REST and SOAP Services with Apache CXF

Andrei Shakirin, Talend

ashakirin@talend.com
ashakirin.blogspot.com/

Agenda

- Introduction in Apache CXF
- New CXF features
- Project using Apache CXF
- How CXF community works?

Background

- Software architect in Talend Team
- Committer in Apache CXF and Syncope projects
- Speaker for Apache and SOA conferences
- Member of OASIS S-Ramp Group

CXF History

- Apache Project since 2006
- Basis: Celtix (ObjectWeb), XFire (Codehaus)
- 7 major versions, 58 patch releases
- 33 committers (17 active)

Who uses CXF?

- Apache: Camel, ServiceMix, Syncope
- JBoss JAX-WS stack
- TomEE JAX-WS and JAX-RS stacks
- Talend, Fusesource, MuleSoft, WSO2
- ‘CXF – Services List‘: Google Adwords, TomTom, …

Why CXF?

Alternatives:

- Axis 2
- Metro
- Jersey
- …

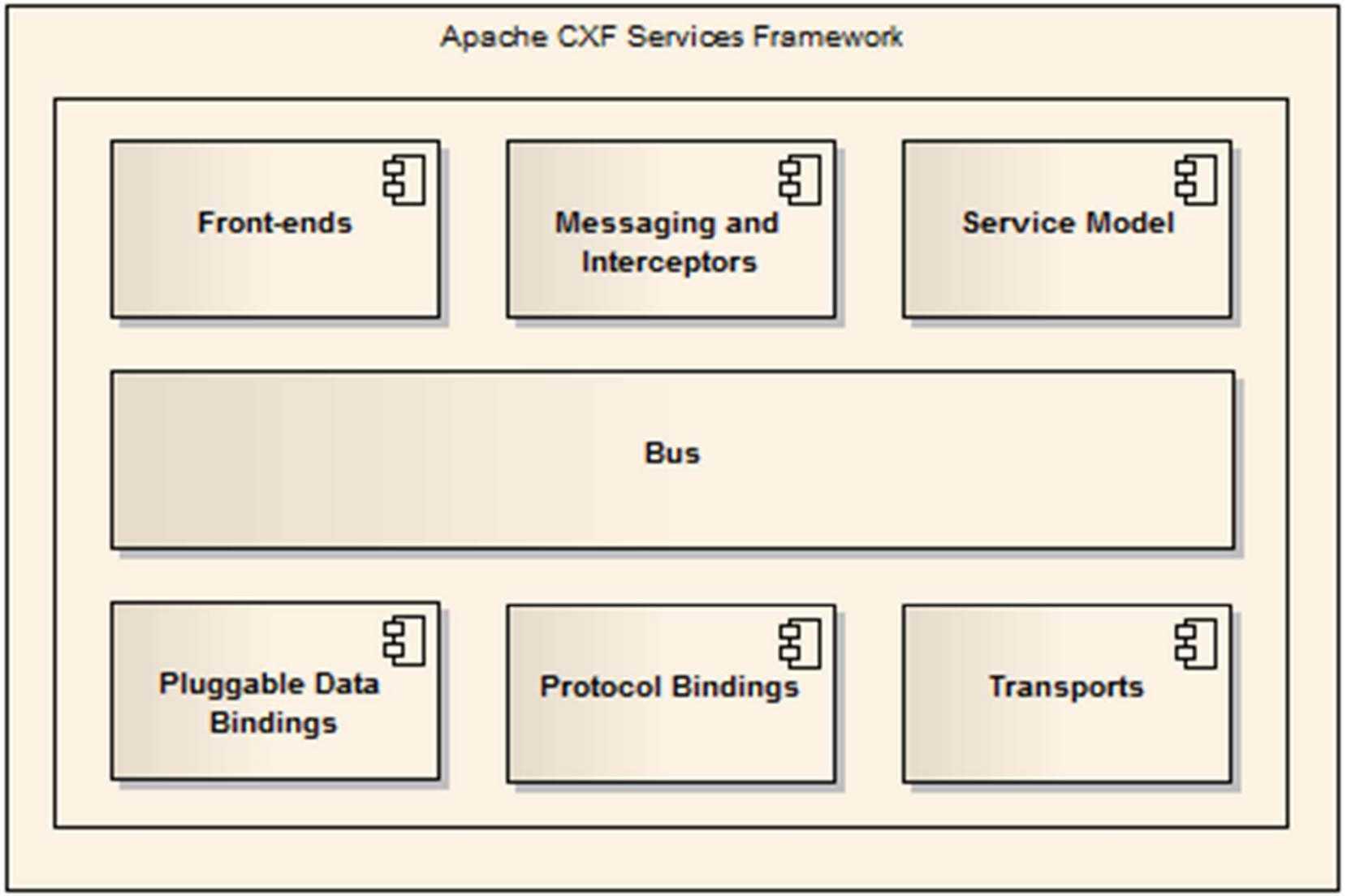
CXF Benefits:

- Strong standards support
- SOAP & Rest services
- Security
- Streaming and performance
- Flexibility
- Large and active community

Standards Support

- JAX-WS 2.2, JAX-RS 1.1, 2.0 (TCK tests)
- Basic support: WS-I Basic Profile
- Metadata: WS-Policy, WSDL 1.1
- Messaging: WS-Addressing, SOAP 1.1/1.2, MTOM
- Security: WS-Security, WS-SecurityPolicy, SAML, WS-Trust, WS-Notification, OAuth 2.0
- Quality of Service: WS-ReliableMessaging

