Fediz OIDC – CXF Powered OpenId Connect Server

Sergey Beryozkin
Dr Colm O hEgeartaigh

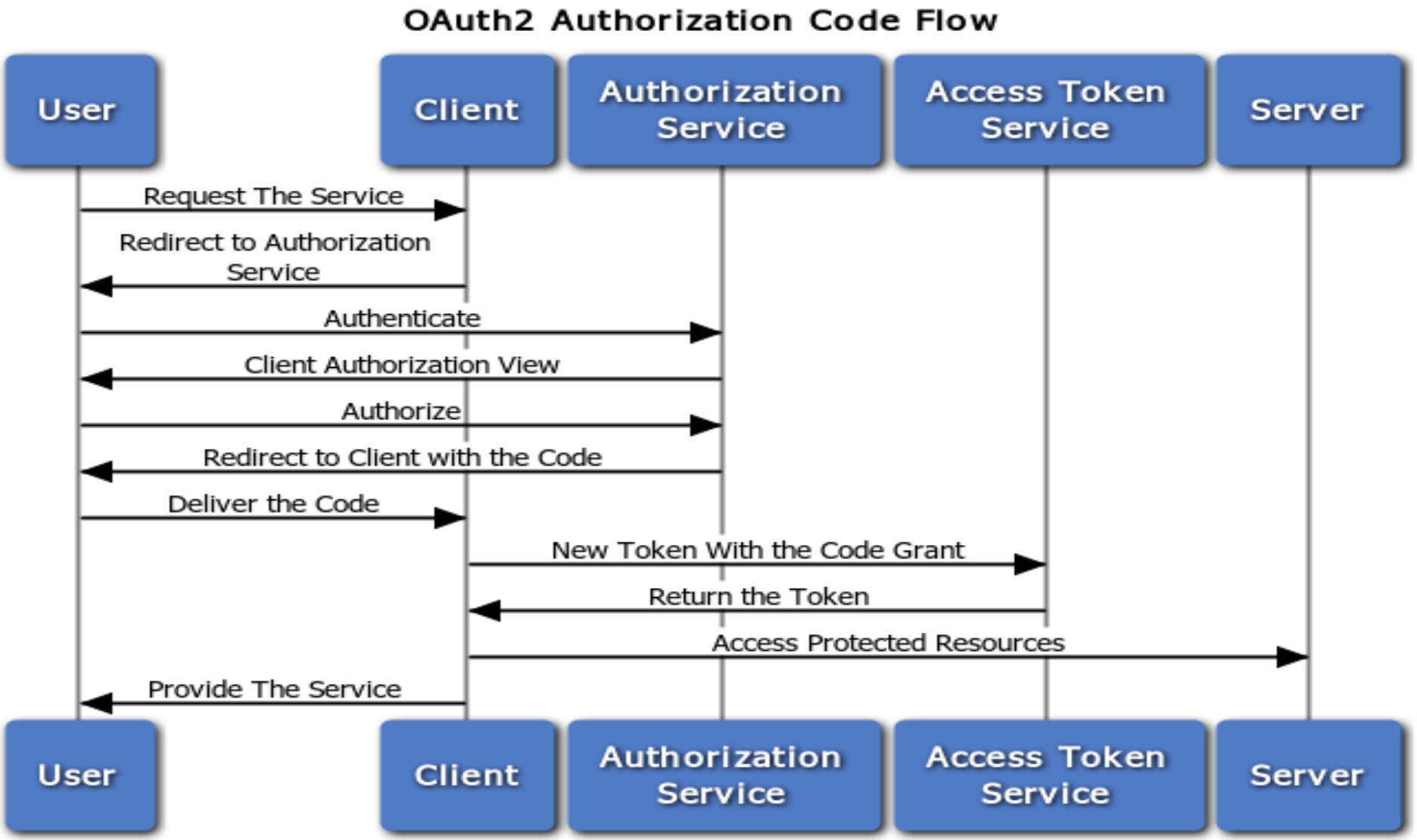
Talend

Introduction to Apache CXF

- Production quality framework for creating Java JAX-RS 2.0 and JAX-WS services
- Widely used and integrated into various containers
- Incubated in 2006, graduated in 2008, going strong in 2016
- Major focus on the security: WS, OAuth2, JOSE, OIDC
- Master: Java 8, JAX-RS 2.1 draft, regular improvements and bug fixes work
- Main subprojects: Fediz, DOSGi

