Create and Secure Your REST APIs with Apache CXF

Andrei Shakirin, Talend

ashakirin@talend.com
ashakirin.blogspot.com

Agenda

- REST Principles in API Design
- Using CXF JAX-RS Features
- Secure REST API

About Me

- Software architect in Talend Team
- PMC in Apache CXF

Representational State Transfer

- Set of principals and restrictions
- HTTP is one instantiation of the REST
- The scope of REST architectural style: loosely coupled application

REST Principles

1. Everything has an ID
2. Using IDs to link things together: hypermedia and HATEOAS
3. Uniform interface
4. Interaction with resources through the representation
5. Communication is stateless

REST Violation: Operations Sematic

```
GET: /users/getUser?name=testUser
GET: /users/addUser?name=testUser&description=…
GET: /users/deleteUser?name=testUser
GET: /users/updateUser?name=testUser&description=newDescription
```

```java
@Path("users")
@Consumes({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
@Produces({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
public interface UserServiceIncorrect {

    @GET
    @Path("/getUser")
    UserTO getUser(@QueryParam("name") String name);

    @GET
    @Path("/addUser")
    Response addUser(@QueryParam("name") String name,
            @QueryParam("description") String description);

    @GET
    @Path("/deleteUser")
    void deleteUser(@QueryParam("name") String name);

    @GET
    @Path("/updateUser")
    Response updateUser(@QueryParam("name") String name,
            @QueryParam("description") String description);
}
```

Correct operations sematic

```
GET: /users/testUser
POST: /users
DELETE: /users/testUser
PUT: /users/testUser
```

```java
/**
 * REST operations for users.
 */
@Path("users")
@Consumes({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
@Produces({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
public interface UserServiceTry {

    @GET
    @Path("{name}")
    UserTO getUser(@PathParam("name") String name);

    @POST
    Response addUser(UserTO user);

    @DELETE
    @Path("{name}")
    void deleteUser(@PathParam("name") String name);

    @PUT
    @Path("{name}")
    Response updateUser(@PathParam("name") String name, UserTO user);
}
```

REST Violation: Tunneling Error Codes

```
HTTP/1.1 200 OK
Content-Type: application/xml
Headers: {Content-Type=[application/xml], Date=[Fri, 26 Jun 2015
12:48:36 GMT]}


<BusinessErrorReponse>
  <ns:globalStatus>ERROR</ns:globalStatus>
  <ns:detailStatusCodes>
    <ns:statusCode>
      <ns:statusCode>PR-C.4001</ns:statusCode>
      <ns:contextPart>PR-C</ns:contextPart>
      <ns:codePart>4001</ns:codePart>
      <ns:type>SERVER_BUSINESS_ERROR</ns:type>
    </ns:statusCode>
    <ns:detailCode>0000000000000650640</ns:detailCode>
    <ns:devMessage>User Id [a1b2c3] is not found</ns:devMessage>
  </ns:detailStatusCodes>
<BusinessErrorReponse>
```

Source: Mike Pearce http://www.slideshare.net/MikePearce/api-anti-patterns-4920731

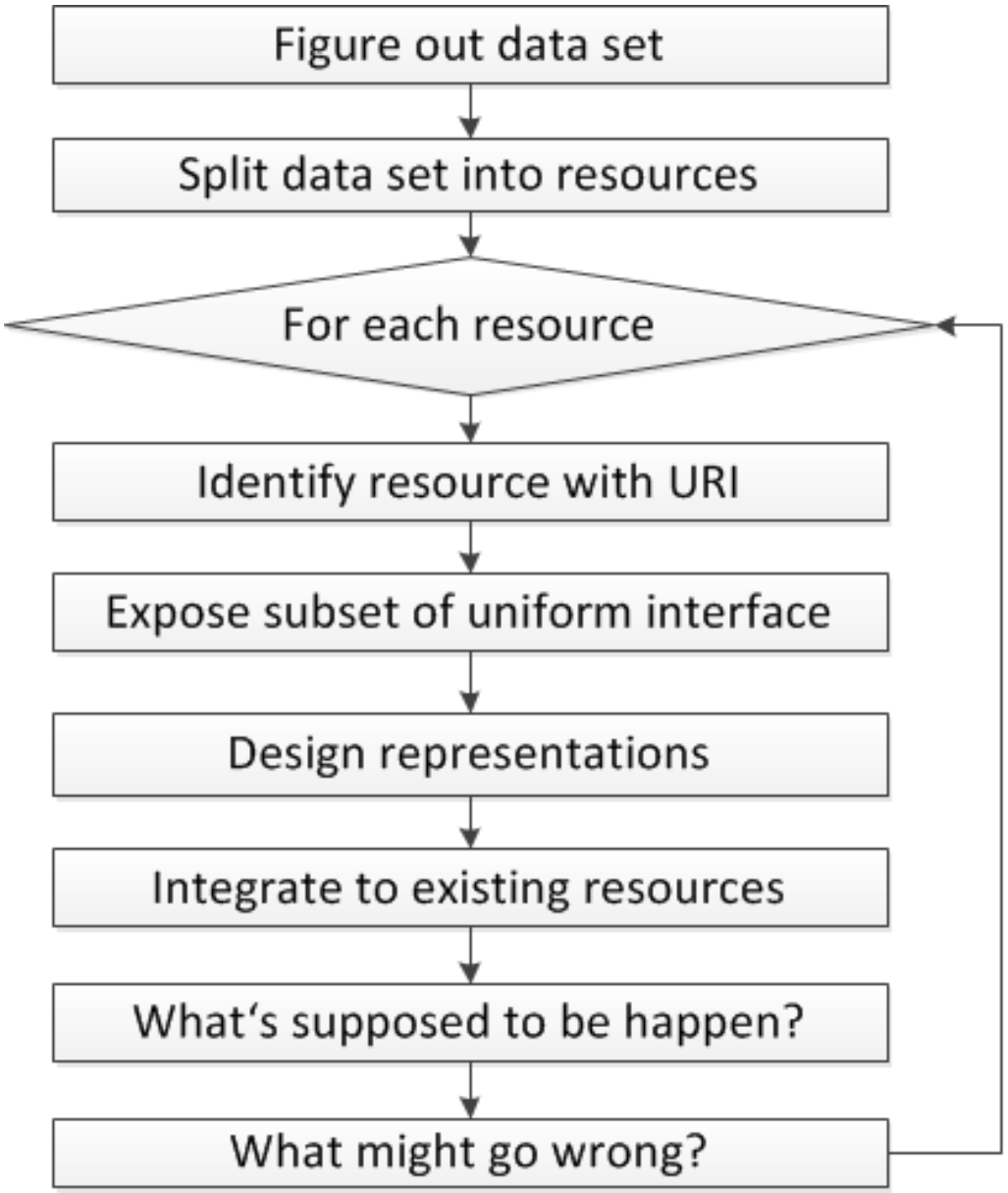
Source: Mike Pearce http://www.slideshare.net/MikePearce/api-anti-patterns-4920731