CXF Architecture

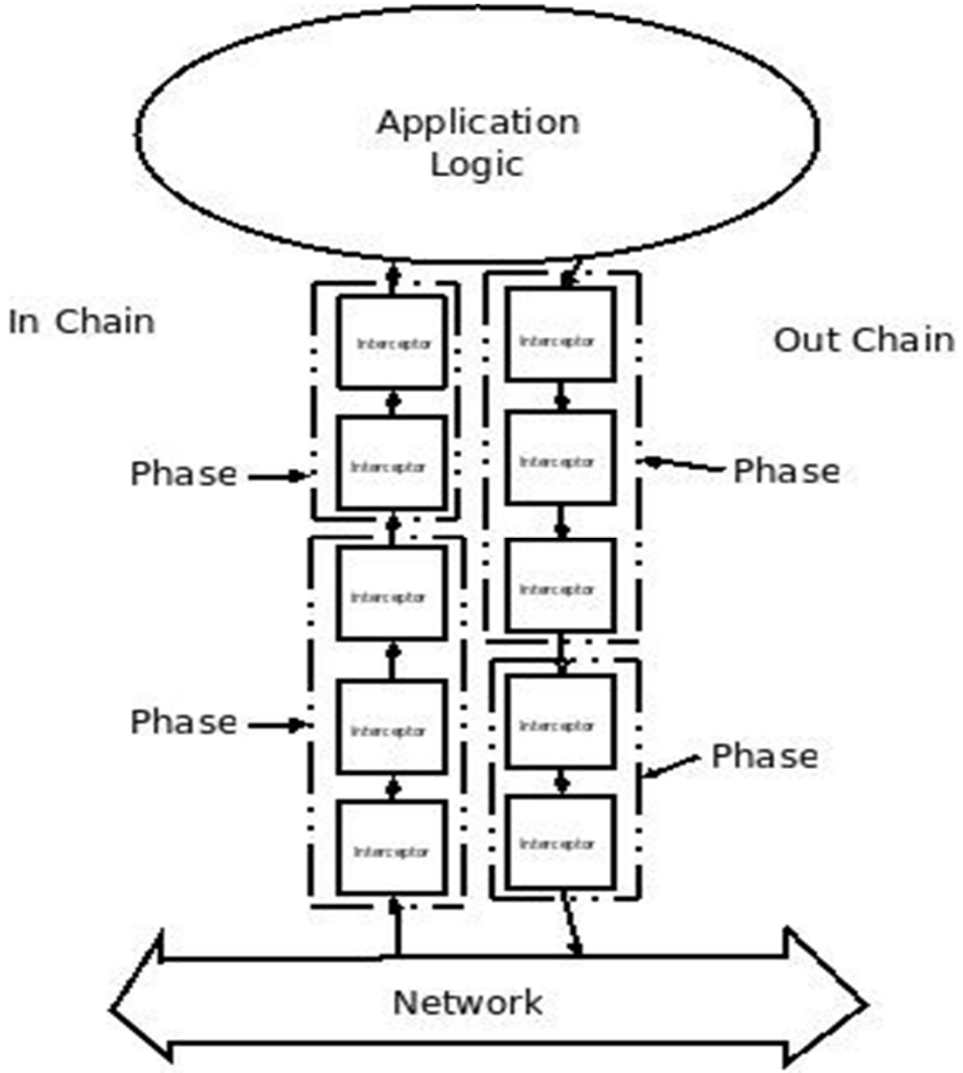


CXF Frontends

1. JAX-WS 2.2
   - Contract first (WSDL); code first
   - Java; Spring; Blueprint based
   - Untyped (Dispatch<>, Provider<>)
   - Dynamic client

2. JAX-RS 1.1, 2.0
   - Contract first (WADL); code first
   - Java; Spring; Blueprint based

Interceptors Chain



Phase Interceptors

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)

Sample Interceptor

```
public class SampleInInterceptor extends
       AbstractPhaseInterceptor<Message> {
    public AttachmentInInterceptor() {
        super(Phase.RECEIVE);
        getAfter().add(SomeOtherInterceptor.class.getName());
    }
    public void handleMessage(Message message) {
        // Process message
    }
    public void handleFault(Message messageParam) {
        // Process fault
    }
}
```

CXF Interceptors: Configuration

1. Programmatically

```
MyInterceptor myInterceptor = new MyInterceptor();
FooService client = ... ; // created from generated JAX-WS client
Client cxfClient = ClientProxy.getClient(client);
cxfClient.getInInterceptors().add(myInterceptor);
```

2. Using Spring/Blueprint configuration

```
<jaxws:endpoint xmlns:tns="http://wsdl.first.soa.symposium/"
   implementor="#HelloWorldService"
   endpointName="tns:HelloWorldServiceEndpoint"
   serviceName="tns:HelloWorldService"
   address="http://localhost:8888/HelloWorld">
        <jaxws:inInterceptors>
                <bean class="demo.interceptor.MyInterceptor" />
        </jaxws:inInterceptors>
</jaxws:endpoint>
```

