Dynamic Endpoints

Marc Hadley
Sun Microsystems, Inc.

Provider Interface

```
public interface Provider {
   boolean invoke(
      LogicalMessageContext context);
}
```

- Alternative to regular SEI
  - Endpoint implements either Provider or regular SEI
- Return true from invoke if response message
- Can throw JAXRPCException or ProtocolException
- Lifecycle same for Provider or SEI based

Context Properties

- javax.xml.rpc.service.provider.wsdl
  - Provides access to the WSDL for the endpoint. Type is InputStream.
- javax.xml.rpc.service.provider.service
  - The name of the service being invoked in the WSDL. Type is QName.
- javax.xml.rpc.service.provider.port
  - The name of the port over which the current message was received in the WSDL. Type is QName.

Example: Echo

```
public class MyService implements Provider {
    public MyService {
    }
    public boolean invoke(LogicalMessageContext
    context) {
        return true;
    }
}
```

Example: Static Reply

```
public class MyService implements Provider {
   public MyService {
   }
   public boolean invoke(LogicalMessageContext
   context) {
      Source request = context.getMessage();
      String replyElement =
         new String("<n:ack xmlns:n='...'/>");
      StreamSource reply =
         new StreamSource(new
         StringReader(replyElement));
      context.setMessage(reply);
      return true;
   }
}
```

Example: Using JAXB

```
public class MyService implements Provider {
   public MyService {
   }
   public boolean invoke(LogicalMessageContext
   context) {
      JAXBContent jc =
         JAXBContext.newInstance(...);
      Object request = context.getMessage(jc);
      ...
      Acknowledgement reply =
         new Acknowledgement(...);
      context.setMessage(reply,jc);
      return true;
   }
}
```