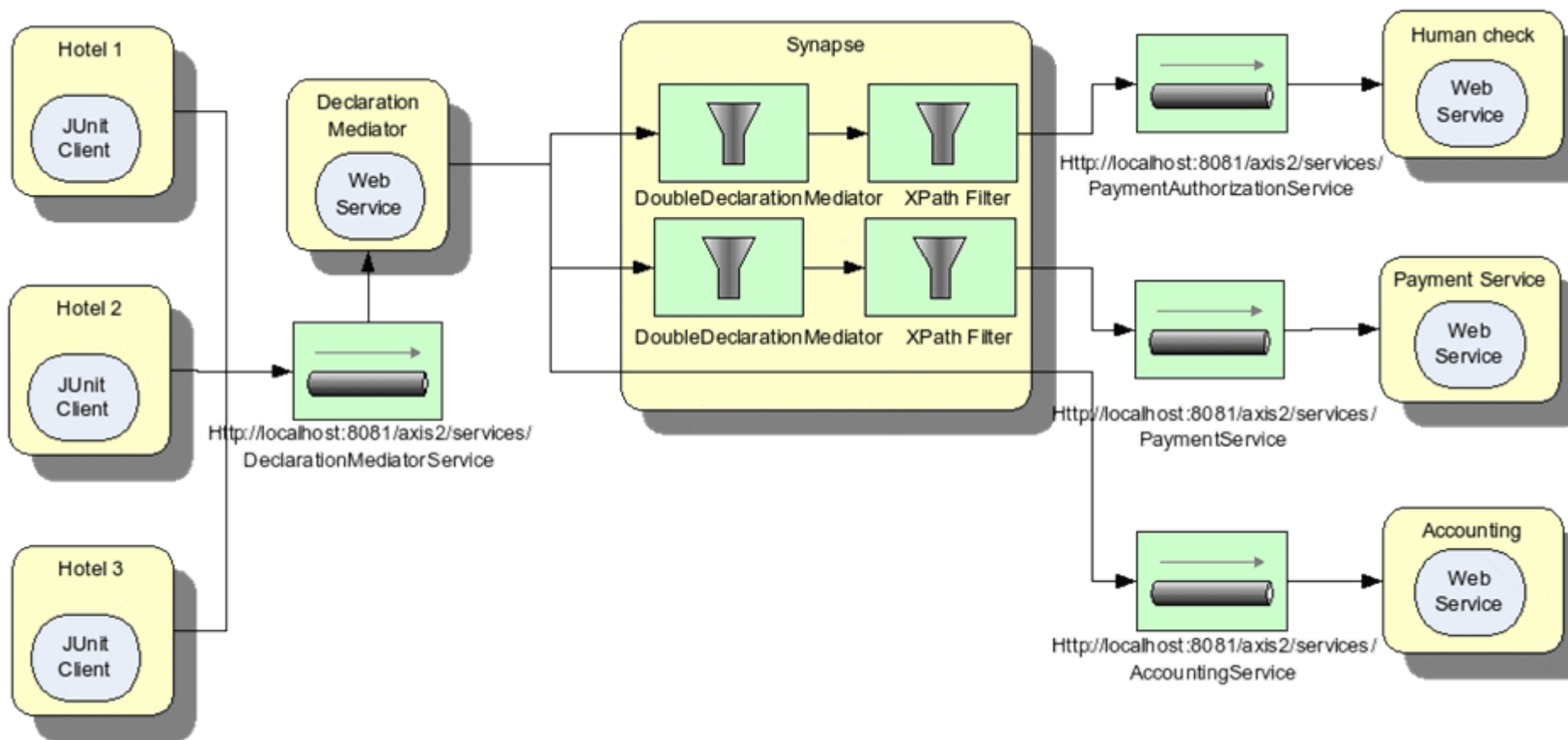
Synapse features

- Content based routing
 - XPath, Regex, Headers, Custom
- Switch
 - between POX, SOAP, Rest
- Virtualization
 - virtual to real endpoint mapping
- Transformation
 - XSLT, SOAP to JSON, Custom
- Manage
 - Logging, tracing
- Filtering

Synapse deployment model

- Transparent Proxy
 - Synapse used as a HTTP Proxy
 - With or without WS-Addressing headers
 - With a `Smart Client`
 - WS-Addressing specifies target
 - Pure rule based mediation
- Gateway
 - `Proxy Services` model
 - Hosting virtual endpoints in Synapse
 - Redirect to an Endpoint or Apply rules/sequences
 - Allows Transport, QoS switching etc. for the `new` service
 - Endpoint based model
 - Rule based mediation