What is OAuth2

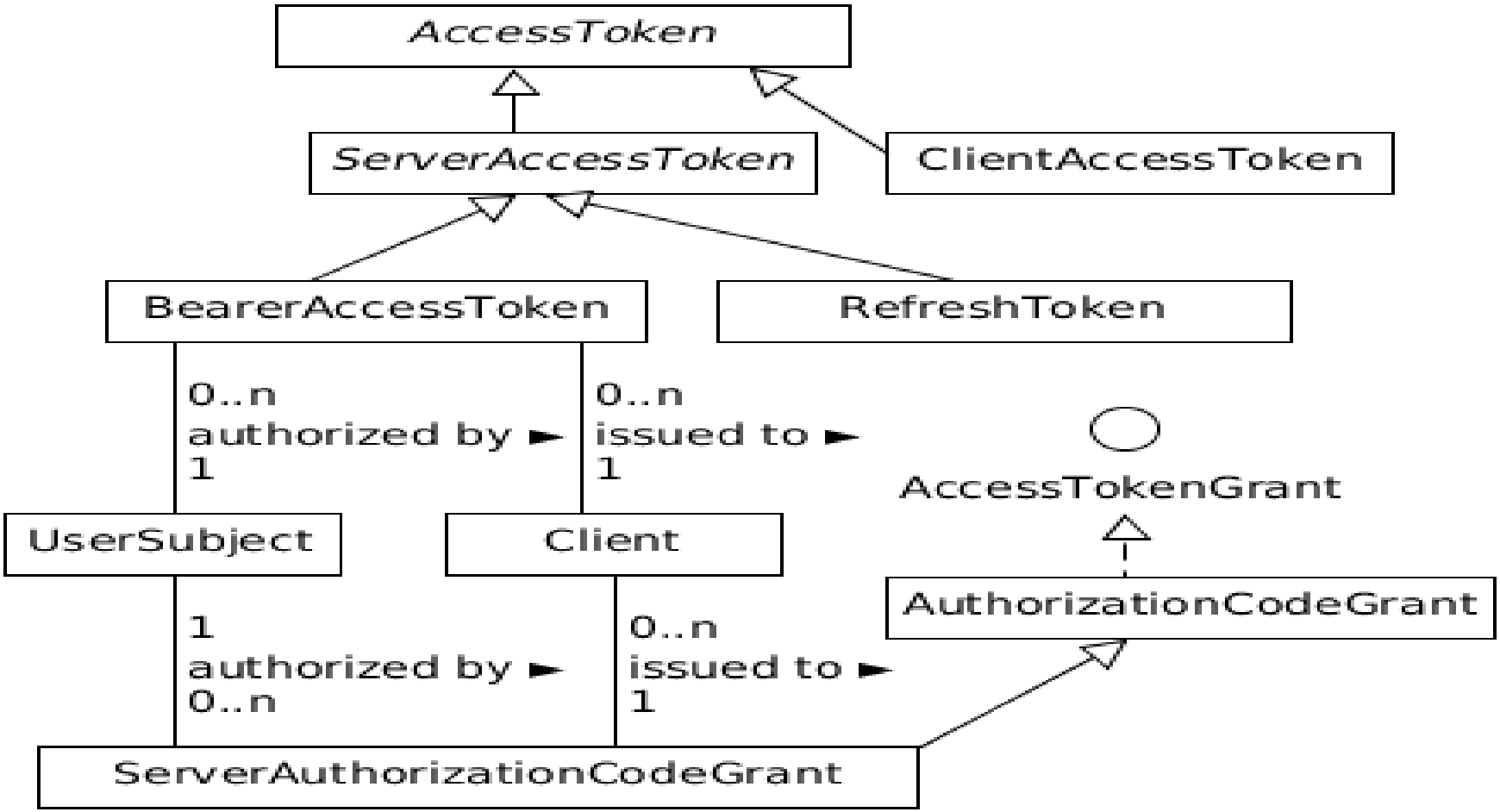
- Protocol for authorizing an access to the resource server (RS)
- Flows supporting human and non-human users are available
- Typical flow: user asks an application acting as an OAuth2 client to do something on RS hosting the user resources, the client redirects the user to OAuth2 authorization service, the user authorizes the client and is returned back to it, the client acquires a time limited access token and uses it to access RS
- Foundation for OpenId Connect


OAuth2 Authorization Code Flow
User
Client
Authorization Service
Access Token Service
Server
Request The Service
Redirect to Authorization Service
Authenticate
Client Authorization View
Authorize
Redirect to Client with the Code
Deliver the Code
New Token With the Code Grant
Return the Token
Access Protected Resources
Provide The Service
User
Client
Authorization Service
Access Token Service
Server
www.websequencediagrams.com