REST: Correct Error Codes

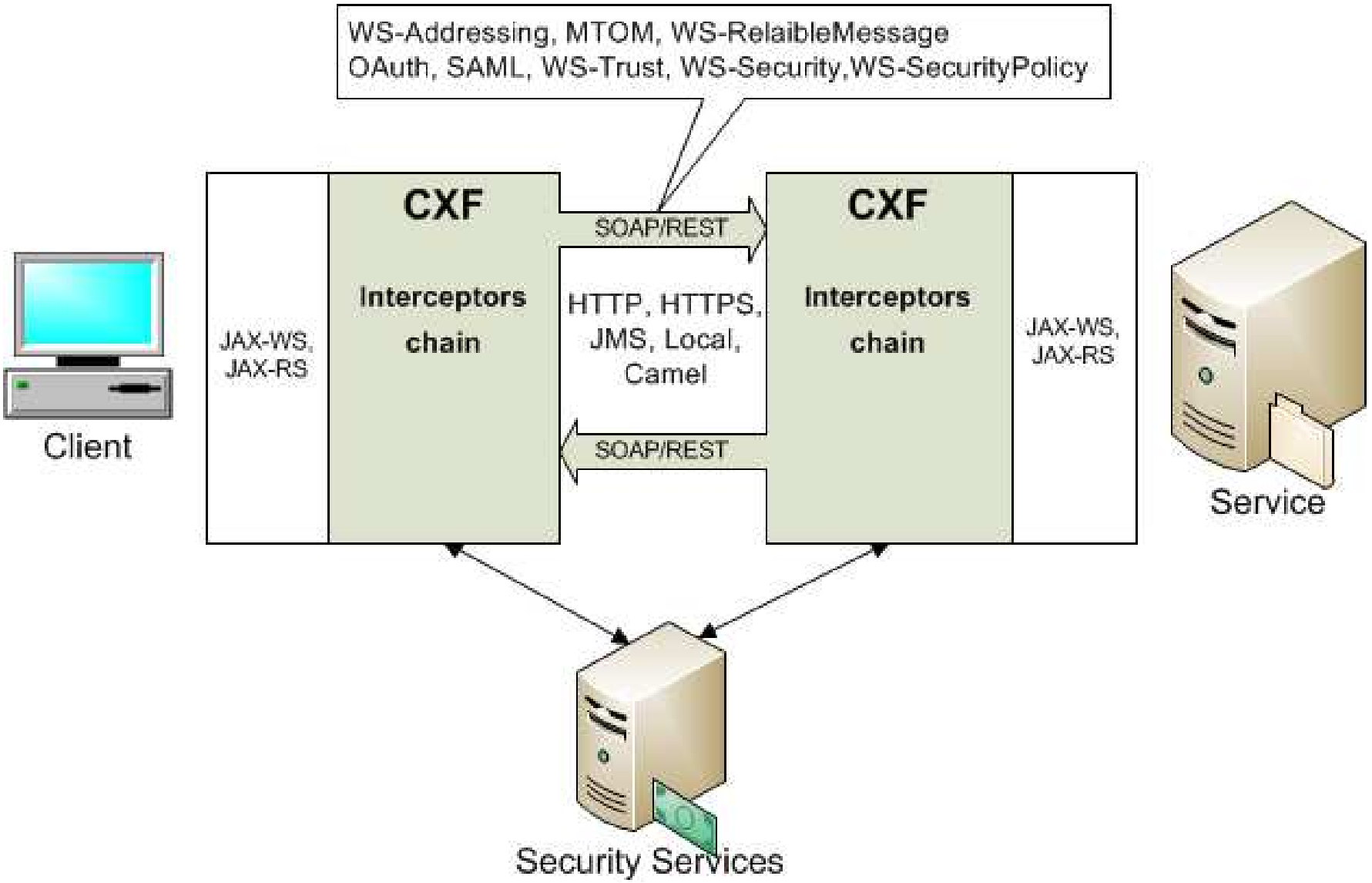
```
HTTP/1.1 404 Not Found
Content-Type: text/plain
X-Application-Error-Code: USER_NOT_FOUND
X-Application-Error-Info: entity=user,id=a1b2c3

A user [a1b2c3] is not found in storage. Check if user id is correct.
Refer following link for the details: https://cwiki.my-
organization.org/confluence/pages/viewpage.action?pageId=30751185/
```

REST API Design



Apache CXF


WS-Addressing, MTOM, WS-RelaibleMessage
OAuth, SAML, WS-Trust, WS-Security,WS-SecurityPolicy
CXF
Interceptors
chain
SOAP/REST
HTTP, HTTPS,
JMS, Local,
Camel
SOAP/REST
CXF
Interceptors
chain
JAX-WS,
JAX-RS
JAX-WS,
JAX-RS
Client
Service
Security Services

Filters and Interceptors

- JAX-RS Client and Server Filters
- JAX-RS Reader and Writer Interceptors
- MessageBodyReaders, MessageBodyWriters
- CXF Interceptors

JAX-RS Container Filters

Container Prematching Request Filter:

```
@Provider
@PreMatching
public class AuthenticationFilter implements ContainerRequestFilter {
    public void filter(ContainerRequestContext req) {
        try {
            List<String> authHeaders = req.getHeaders().get(
                    HttpHeaders.AUTHORIZATION);
            if (authHeaders == null || authHeaders.size() != 1) {
                // Exception
            }
            String[] authPair = StringUtils.split(authHeaders.get(0), " ");
            if (authPair.length != 2
                    || !NEGOTIATE_SCHEME.equalsIgnoreCase(authPair[0])) {
                // Exception
            }
            byte[] serviceTicket = getServiceTicket(authPair[1]);
            login(serviceTicket);
        } catch (Exception e) {
            req.abortWith(handleAuthenticationException(ex, m));
        }
    }
}
```

Container Response Filter:

```
@Provider
public class CacheControlFilter implements ContainerResponseFilter {
   public void filter(ContainerRequestContext req, ContainerResponseContext res) {
      if (req.getMethod().equals("GET")) {
         req.getHeaders().putSingle(HttpHeaders.CACHE_CONTROL, "max-age=86400");
      }
   }
}
```

JAX-RS Client Filters

Client Request Filter:

```
@Provider
public class ConditionalGetFilter implements ClientRequestFilter {
   public void filter(ClientRequestContext req) {
      if (req.getMethod().equals("GET")) {
         CacheEntry entry = cache.getEntry(req.getUri());
         if (entry != null) {
            req.getHeaders().putSingle("If-Modified-Since", entry.getLastModified());
         }
      }
   }
}
```

Client Response Filter:

```
@Provider
public class ConditionalGetCachingFilter implements ClientResponseFilter {
   public void filter(ClientResponseContext res) {
      if (res.getMethod().equals("GET")) {
         if (Response.Status.NOT_MODIFIED.equals(res.getStatus())) {
             // Set response from cache
         } else {
             // Cache the response
         }
      }
   }
}
```