Message filtering with Synapse

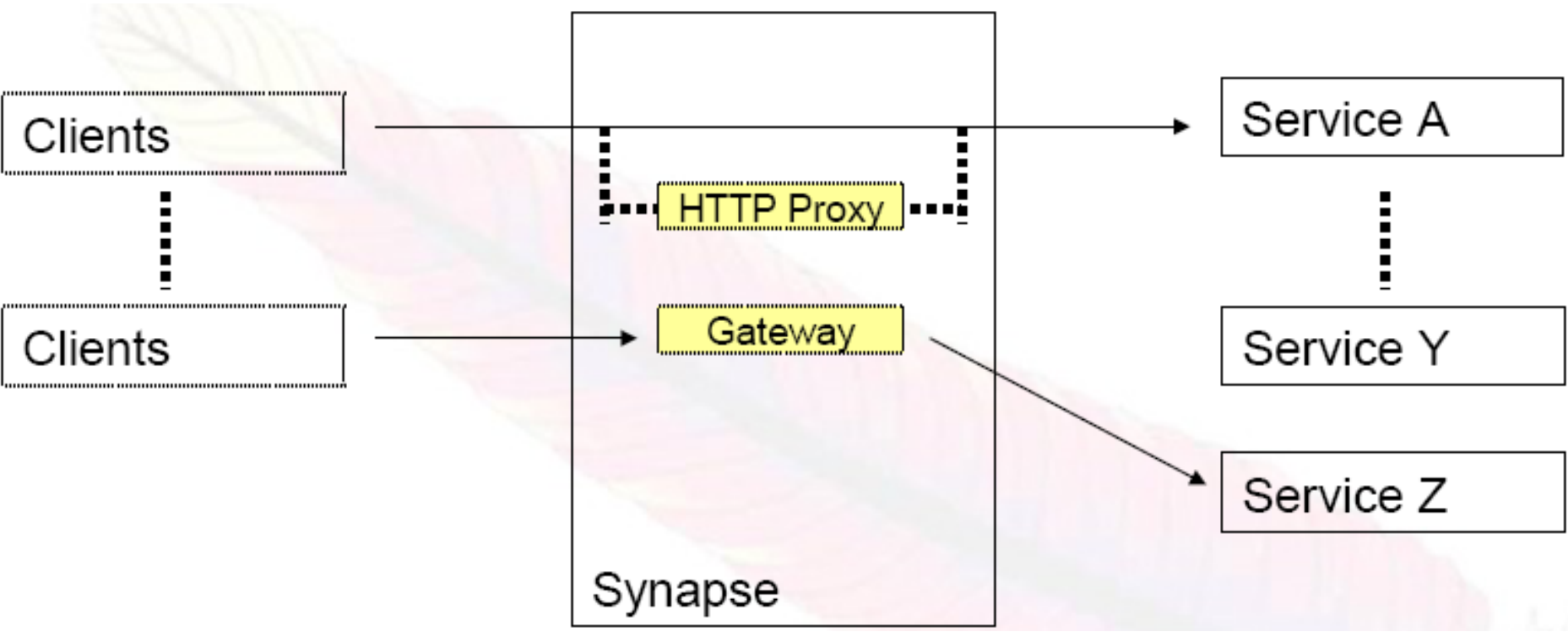


Agenda

- Introduction 2 min.
- Axis2 overview 13 min.
- WS-* support from Axis2 20 min.
- EIP with Synapse 10 min.
- Questions 5 min.



Synapse deployment model (1)



WS-Addressing

- SOAP doesn't provide a standard way to:
 - specify where a message is going...
 - specify where the reply must be sent to...
 - specify where an error should be sent to...
- WS-Addressing fills this gap

```
<SOAP-ENV:Header>
  <wsa:MessageID>http://example.com/UniqueMessageIdString</wsa:MessageID>
  <wsa:ReplyTo>
    <wsa:Address>http://myClient.example/someClientUser</wsa:Address>
  </wsa:ReplyTo>
  <wsa:FaultTo>
    <wsa:Address>http://myserver.example/DemoErrorHandler</wsa:Address>
  </wsa:FaultTo>
  <wsa:To>http://myserver.example/DemoServiceURI</wsa:To>
  <wsa:Action>http://myserver.example/DoSomething</wsa:Action>
</SOAP-ENV:Header>
```

WS-Addressing Demo

Request-Reply enterprise integration pattern

“When an application sends a message, how can it get a response from the receiver”

1. Send a message to a webservice.
2. WSA:Action is used to determine the method to invoke.
3. WSA:ReplyTo is used to send the result to a second webservice.

Invalid message channel with Synapse