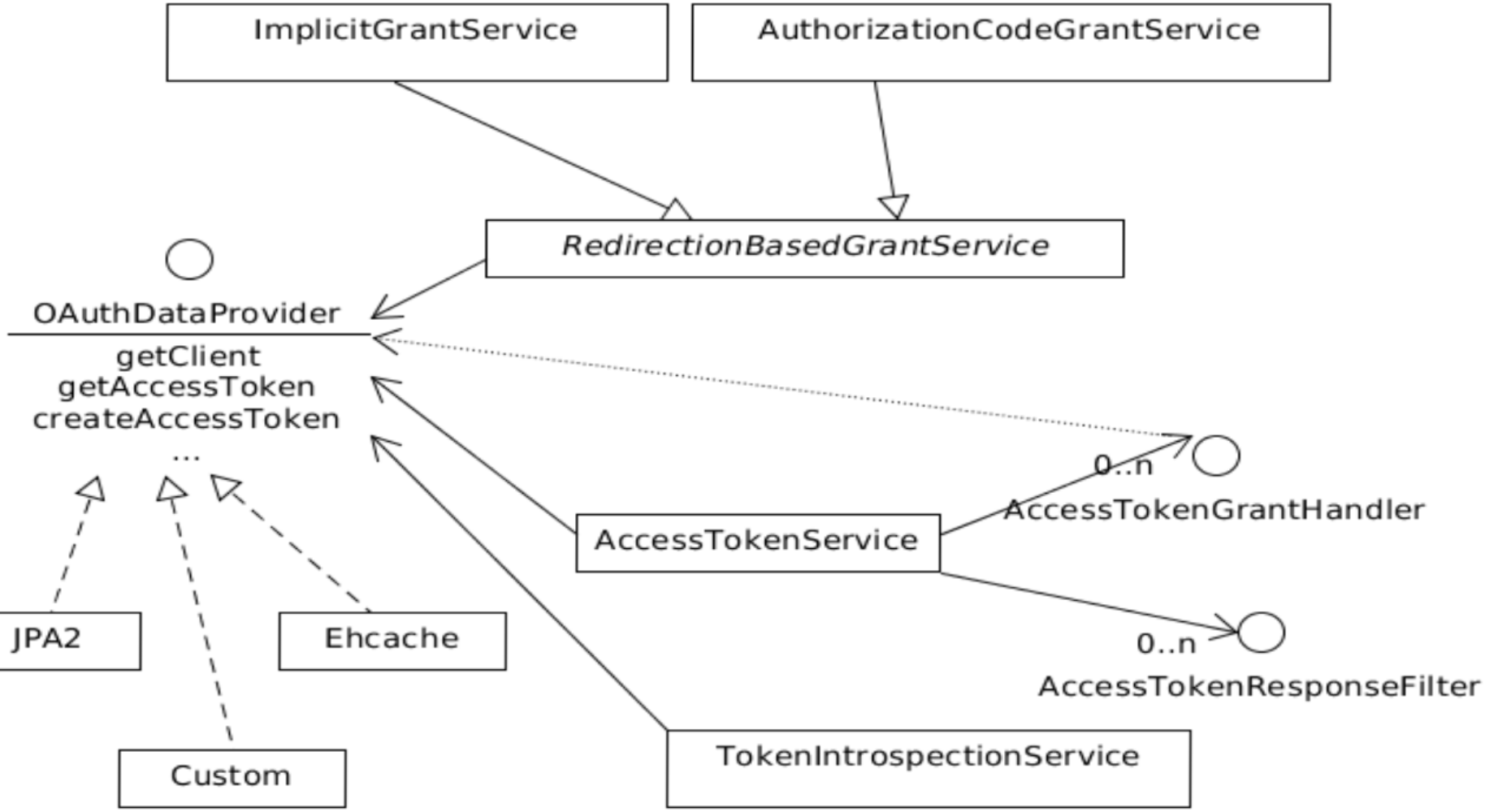
CXF and OAuth2

- Comprehensive OAuth2 Data and Service Model
- Goal: minimize the exposure to OAuth2 specifics, let developers focus on persisting the data model only
- Authorization Code and Implicit redirection services
- Support for the well-known and custom token grants
- Bearer, Hawk, custom access tokens, Refresh Tokens
- EHCache, JCache, JPA2 data providers OOB.
- New: EHCache and JCache providers support AT in JWT
- New: DynamicRegistration and (.well-known) Configuration services

CXF OAuth2 Data Model



CXF OAuth2 Service Model



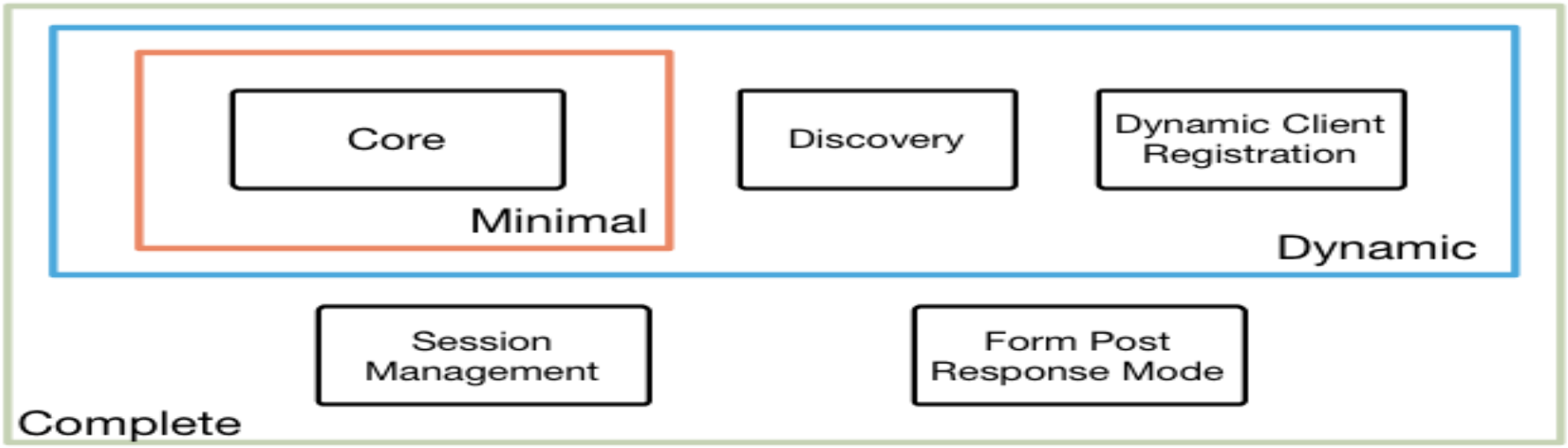
What is OpenId Connect (OIDC)