JAX-RS Writer Interceptors

```
public class GZipWriterInterceptor implements WriterInterceptor {

    @Override
    public void aroundWriteTo(WriterInterceptorContext ctx)
            throws IOException, WebApplicationException {
        OutputStream old = ctx.getOutputStream();
        GZIPOutputStream gzipOutputStream = new GZIPOutputStream(old);
        ctx.setOutputStream(gzipOutputStream);
        updateHeaders(ctx);
        try {
            ctx.proceed();
        } finally {
            gzipOutputStream.finish();
            ctx.setOutputStream(old);

        }
    }
}
```

JAX-RS Reader Interceptors

```
public class GZipReaderInterceptor implements ReaderInterceptor {

    @Override
    public Object aroundReadFrom(ReaderInterceptorContext ctx)
            throws IOException, WebApplicationException {
        if (isGzipped(ctx)) {
            InputStream old = ctx.getInputStream();
            ctx.setInputStream(new GZIPInputStream(old));
            try {
                return ctx.proceed();
            } finally {
                ctx.setInputStream(old);
            }
        } else {
            return ctx.proceed();
        }
    }
}
```

JAX-RS MessageBody Writer

```
PUT /user/blocked HTTP/1.1
Content-Type: application/boolean+xml

<?xml version="1.0" encoding="utf-8"?>
<boolean>true</boolean>
```

```java
@Provider
@Produces("application/boolean+xml")
public class BooleanMessageBodyWriter implements MessageBodyWriter<Boolean> {
    @Override
    public boolean isWriteable(Class<?> type, Type genericType,Annotation[] annotations, MediaType mediaType) {
        return type == Boolean.class;
    }

    @Override
    public void writeTo(Boolean myBool, Class<?> type, Type genericType,Annotation[] annotations, MediaType mediaType,
            MultivaluedMap<String, Object> httpHeaders, OutputStream entityStream) throws IOException,
            WebApplicationException {
        StringBuilder sb = new StringBuilder();
        sb.append("<boolean><boolean>").append(myBool.toString()).append("</boolean></boolean>");
        DataOutputStream dos = new DataOutputStream(entityStream);
        dos.writeUTF(sb.toString());
    }

    @Override
    public long getSize(Boolean t, Class<?> type, Type genericType,
            Annotation[] annotations, MediaType mediaType) {
        return -1;
    }
}
```

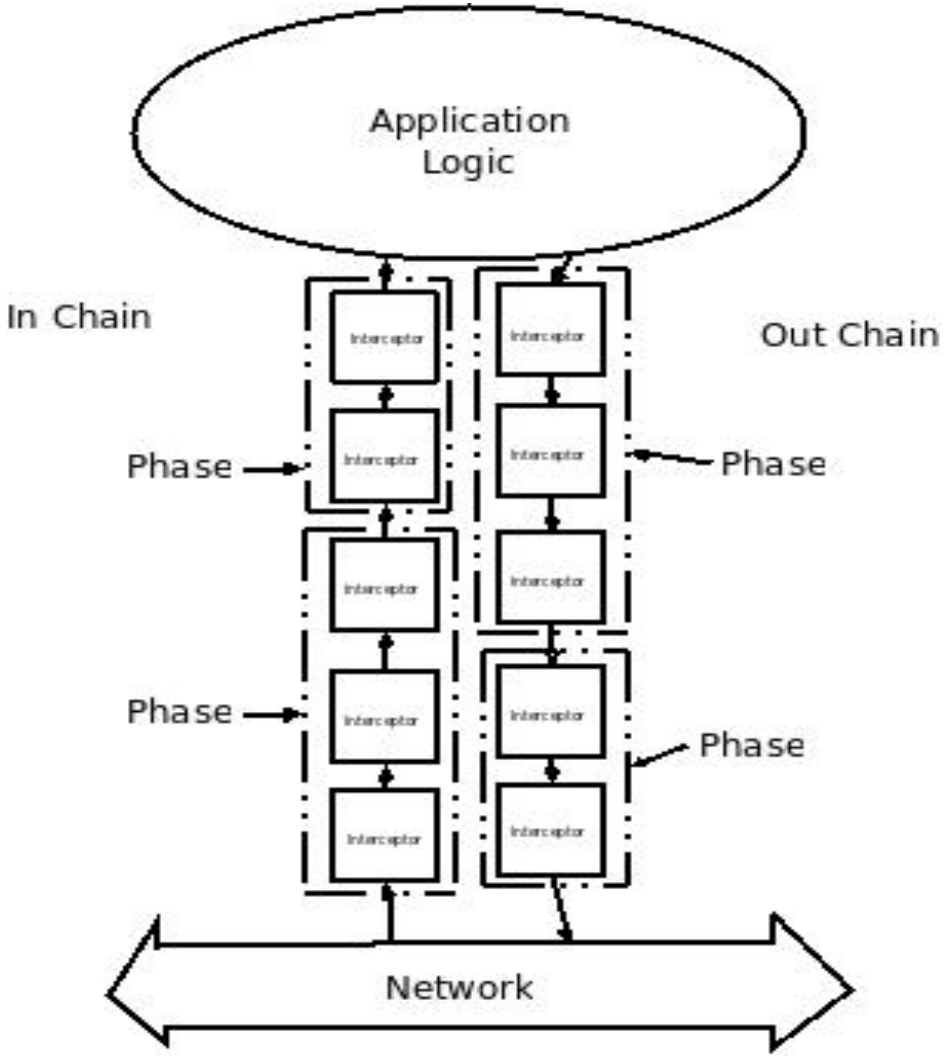
JAX-RS MessageBody Reader

```java
@Provider
@Consumes("application/boolean+xml")
public class BooleanMessageBodyReader implements MessageBodyReader<Boolean> {

    @Override
    public boolean isReadable(Class<?> type, Type genericType,
            Annotation[] annotations, MediaType mediaType) {
        return type == Boolean.class;
    }

    @Override
    public Boolean readFrom(Class<Boolean> type, Type genericType,
            Annotation[] annotations, MediaType mediaType,
            MultivaluedMap<String, String> httpHeaders, InputStream entityStream)
            throws IOException, WebApplicationException {
        try {
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(entityStream);
            NodeList nodes = doc.getElementsByTagName("boolean");
            if (nodes.getLength() == 0) {
                throw new IllegalArgumentException("Wrong XML format");
            }
            return Boolean.parseBoolean(nodes.item(0).getTextContent());
        } catch (Exception e) {
            throw new IllegalStateException("Cannot pasre XML stream: " + e.getMessage(), e);
        }
    }
}
```

CXF Inteceptors



CXF Inteceptors

Outgoing Chain	Business code (PRE/POST/_INVOKE, PRE/USER/POST/_LOGICAL)
	Umarshalling/Marshalling (MARSHAL)
	Protocol processing (PRE/USER/POST/_PROTOCOL, WRITE)
	Stream processing (PRE/USER/POST/STREAM)
	Transport (SEND)

Incoming Chain	Transport (RECIEVE)
	Stream processing (PRE/USER/POST/STREAM)
	Protocol processing (PRE/USER/POST/_PROTOCOL, READ)
	Umarshalling/Marshalling (UNMARSHAL)
	Business code (PRE/POST/_INVOKE, PRE/USER/POST/_LOGICAL)

