WS-Addressing

WS-Addressing via XML Configuration/Java API

CXF provides support for the 2004-08 and 1.0 versions of WS-Addressing.

To enable WS-Addressing you may enable the WSAddressingFeature on your service. If you wish to use XML to configure this, you may use the following syntax:

```
<jaxws:endpoint id="{your.service.namespace}YourPortName">
  <jaxws:features>
    <wsa:addressing xmlns:wsa="http://cxf.apache.org/ws/addressing"/>
  </jaxws:features>
</jaxws:endpoint>
```

You can also use the same exact syntax with a <jaxws:client>

```
<jaxws:client id="{your.service.namespace}YourPortName">
  <jaxws:features>
    <wsa:addressing xmlns:wsa="http://cxf.apache.org/ws/addressing"/>
  </jaxws:features>
</jaxws:client>
```

From an API point of view this looks very similar:

```
import org.apache.cxf.jaxws.EndpointImpl;
import org.apache.cxf.ws.addressing.WSAddressingFeature;

MyServiceImpl implementor =  new MyServiceImpl()
EndpointImpl ep = (EndpointImpl) Endpoint.create(implementor);
ep.getFeatures().add(new WSAddressingFeature());
ep.publish("http://some/address");
```

You can also use it with the ClientProxyFactoryBeans and ServerFactoryBeans (and their JAX-WS versions, namely JaxWsProxyFactoryBean and JaxWsServerFactoryBean):

```
import org.apache.cxf.frontend.simple.ClientProxyFactoryBean;
import org.apache.cxf.ws.addressing.WSAddressingFeature;

ClientProxyFactoryBean factory = new ClientProxyFactoryBean();
factory.setServiceClass(MyService.class);
factory.setAddress("http://acme.come/some-service");
factory.getFeatures().add(new WSAddressingFeature());
MyService client = (MyService) factory.create();
```

Enabling WS-Addressing with WS-Policy

If you're using WS-Policy, CXF can automatically set up WS-Addressing for you if you use the <Addressing> policy expression.

Decoupled responses

By default, WS-Addressing uses anonymous Reply-To addresses. This means the request/response patterns are synchronous in nature and the response is sent back via the normal reply channel. However, WS-Addressing allows for a decoupled endpoint to be used for receiving the response and CXF will then correlate it with the appropriate request. There are a few ways for configuring the address on which CXF will listen for decoupled WS-Addressing responses.

HTTP Conduit configuration

The HTTP Conduit's client configuration has an option for a DecoupledEndpoint address. If the conduit has this configured, all requests sent via that conduit that have WS-Addressing enabled will have their responses sent to that endpoint:

```
<http:conduit name="{http://apache.org/hello_world_soap_http}SoapPort.http-conduit">
    <http:client DecoupledEndpoint="http://localhost:9090/decoupled_endpoint"/>
</http:conduit>
```

Request Property

The address can be set via a Request Context property.

```
((BindingProvider)proxy).getRequestContext()
    .put("org.apache.cxf.ws.addressing.replyto", "http://localhost:9090/decoupled_endpoint");
```

AddressingProperties

The CXF org.apache.cxf.ws.addressing.impl.AddressingPropertiesImpl object can be used to control many aspects of WS-Addressing including the Reply-To:

```
AddressingProperties maps = new AddressingPropertiesImpl();
EndpointReferenceType ref = new EndpointReferenceType();
AttributedURIType add = new AttributedURIType();
add.setValue("http://localhost:9090/decoupled_endpoint");
ref.setAddress(add);
maps.setReplyTo(ref);
maps.setFaultTo(ref);

((BindingProvider)port).getRequestContext()
        .put("javax.xml.ws.addressing.context", maps);
```


This method can also be used to configure the namespace/version of the WS-Addressing headers, exact message ID's, etc...