- Identity layer built on top of OAuth2 and heavily depending on JOSE
- User authentication info is available in IdToken – crypto-protected Json Web Token (JWT)
- Code flow extends the OAuth2 code flow by returning IdToken in the access token response
- Implicit flow is different from the OAuth2 Implicit flow as only IdToken is returned to the browser/mobile client
- Hybrid flow combines code and implicit flows
- Client uses IdToken to work with the user and optionally AccessToken to access this user's resources elsewhere

4 Feb 2014

OpenID Connect Protocol Suite

http://openid.net/connect



Underpinnings

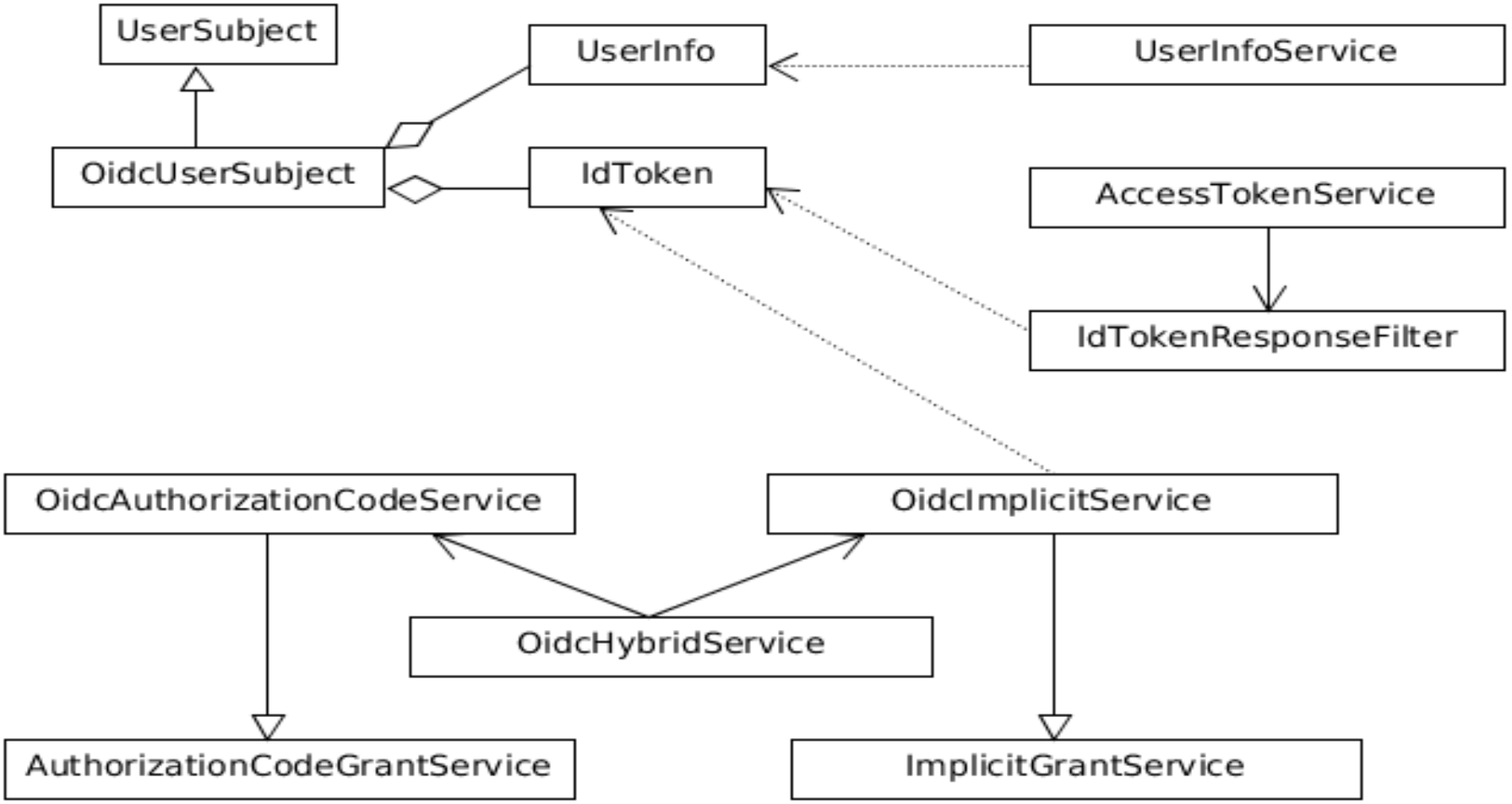
What is JOSE

- Set of standards for creating cryptographically protected compact or JSON containers for the arbitrary data formats
- JWS – signed data, JWE – encrypted data, JWK – secure key representation in JSON
- JWT: JSON Web Token which is a standard JSON where each top level property key is a 'claim'.
- Example: OIDC IdToken – JWT with claims such as the user name, etc which is typically signed by an OIDC private key (RS256, etc) or by a shared secret key (client secret allocated during the client registration – HS256, etc) with JWS Compact sequence being produced. Client will validate with OIDC public key, etc

CXF and OIDC

- Services model which builds upon CXF OAuth2 and JOSE code
- OIDC Code and Implicit Services are OAuth2 services with simple extensions (example, IdToken is added to AccessToken responses, input parameters go through the extra validation, etc). Hybrid service combines the two. UserInfoService returns more info about the user.
- New: DynamicClientRegistration and (.well-known) Configuration services
- IdToken, JWKs are signed or encrypted with CXF JOSE
- Advanced CXF OIDC RP (client) support