CXF Interceptors

```java
public class SampleInInterceptor extends AbstractPhaseInterceptor<Message> {

    public SampleInInterceptor() {
        super(Phase.RECEIVE);
        getAfter().add(SomeOtherInterceptor.class.getName());
    }

    public void handleMessage(Message message) {
        // Process message
    }

    public void handleFault(Message messageParam) {
        // Process fault
    }
}
```

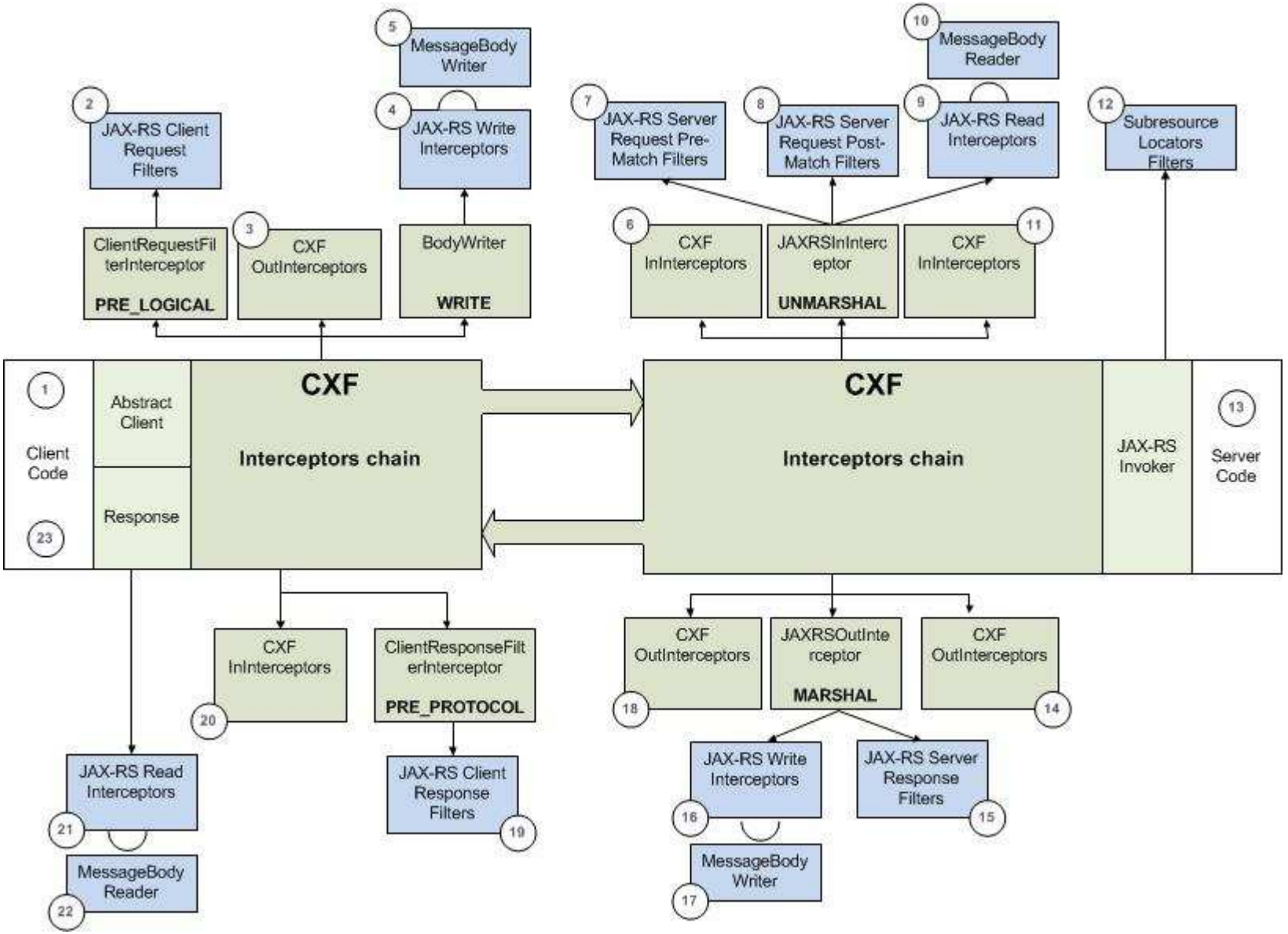
```java
public class SampleOutInterceptor extends AbstractPhaseInterceptor<Message> {

    public SampleOutInterceptor() {
        super(Phase.SEND);
        getAfter().add(SomeOtherInterceptor.class.getName());
    }

    public void handleMessage(Message message) {
        // Process message
    }

    public void handleFault(Message messageParam) {
        // Process fault
    }
}
```

CXF Filters and Interceptors



Exception Handling: Server Side

JAXRS exception mappers:

```
@Provider
public class SyncopeExceptionMapper implements
      ExceptionMapper<SyncopeClientException> {

    @Override
    public Response toResponse(final SyncopeClientException ex) {
        LOG.error("SyncopeClientException thrown by REST method: " +
                ex.getMessage(), ex);

        ResponseBuilder builder = ex.isComposite() ?
                getSyncopeClientCompositeExceptionResponse(ex.asComposite())
                : getSyncopeClientExceptionResponse(ex);

        return builder.build();
    }
}
```

404 Not found
X-Application-Error-Code: *EntityNotFound*
X-Application-Error-Info: entity=user,id=a1b2c3

A user ‘a1b2c3‘ is not found in Syncope storage. Check if user name is correct. Refer following link for the details: https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=30751185/

Exception Handling: Client Side

Client exception mapper:

```
public class ClientExceptionMapper implements ResponseExceptionMapper<Exception> {

    public Exception fromResponse(Response r) {
        if (r.getStatus() == HttpStatus.NOT_FOUND_404) {
            return new NotFoundException();
        } else {
            return null;
        }
    }
}
```

Catch WebApplicationException:

```
public void invoke() {
    BookStore proxy = JAXRSClientFactory.create("http://books", BookStore.class);
    try {
        proxy.getBook();
    } catch(WebApplicationException ex) {
      Response r = ex.getResponse();
      String message = ex.getMessage();
    }
}
```

Document Your API: Swagger

- Formal specification for REST APIs (JSON and JSON schema)
- Ecosystem with Tools: codegeneration, documentation, test and sandbox
- Top-down and bottom-up approaches

Swagger in JAXRS API

```java
@Path("/sample")
@Api(value = "/sample", description = "Sample JAX-RS service with Swagger documentation")
public class Sample {

    @Produces({ MediaType.APPLICATION_JSON })
    @GET
    @ApiOperation(
        value = "Get operation with Response and @Default value",
        notes = "Get operation with Response and @Default value",
        response = Item.class,
        responseContainer = "List"
    )
    public Response getItems(
        @ApiParam(value = "Page to fetch", required = true) @QueryParam("page") @DefaultValue("1") int page) {

        return Response.ok(items.values()).build();
    }

    @Consumes({ MediaType.APPLICATION_JSON })
    @POST
    @ApiOperation(
        value = "Post operation with entity in a body",
        notes = "Post operation with entity in a body",
        response = Item.class
    )
    public Response createItem(
        @Context final UriInfo uriInfo,
        @ApiParam(value = "item", required = true) final Item item) {
        items.put(item.getName(), item);
        return Response
            .created(uriInfo.getBaseUriBuilder().path(item.getName()).build())
            .entity(item).build();
    }
}
```