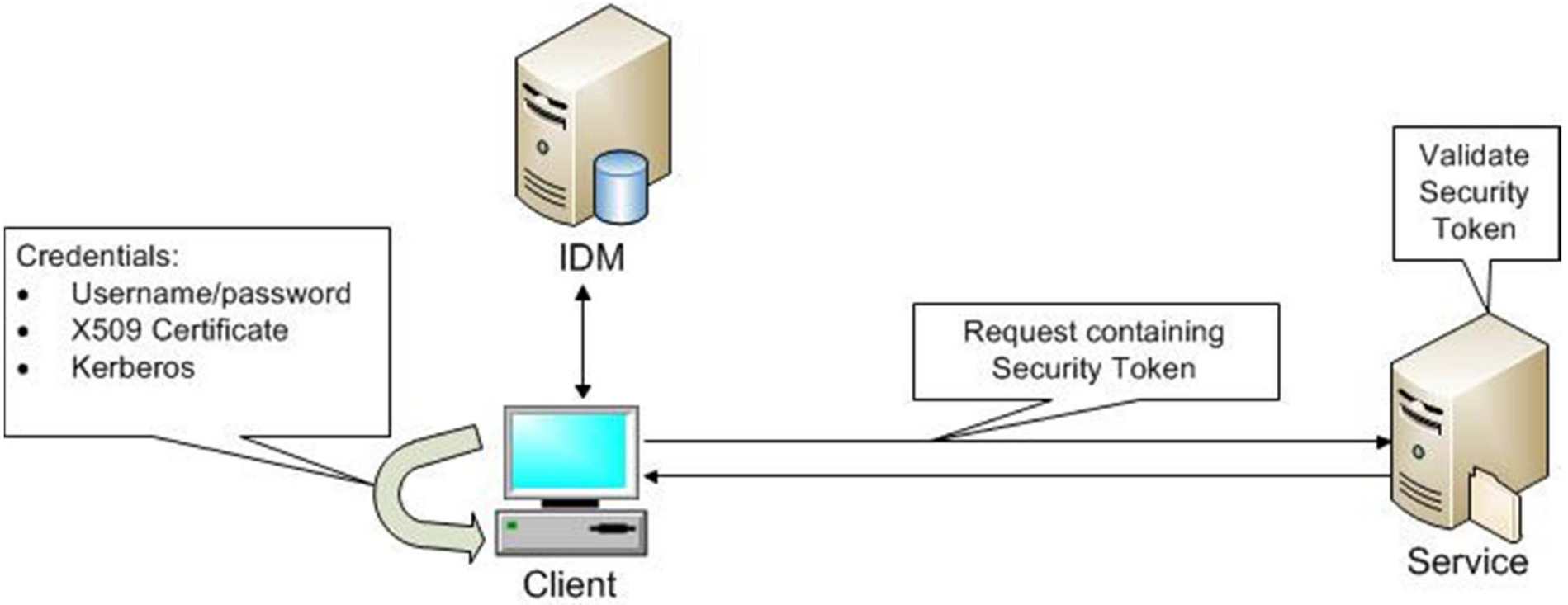
3. Dynamically using WS-Policy

4. Features

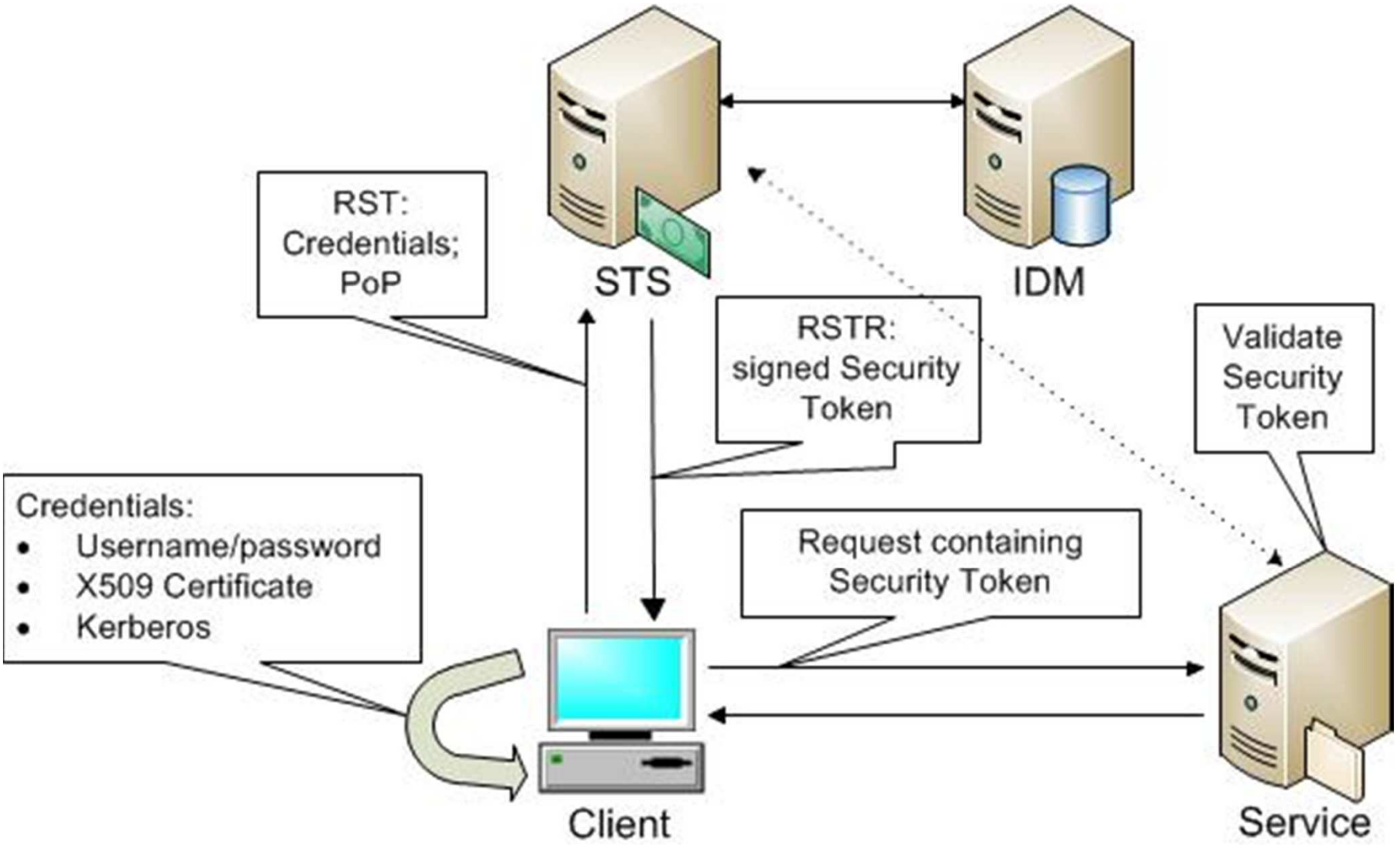
CXF Security

- Authentication (HTTP basic, UsernameToken, X500, Kerberos); WS-Trust 1.4: STS, SAML 2.0; JAAS
- Authorization (SimpleAuthorizingInterceptor, XACML*)
- Encrypt message and parts of message
- Sign message and message elements
- Timestamp message
- WS-SecurityPolicy: bindings, IssuedToken
- PKI support: XKMS service
- Transport level security