CXF OIDC Service Model



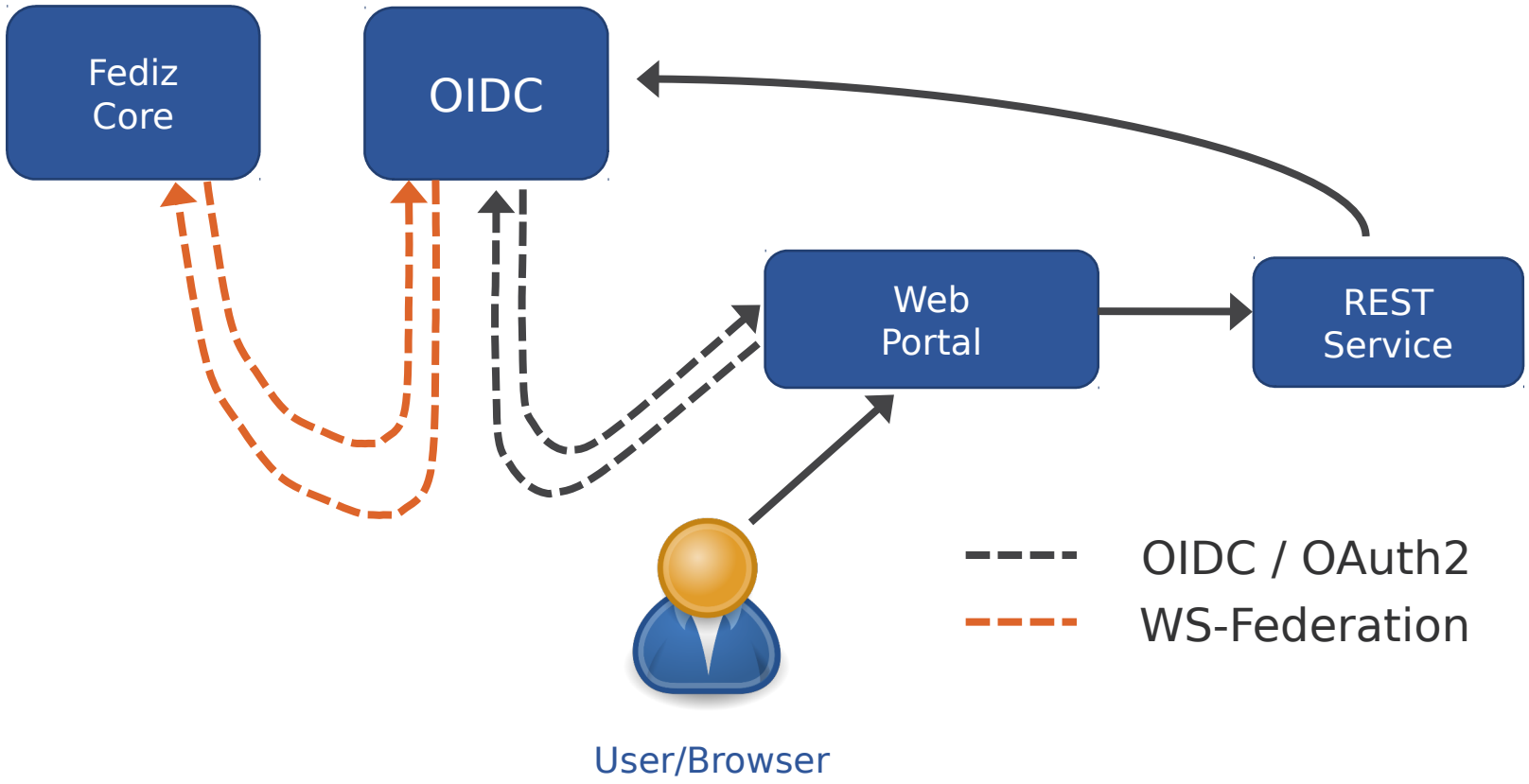
Introduction to Apache CXF Fediz

- The initial goal was to provide an open source implementation of the WS-Federation SSO protocol
- Concrete use case: enabling SSO for web applications against an ADFS Identity Provider (IDP)
- Added as a new CXF subproject in Dec 2011, first release – June 2012
- Offers a flexible local or trusted provider authentication support
- Deployed in concrete productions
- WS-Fed, SAML2 SSO and finally OpenId Connect