Swagger Configuration in CXF

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:cxf="http://cxf.apache.org/core"
       xmlns:jaxrs="http://cxf.apache.org/jaxrs"
       xsi:schemaLocation="http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd
                           http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd
                           http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

    <import resource="classpath:META-INF/cxf/cxf.xml" />

    <!-- JAXRS providers -->
    <bean id="jsonProvider" class="com.fasterxml.jackson.jaxrs.json.JacksonJsonProvider" />
    <bean id="multipartProvider" class="org.apache.cxf.jaxrs.provider.MultipartProvider" />

    <!-- Application resources -->
    <bean id="sampleResource" class="demo.jaxrs.swagger.server.Sample" />

    <!-- CXF Swagger2Feature -->
    <bean id="swagger2Feature" class="org.apache.cxf.jaxrs.swagger.Swagger2Feature">
        <property name="basePath" value="/app/swaggerSample"/>
    </bean>

   <jaxrs:server id="sampleServer" address="/swaggerSample">
        <jaxrs:serviceBeans>
            <ref bean="sampleResource" />
        </jaxrs:serviceBeans>
        <jaxrs:providers>
            <ref bean="jsonProvider" />
            <ref bean="multipartProvider" />
        </jaxrs:providers>
        <jaxrs:features>
            <ref bean="swagger2Feature" />
        </jaxrs:features>
    </jaxrs:server>
</beans>
```

CXF Security

	Authentication	Authorization	Confidentiality	Integrity	Non-repudiation
HTTPS	●		●	●	
HTTP Basic	●				
HTTP Digest	●				
Kerberos	●				
SAML	●				
JWT	●				
RBAC		●			
XACML		●			
OAuth		●			
XML Encryption			●		
XML Signature				●	●
JWE			●		
JWS				●	●

SAML