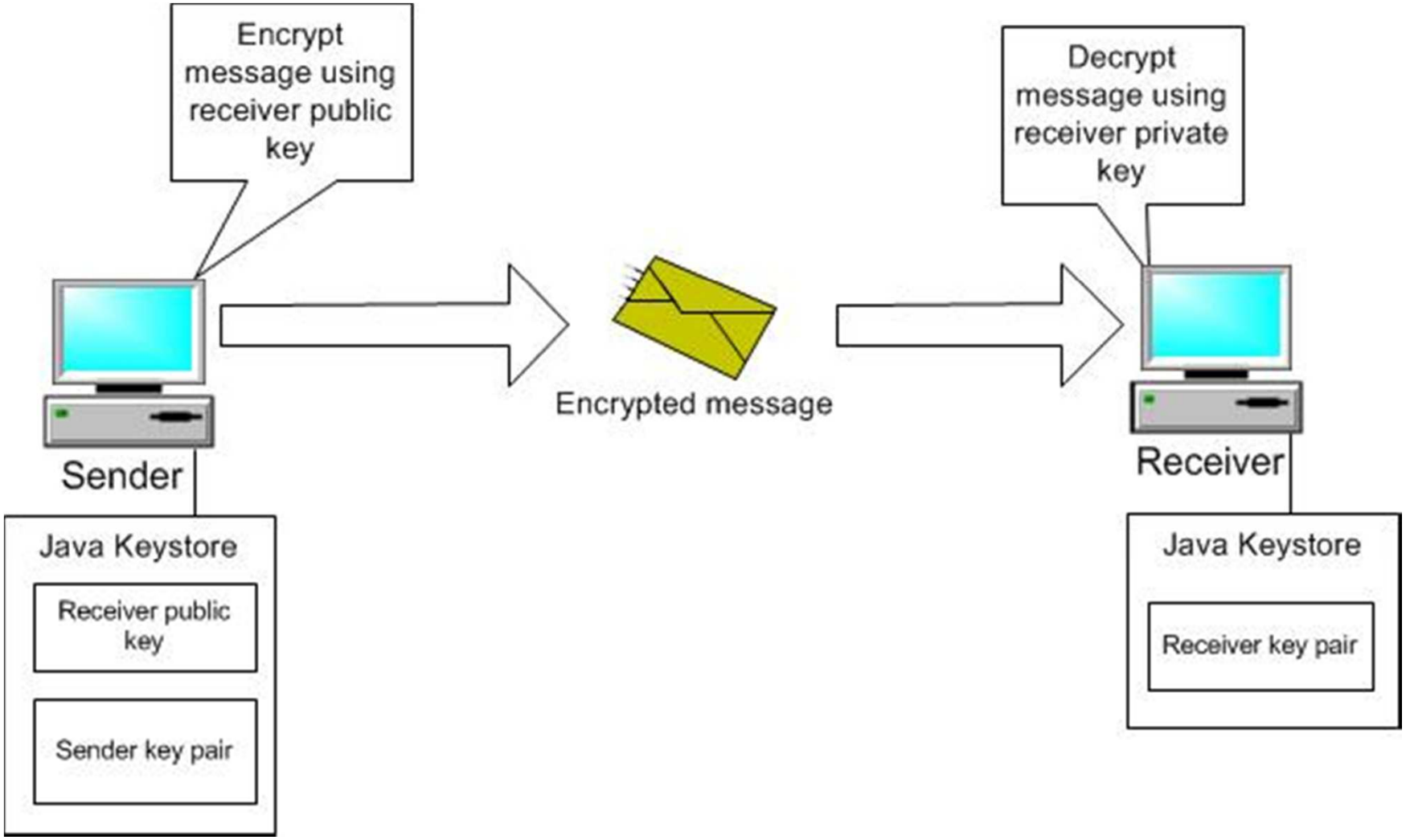
CXF Security: Security Token Service



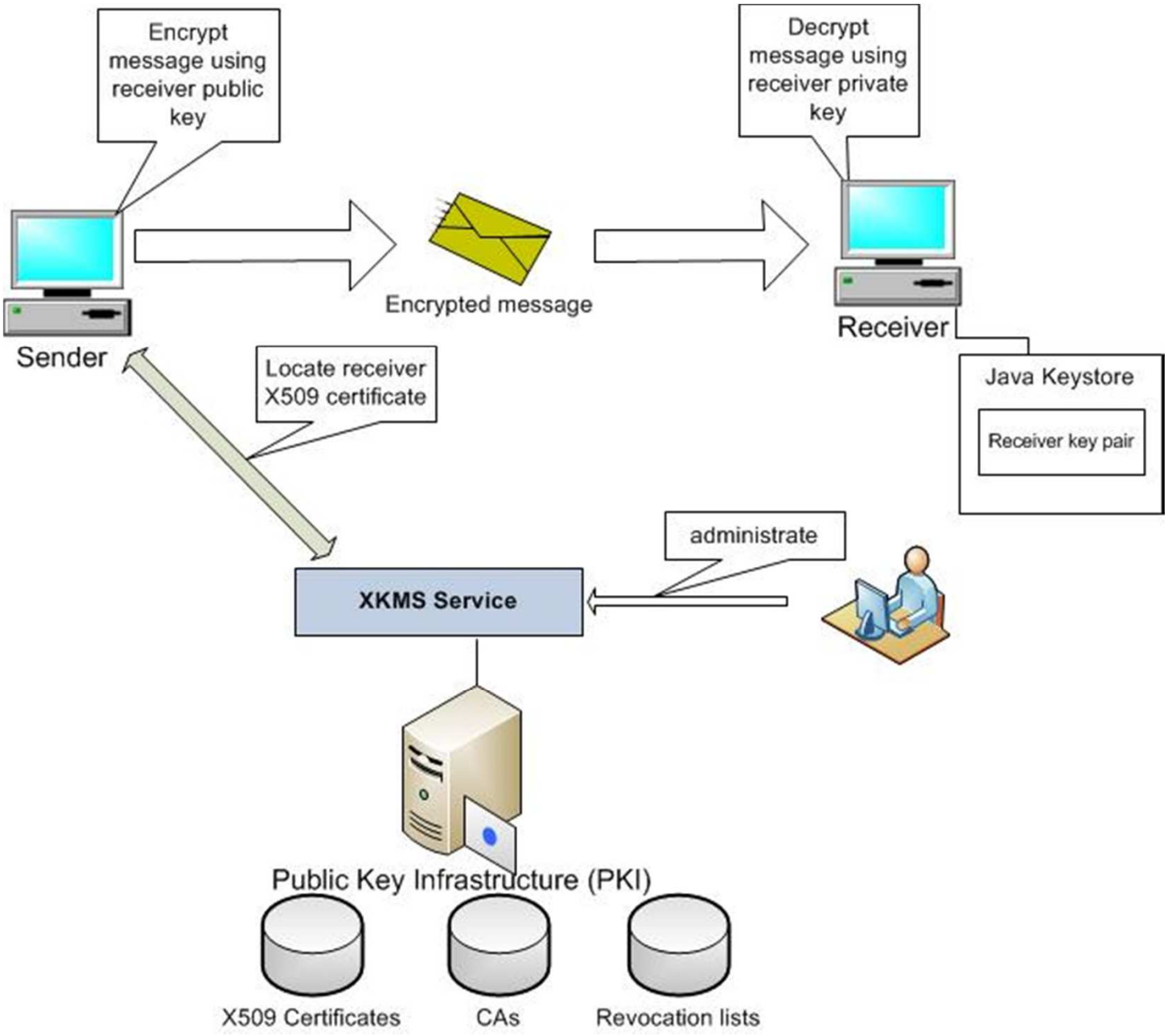
CXF Security: Security Token Service (2.5.x)



CXF Security: XKMS (3.0.0)