What is Fediz OIDC

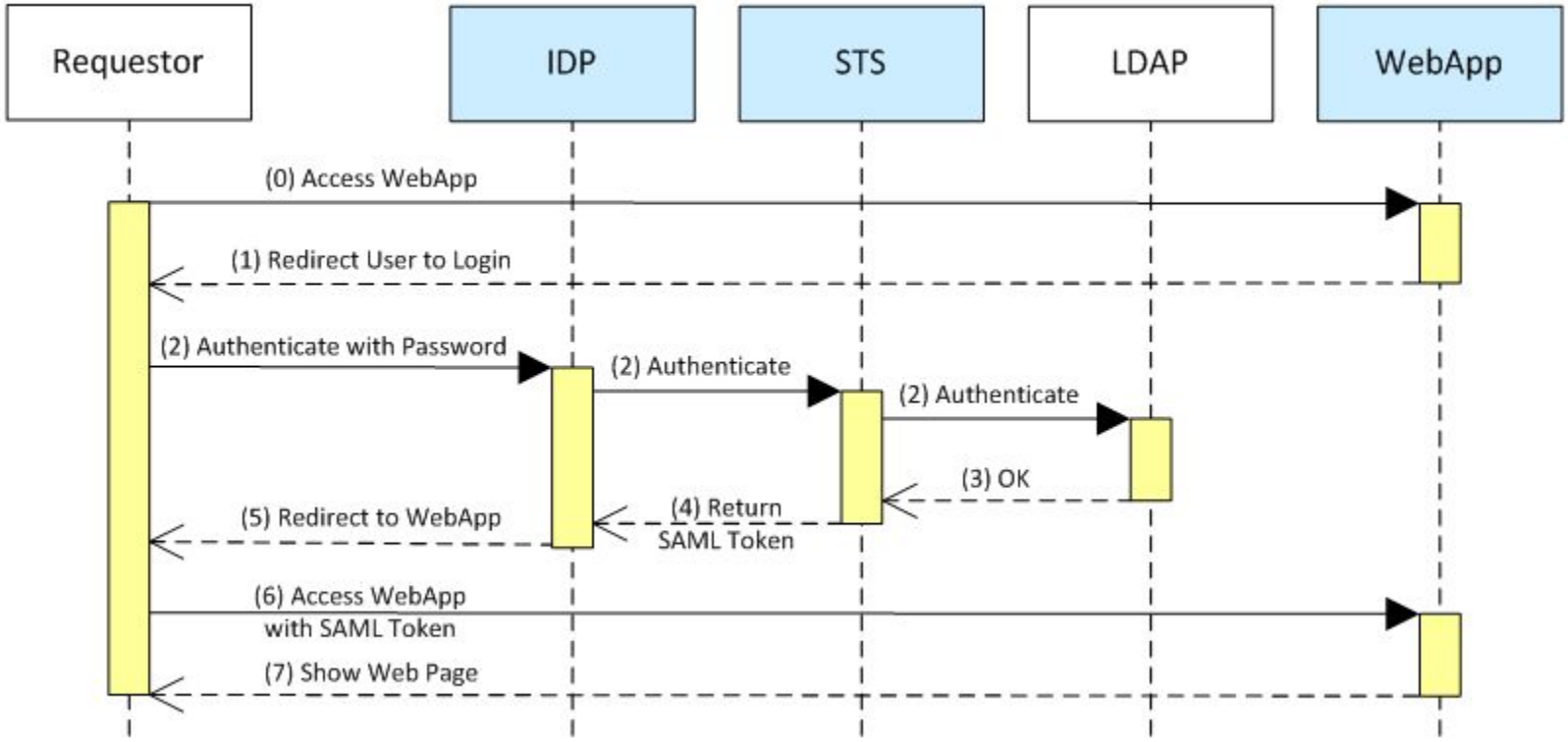
- Fediz OIDC = Fediz Core + CXF OIDC (OAUTH2, JOSE)
- Young project but already in the production
- Perfect Demonstration of the Fediz flexibility and CXF OIDC independence
- Initial Implementation: CXF OIDC JAX-RS service endpoints, default EHCache data provider, simple support for registering the clients and managing the client registrations and access tokens
- All packaged as a WAR which temporarily acts as a Fediz WS-Fed Relying Party application. WS-Fed connector ensures OIDC sees a user's SAML token which it converts to IdToken
- Code...

WS-Fed Bridge between OIDC and Core IDP



How Fediz Authentication works

- Here the Fediz OIDC is acting as the Requestor



Fediz Plugins

- Core Plugin component:
  - Code for creating/validating WS-Federation requests
  - XML Schema for configuring the plugins
- Specific container based plugins:
  - Integrate with specific container architecture
  - Responsible for performing the redirects, setting up the security context etc.
- Containers supported:
  - Jetty 8/9
  - Tomcat 7/8
  - Spring Security
  - Websphere
  - CXF