```
<saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    ID="_EAF1D65DB246FCB19013668749383952" IssueInstant="2013-04-25T07:28:58.395Z"
    Version="2.0" xsi:type="saml2:AssertionType">
    <saml2:Issuer>issuer]</saml2:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:Reference URI="#_EAF1D65DB246FCB19013668749383952">
                <ds:DigestMethod
                    Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                <ds:DigestValue>BeqHM3hDi2jkd1i/9VgL52HpqtE=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>d4rbn3fo1t1349pri</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>MIIC</ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </ds:Signature>
    <saml2:Subject>
        <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
            NameQualifier="http://cxf.apache.org/sts">alice@EXAMPLE.COM</saml2:NameID>
        <saml2:SubjectConfirmation
            Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
            <saml2:SubjectConfirmationData
                xsi:type="saml2:KeyInfoConfirmationDataType">
                <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:X509Data>
                        <ds:X509Certificate>MIIEFjCCA3+</ds:X509Certificate>
                    </ds:X509Data>
                </ds:KeyInfo>
            </saml2:SubjectConfirmationData>
        </saml2:SubjectConfirmation>
    </saml2:Subject>
    <saml2:Conditions NotBefore="2013-04-25T07:28:58.397Z"
        NotOnOrAfter="2013-04-25T07:58:58.397Z">
        <saml2:AudienceRestriction>
            <saml2:Audience>https://localhost:9001/services/ReservationServiceProvider
            </saml2:Audience>
        </saml2:AudienceRestriction>
    </saml2:Conditions>
    <saml2:AttributeStatement>
        <saml2:Attribute
            Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue xsi:type="xs:string">manager</saml2:AttributeValue>
        </saml2:Attribute>
    </saml2:AttributeStatement>
</saml2:Assertion>
```

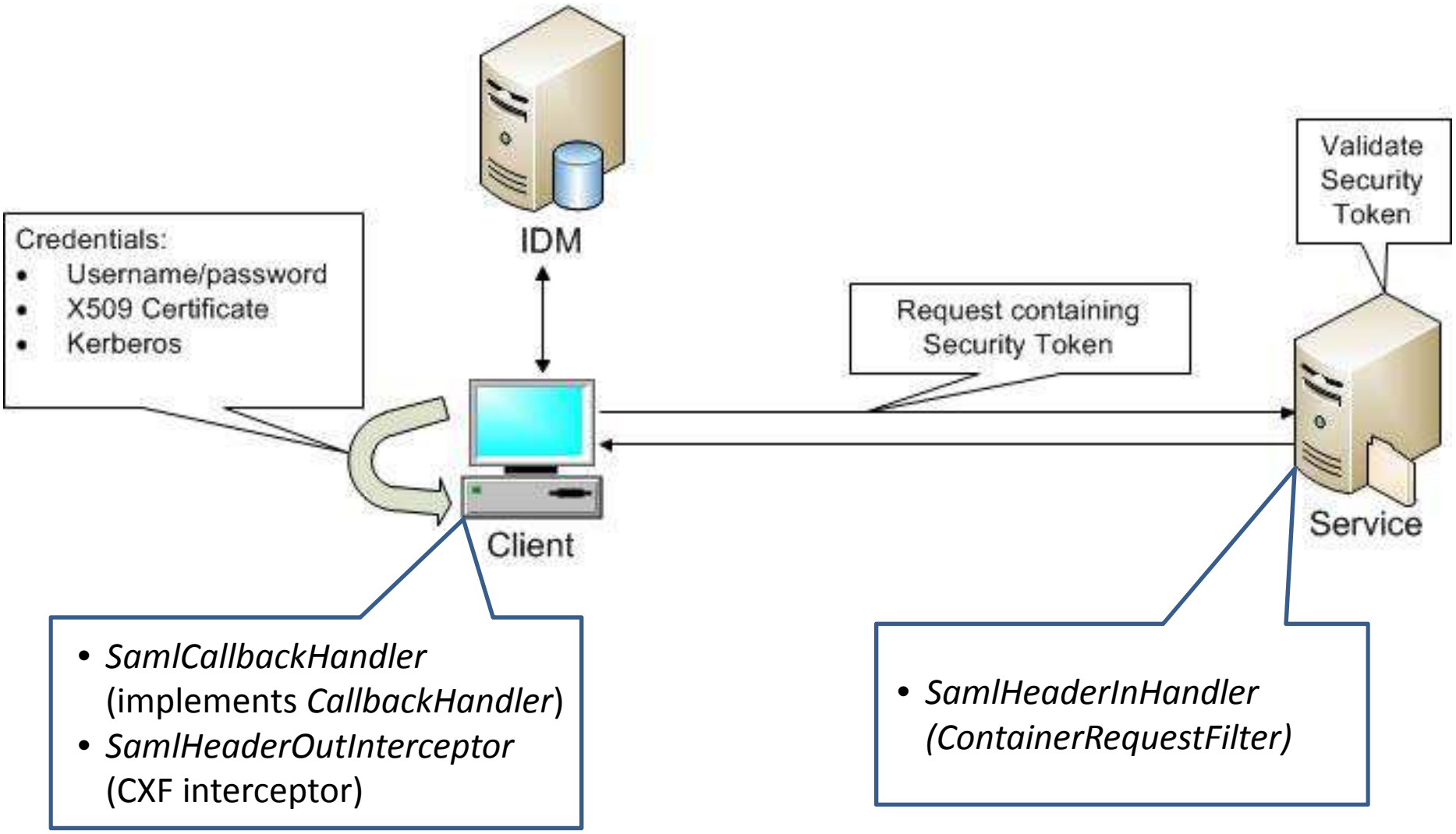
Issuer Signature

Subject

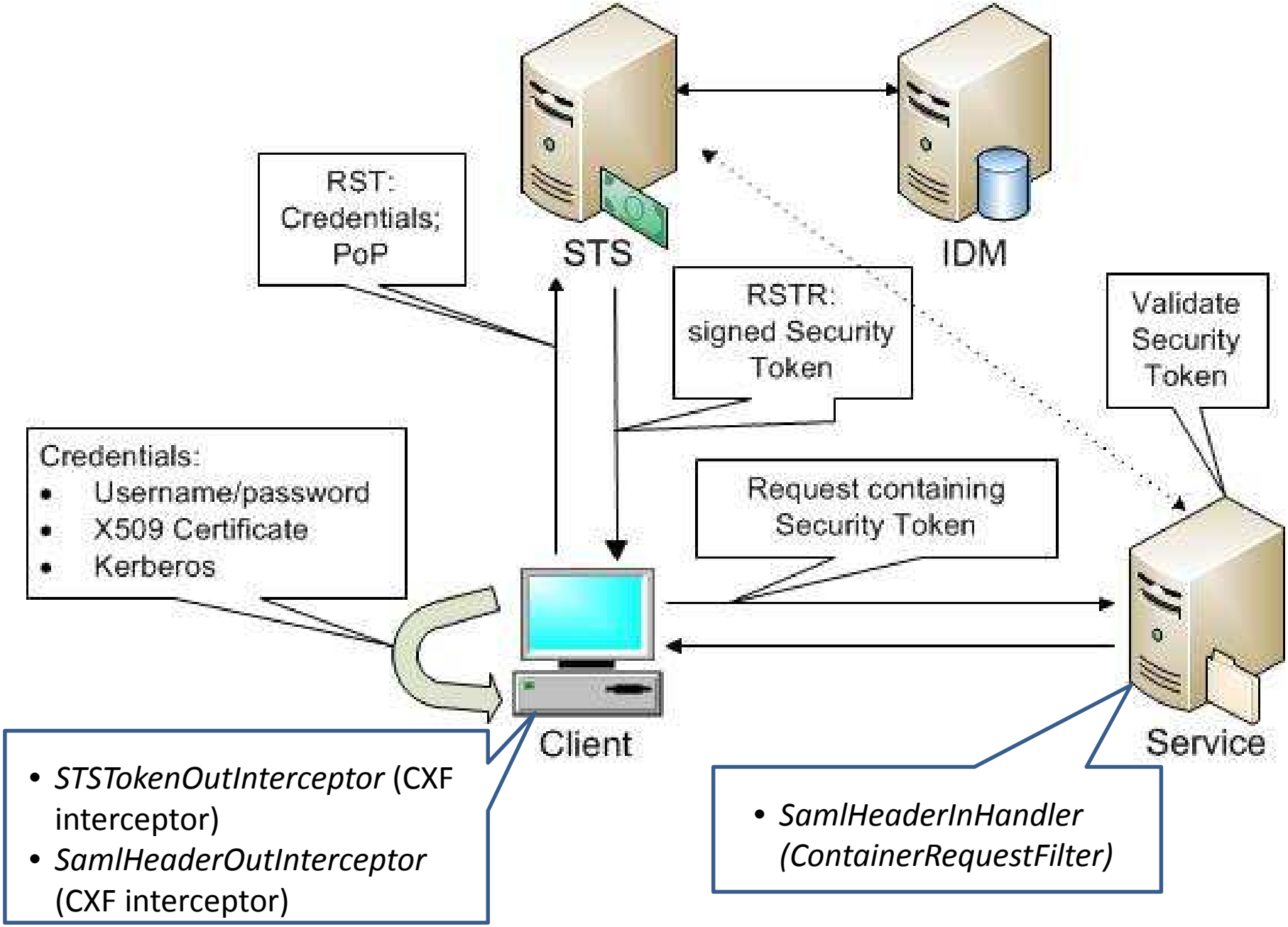
Conditions

Attribute Statements

SAML Token Issued By Client



SAML Token Issued By STS



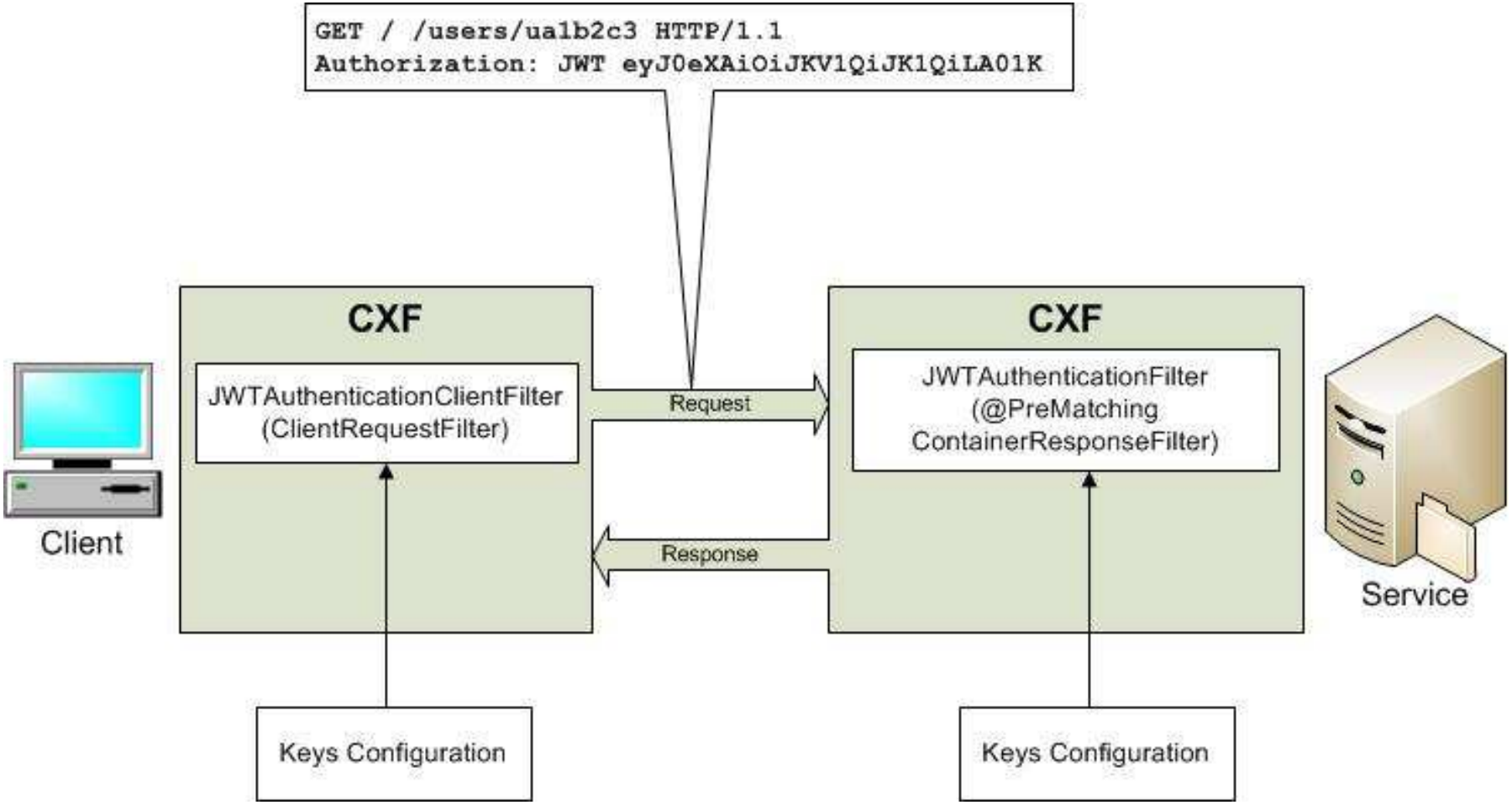
JSON Web Token

Header	{"typ":"JWT", "alg":"HS256"}
Claims	{"iss":"cxf:oauth", "sub":"userA" "exp":1300819380, "iat":1300819370, "nbf":1300800000, "aud":"api.mydomain.org" "mycustomclaim":"roleA"}

Header	eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9 .
Claims	eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFt cGxlLmNvbS9pc19yb290Ijp0cnVlfQ .
Signature	dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk

Signature = HMACSHA256(BASE64URL(UTF8(JWT Header)) + ’.’ + BASE64URL(JWT Claims), key)

JWT Authentication in CXF



Conclusion

- Be aware of and follow REST Principles in your API design
- Use JAX-RS Filters, Reader/Writer interceptors to implement cross-cutting functionality
- Consider Swagger to make your API attractive for the clients
- Choose the most suitable mechanism to secure your REST API

Links

- Apache CXF :
  http://cxf.apache.org/
  http://cxf.apache.org/docs/jax-rs.html

- Swagger :
  http://swagger.io/

- Blogs:
  http://sberyozkin.blogspot.com
  http://ashakirin.blogspot.de/
  http://aredko.blogspot.de/

JSON Web Signature