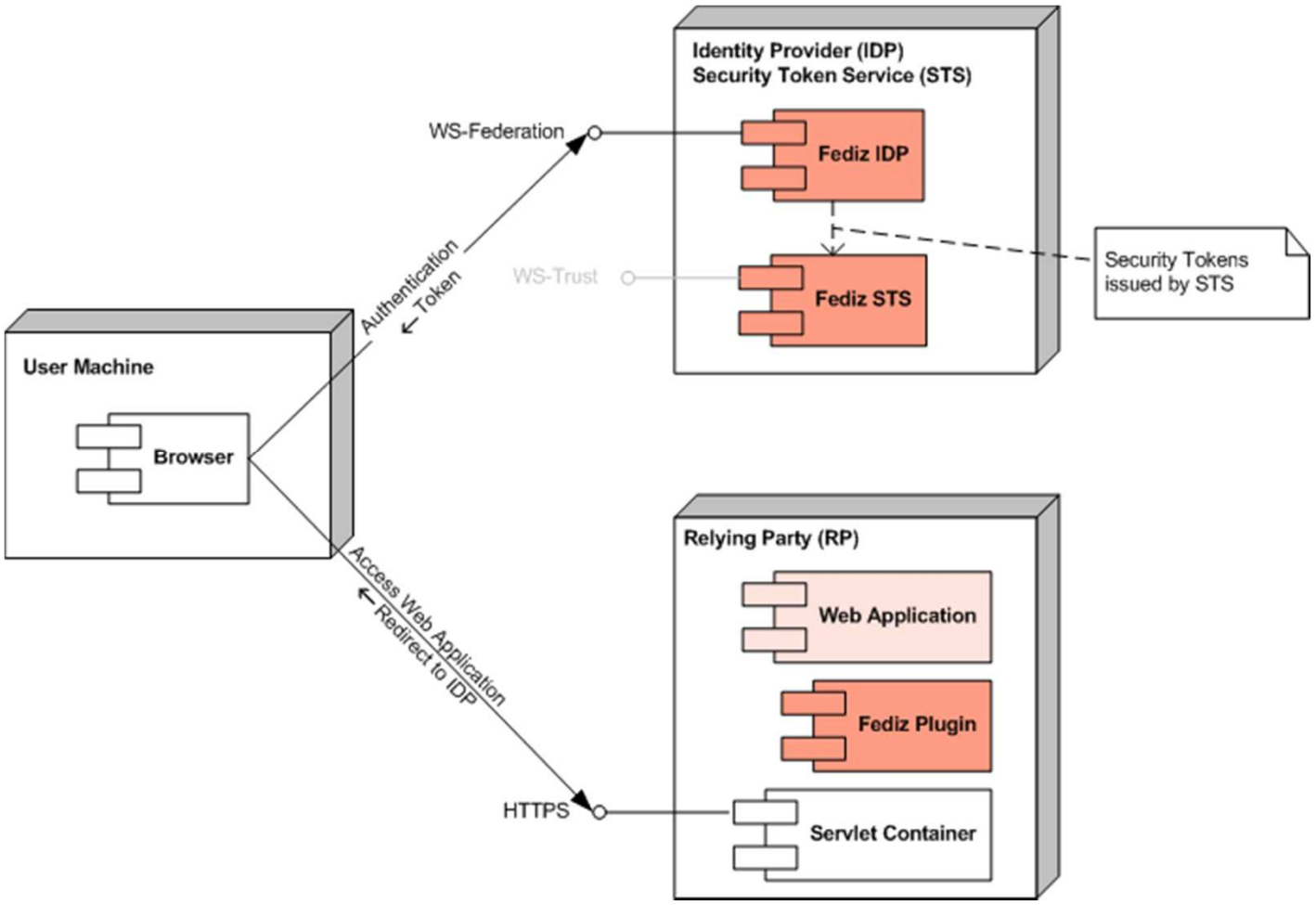
CXF Security: XKMS



CXF Security: Rest Security

- Transport level security - SSL
- OAuth 1.0, 2.0
- Handled by App Server, Spring Security framework: customization
- XML message protection
- Propritary extensions: SAML based authentication
- Fediz project (WS-Federation Passive Requestor profile)

CXF Security: Fediz Subproject



CXF Deployment

- Standalone
- JEE (Servlet Container): Tomcat, JBoss, Glassfish, ...
- JEE (JCA): JBoss, WebSphere, WebLogic, ...
- OSGi: Equinox, ApacheFelix, Karaf

Version 2.6:

1. CXF is splitted to different OSGi bundles
2. Configuration Admin: work queues, keystores, http conduits

CXF 3.0.0

- WSS4J 2.0
- Certified JAX-RS 2.0
- XKMS 2.0
- WS-Eventing
- Internal refactorings and simplificationts

Custom Project: Rudi

Idea: Dynamic multi-domain SOAP/Rest Services Platform for navigation and cartography purposes (military area)