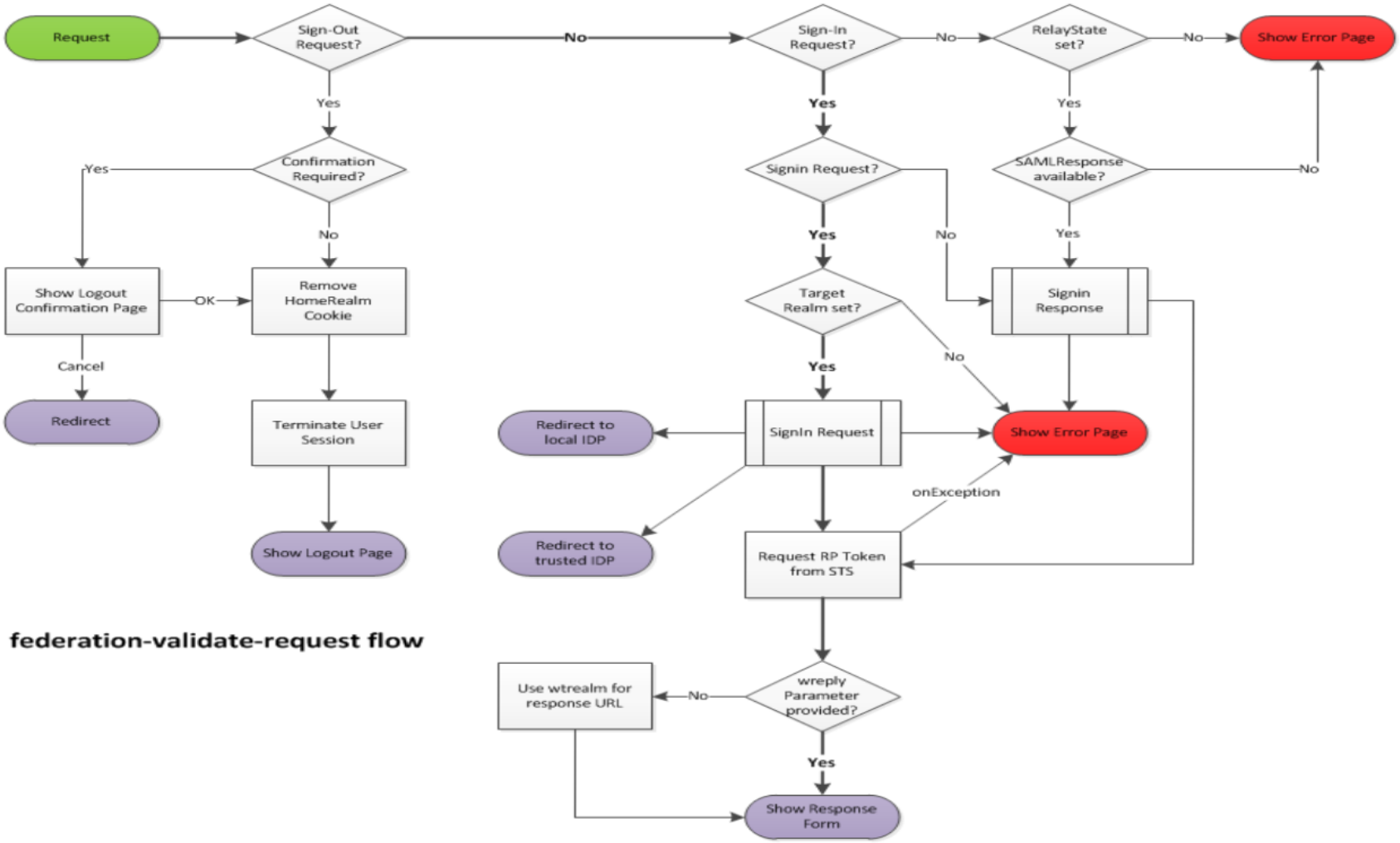
Fediz Plugin Configuration

```
<FedizConfig>
 <contextConfig name="/fedizhelloworld">
  <audienceUris>
   <audienceItem>urn:org:apache:cxf:fediz:fedizhelloworld</audienceItem>
  </audienceUris>
  <certificateStores>
   <trustManager>
    <keyStore file="ststrust.jks" password="storepass" type="JKS"/>
   </trustManager>
  </certificateStores>
  <trustedIssuers>
   <issuer certificateValidation="PeerTrust"/>
  </trustedIssuers>
  <maximumClockSkew>1000</maximumClockSkew>
  <protocol xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="federationProtocolType" version="1.0.0">
   <realm>urn:org:apache:cxf:fediz:fedizhelloworld</realm>
   <issuer>https://localhost:9443/fediz-idp/federation</issuer>
   <roleDelimiter>,</roleDelimiter>
   <roleURI>http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role</roleURI>
   <claimTypesRequested>
    <claimType type="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/role" optional="false"/>
    <claimType type="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" optional="true"/>
   </claimTypesRequested>
  </protocol>
 </contextConfig>
</FedizConfig>
```

Fediz IdP

- The Fediz IdP supports both WS-Federation and SAML SSO.
- User authentication and token creation is delegated to the Apache CXF STS via the WS-Trust protocol.
- The IdP was rewritten in the 1.1.0 release to be based on Spring Web Flow and is very extensible as a result.
- Secured via Spring Security
- Authentication methods supported:
  - HTTP/BA
  - TLS client authentication
  - Kerberos