```
Header:
{"alg":"HS256","cty":"json"}

Payload:
{"Book":{"id":123,"name":"book"}}
```

```
JWS JSON serialization:
 {"payload":
       "eyJpc3MiOiJqb2UiLA0KICJleHAiLy9leGF
        tcGxlLmNvbS9pc19yb290Ijp0cnVlfQ",
      "signatures":[
       {"protected":"eyJhbGciOiJSUzI1NiJ9",
        "header":
         {"kid":"2010-12-29"},
        "signature":
         "cC4hiUPoj9EetdgQGe77Rw ..."},
       {"protected":"eyJhbGciOiJFUzI1NiJ9",
        "header":
         {"kid":"e9bc097a-ce51-4036-9562-d2ade882db0d"},
        "signature":
         "DtEhU3ljbEg8L38VWAfUAqO-Kg6NU1Q ..."}]
}
```

```
Compact JWS:
 eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
 .
 eyJpc3MiOiJqb2UiLA0KICJleHAmh0dHA6Ly9leGFt
 cGxlLmNvbS9pc19yb290Ijp0cnVlfQ
 .
 dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

JSON Web Encryption

Header:
```
{"alg":"RSA-OAEP",
 "enc":"A256GCM"}
```

Payload:
```
{"Book":{"id":123,"name":"book"}}
```

Encrypted structure:
```
BASE64URL(UTF8(JWE Protected Header)) + '.'
+ BASE64URL(JWE Encrypted Key) + '.'
+ BASE64URL(JWEInitialization Vector) + '.'
+ BASE64URL(JWE Ciphertext) + '.'
+ BASE64URL(JWE Authentication Tag)
```

```
eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkEyNTZHQ00ifQ.
OKOawDo13gRp2ojaHV7LFpZcgV7T6DVZKTyKOMTYUmKoTCVJRgckCL9kiMT03JGe
ipsEdY3mx_etLbbWSrFr05kLzcSr4qKAq7YN7e9jwQRb23nfa6c9d-StnImGyFDb
Sv04uVuxIp5Zms1gNxKKK2Da14B8S4rzVRltdYwam_lDp5XnZAYpQdb76FdIKLaV
mqgfwX7XWRxv2322i-vDxRfqNzo_tETKzpVLzfiwQyeyPGLBIO56YJ7eObdv0je8
1860ppamavo35UgoRdbYaBcoh9QcfylQr66oc6vFWXRcZ_ZT2LawVCWTIy3brGPi
6UklfCpIMfIjf7iGdXKHzg.
48V1_ALb6US04U3b.
5eym8TW_c8SuK0ltJ3rpYIzOeDQz7TALvtu6UG9oMo4vpzs9tX_EFShS8iB7j6ji
SdiwkIr3ajwQzaBtQD_A.
XFBoMYUZodetZdvTiFvSkQ
```