Team: 5 persons

Terms: January 2011 – Mai 2012

Methodology: Scrum

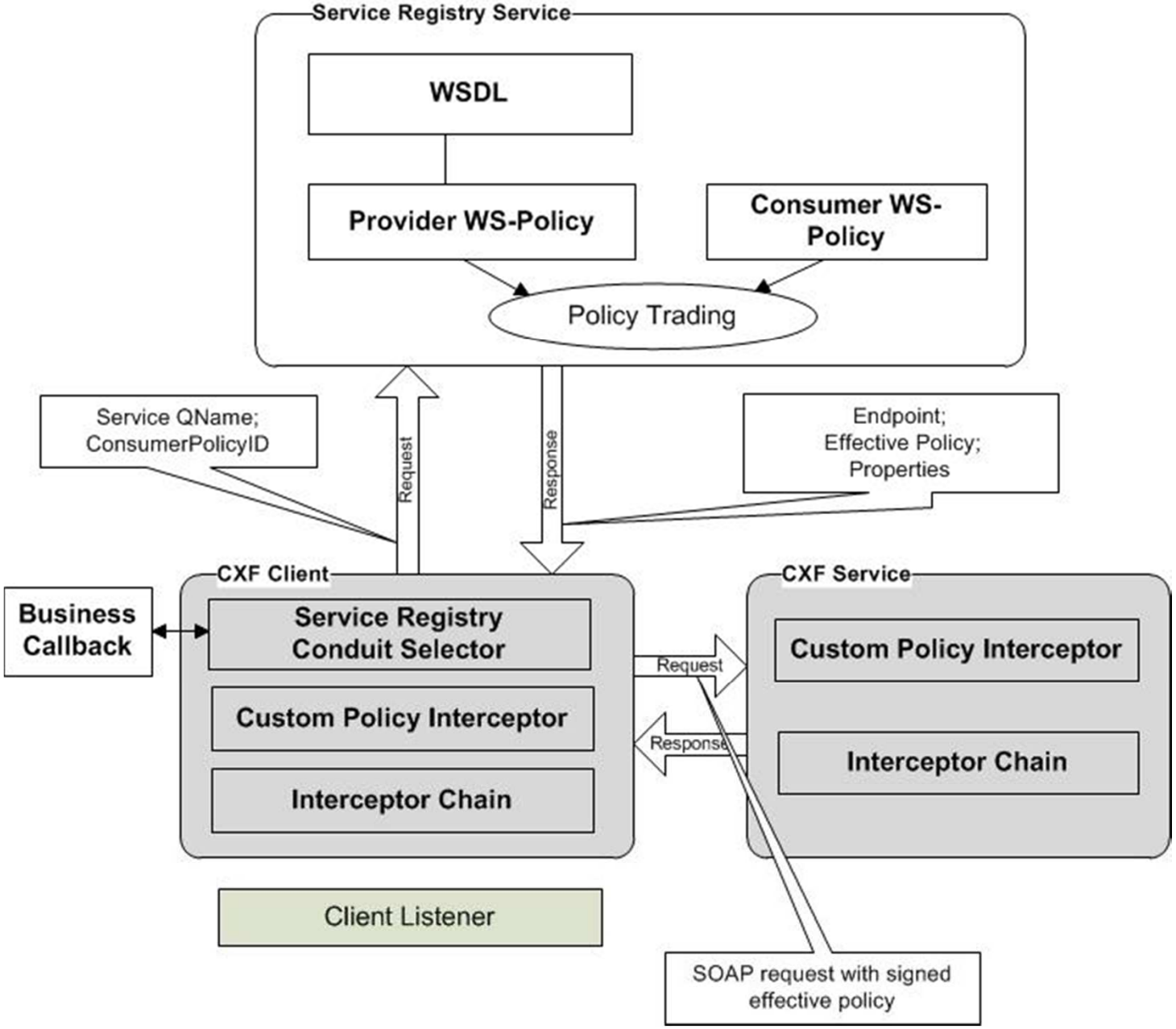