IdP Flow Example



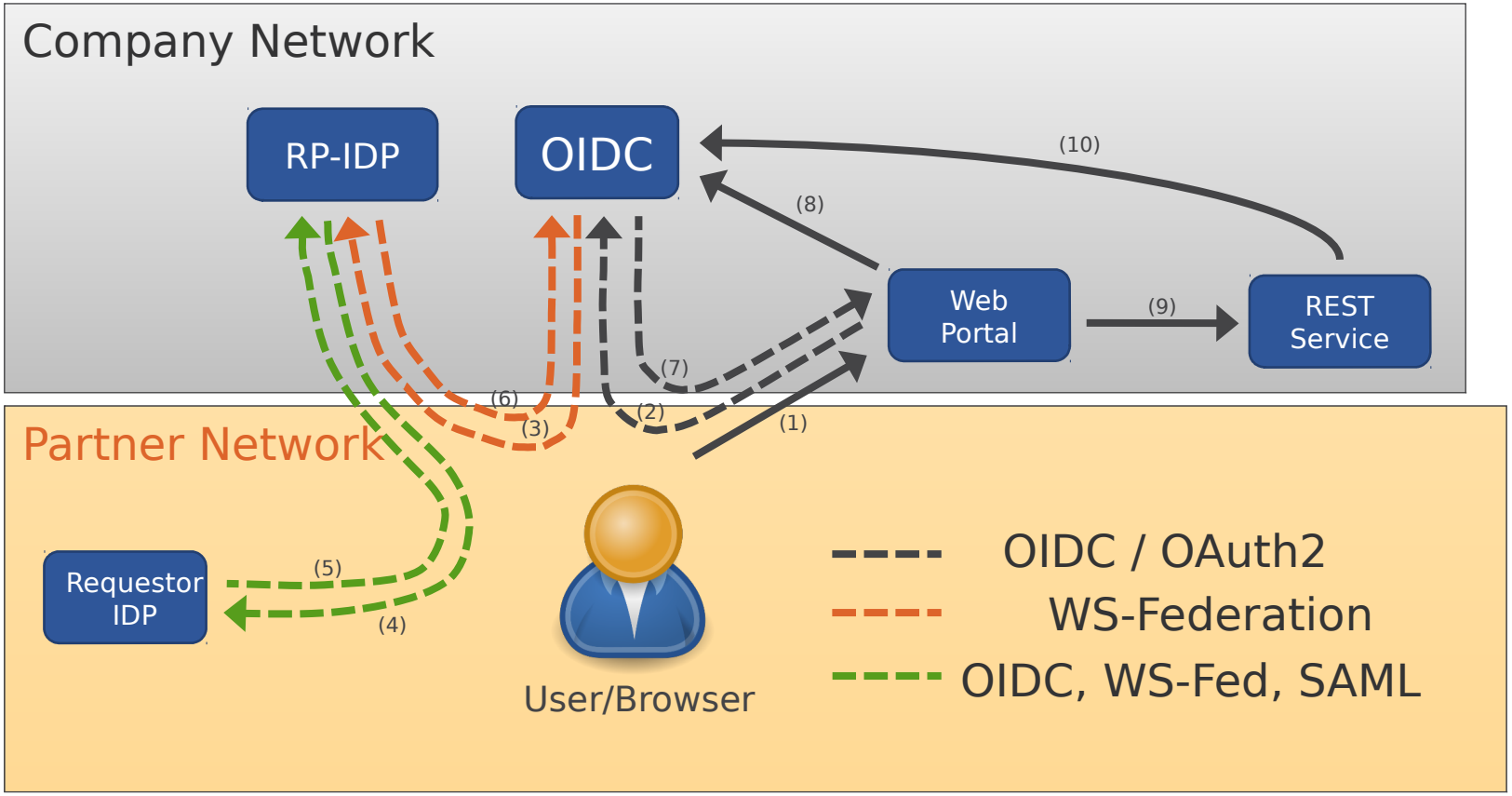
Fediz IdP REST Interface

- New REST configuration interface added to the IdP in 1.2.0:
  - to get current status of a user session (debug purpose)
  - read IDP configurations (trusted IDP, service configs, etc.)
  - update IDP configurations
- Information on the REST API can be accessed via:
  - WADL: services/rs?_wadl
  - Swagger Document: services/rs/swagger.yaml
- Configuration stored in a database and persisted using JPA

Delegation to Trusted Providers

- Since the 1.1 release, the Fediz IdP supports redirecting the user to a trusted third party IdP for authentication
- WS-Federation only supported in 1.1
- From 1.2.0, the concept of protocol bridging is introduced for trusted third parties.
- The admin can “plug in” different protocols for different realms:
  - WS-Federation
  - SAML SSO (1.2.0)
  - OpenId Connect (1.3.0)
  - Facebook (1.3.1)

Delegation to OIDC/WS-Fed/SAML SSO IdP



TrustedIdp example config for OIDC

```
<bean id="trusted-idp-realmF"
    class="org.apache.cxf.fediz.service.idp.service.jpa.TrustedIdpEntity">
    <property name="realm" value="urn:org:apache:cxf:fediz:idp:realm-F" />
    <property name="cacheTokens" value="true" />
    <property name="url" value="https://localhost:${idp.oidc.https.port}/idpoidc/services/authorize" />
    <property name="certificate" value="realmb.cert" />
    <property name="trustType" value="PEER_TRUST" />
    <property name="protocol" value="openid-connect-1.0" />
    <property name="federationType" value="FEDERATE_IDENTITY" />
    <property name="name" value="Realm B" />
    <property name="description" value="Realm F description" />
    <property name="parameters">
        <util:map>
            <entry key="client.id" value="consumer-id"/>
            <entry key="client.secret" value="this-is-a-secret"/>
            <entry key="token.endpoint" value="https://localhost:${idp.oidc.https.port}/idpoidc/services/token"/>
        </util:map>
    </property>
</bean>
```

Demo

Future Plans

- 1.4.0 release due end of year 2016
  - The STS and IdP configuration are substantially refactored to make it easier to customize them.
  - Main new feature is support for delegation to trusted third party IdPs for SAML SSO
- 2.0.0 planned for mid/late 2017
  - The plan is to decouple the OIDC service from the existing WS-Federation authentication mechanism to simplify deployment for OIDC users.
  - The OIDC service will re-use the existing STS component to authenticate users.

Questions ?

- Please visit

  http://cxf.apache.org/fediz-oidc.html

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

- Ask about Fediz and CXF OIDC at users@cxf.apache.org or propose new ideas at dev@cxf.apache.org
- Check our blogs:

  http://coheigea.blogspot.com/

  http://sberyozkin.blogspot.com/

  http://janbernhardt.blogspot.com/

  Thank You !