OAuth 2.0

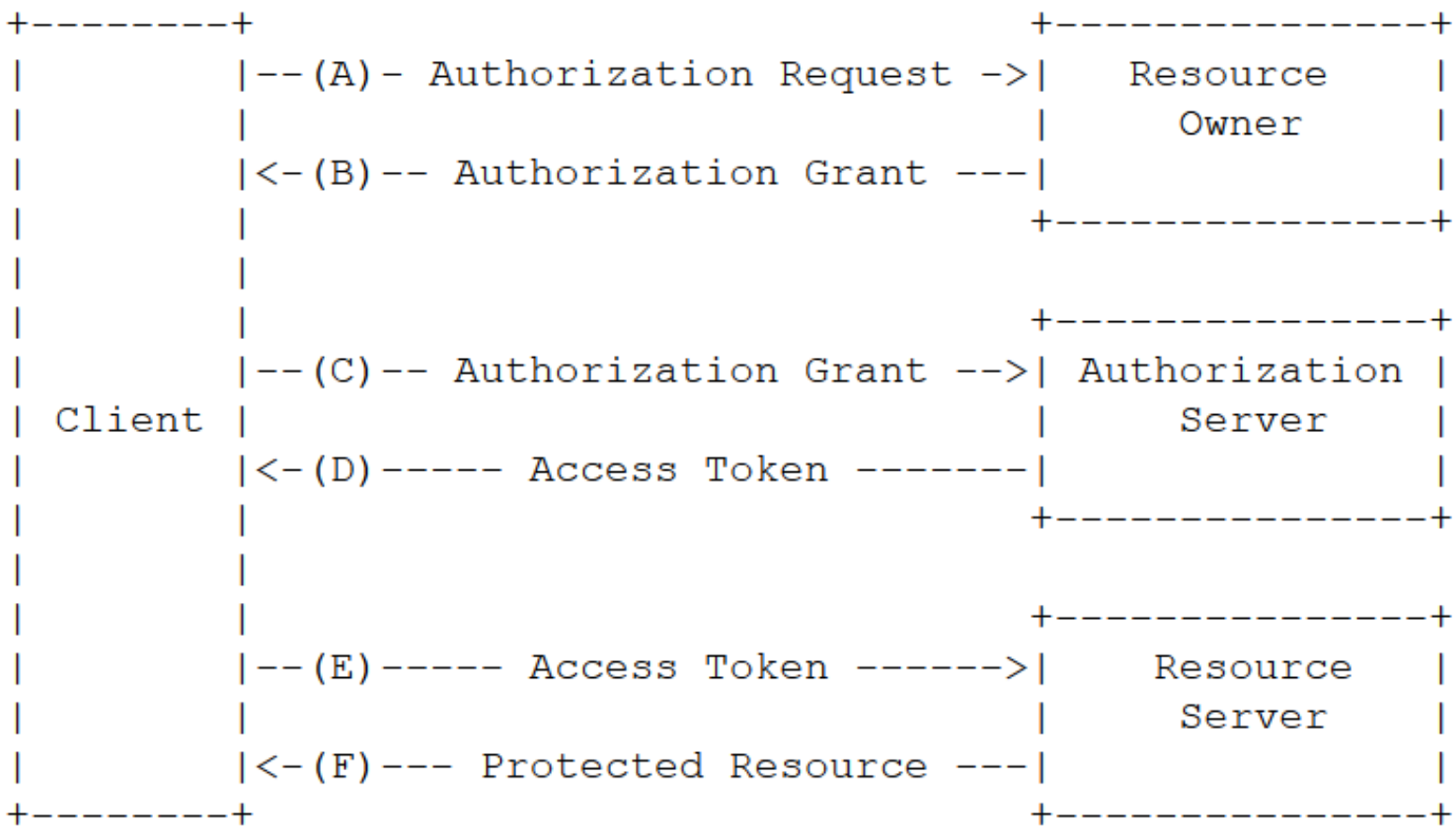
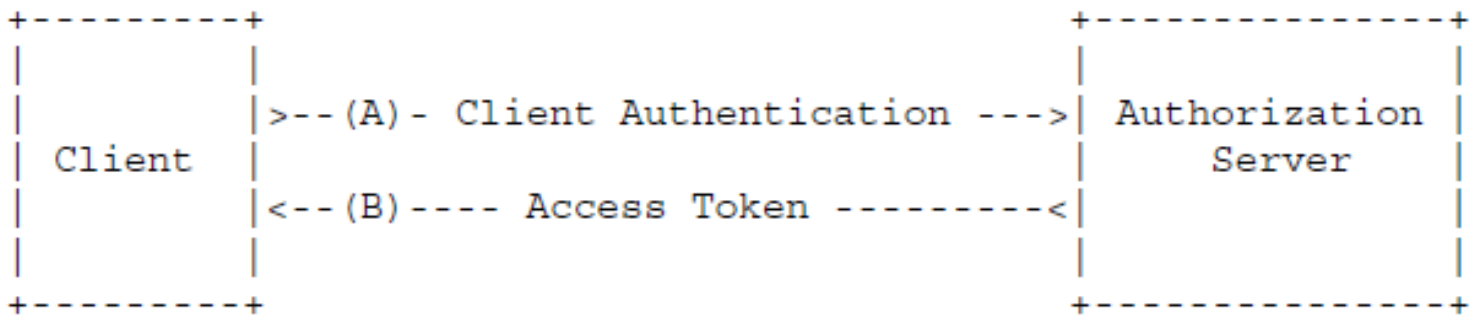


Diagram from OAuth 2.0 spec

OAuth 2.0

Client Credentials Grant



What is new in JAX-RS 2.1?

- Server Side Events
- Reactive programming model
- JEE Alignments (CDI support, declarative security)
- Performance improvements (non-blocking I/O)
- Improved Hypermedia support

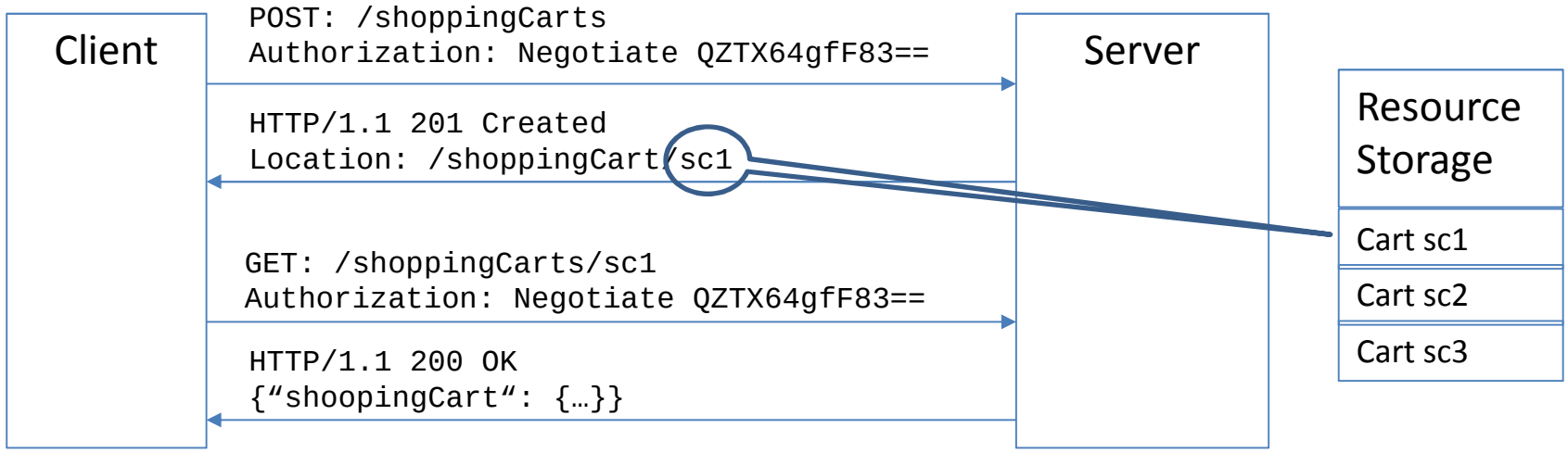
CXF Roadmap

- JAX-RS 2.1
- JWT, JWS, OAuth 2.0
- Providers and Resources autowiring
- Extended CDI support
- Swagger Integration improvements
- Logging improvements
- FIQL Extensions

REST Violation: Sessions & Cookies

- Server stores client specific application state
- State is not identifiable through URL
- Client sends Cookies implicitly with every request

Addressable Resource State



- ShoppingCart is stored as resource state
- Resource state is addressable through URL
- Request includes all information necessary to fulfil