Customer: IABG

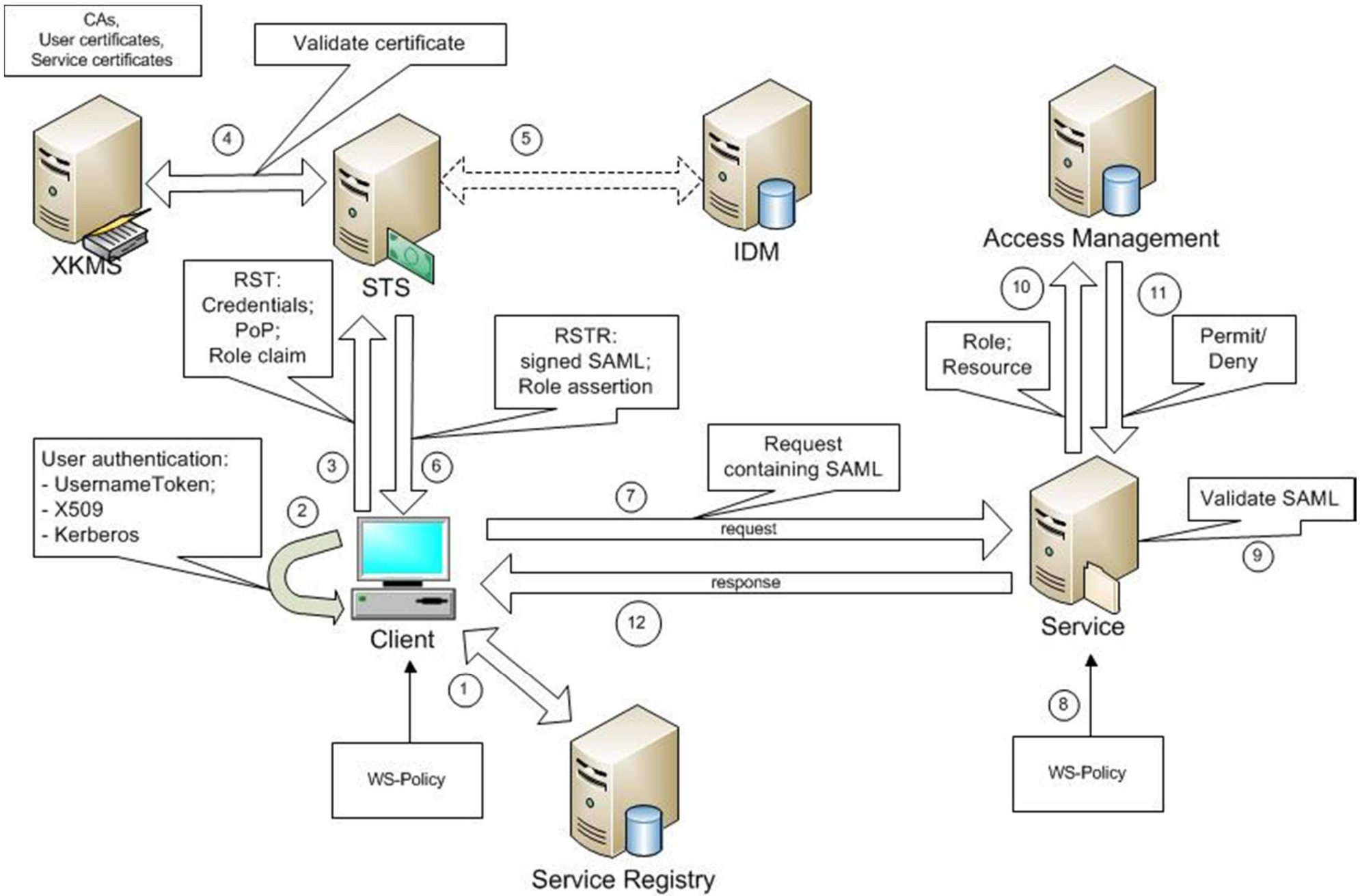
Custom Project: Rudi

Requirements:

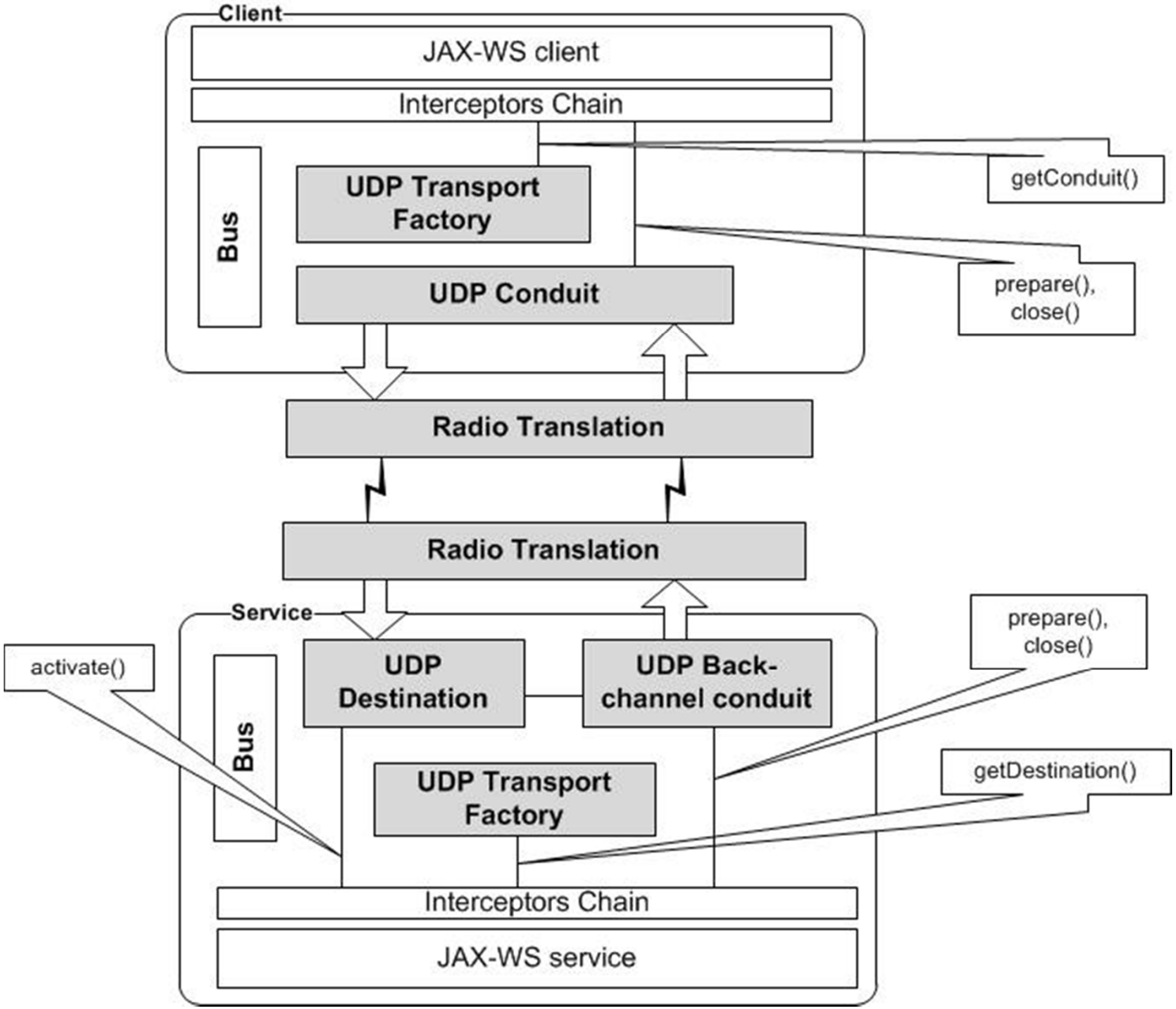
- Dynamic service resolving based on WS-Policy
- SAML based federated authentication
- Role based authorization
- Message level security
- Custom messages compression
- Custom transport (SOAP over UDP)
- OSGi (radar devices, dynamic updates)

Dynamic Service Resolving





UDP Transport (Radio translation based)



CXF Community

Getting involved

- Creating JIRA entries
- Submitting patches
- Participate in mailing list
- Improve Wiki documentation
- Publishing articles

CXF Community

Roles

- User
- Developer (Contributor)
- Committer
- PMC Member
- PMC Chair
- ASF Member (Board Directors)

Links

- http://cxf.apache.org - Apache CXF Project
- http://www.talend.com/ - CXF based Open Source ESB
- http://coheigea.blogspot.de/ - CXF Security
- http://sberyozkin.blogspot.de/ - CXF Rest
- http://ashakirin.blogspot.de/ - Policies, transport, XKMS