Archivos de configuración a editar:

CLIENTE:

clientConfig.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<OAuthClient>
     <AuthServerConfig>
          <AssertionType>urn:mace:rediris.es:papi</AssertionType>
          <GrantType>assertion</GrantType>
          <AuthServerURL> /url del AS token endpoint/ </AuthServerURL>
     </AuthServerConfig>
     <ResServerConfig>
          <RequestType>Form-Encoded_Body_Parameter</RequestType>
          <ResServerURL> /url del RS server endpoint/ </ResServerURL>
          <ResponseFormats>
               <Scope id="scope_de_prueba">
                    <FormatClass></FormatClass>
                    <FormatFile></FormatFile>
               </Scope>
          </ResponseFormats>
     </ResServerConfig>
     <ClientConfig>
          <ClientID>prueba_oauth</ClientID>
          <ClientSecret>clave_de_prueba_oauth</ClientSecret>
          <ErrorResponseType>HTML</ErrorResponseType>
          <DefaultScope>scope_de_prueba</DefaultScope>
          <DebugActive>TRUE</DebugActive>
          <StorageType>session</StorageType>
     </ClientConfig>
</OAuthClient>
```

Contiene la configuración del cliente, dividida en 3 partes:

- `AuthServerConfig` : configura el servidor de autorización en el cual el cliente ha sido registrado. Parámetros a configurar:

    - `AssertionType` - puede tomar dos valores:

        · ***urn:mace:rediris.es:papi*** , si la aserción que va a usar el cliente para obtener el token de acceso es de tipo PAPI.

        · ***urn:oasis:names:tc:SAML:2.0:assertion*** , si la aserción que va a usar el cliente para obtener el token de acceso es de tipo SAML2.

    - `GrantType` - debe de tener el valor ***assertion***.

    - `AuthServerURL` - la URL del token endpoint del Servidor de Autorización.

- `ResServerConfig` : configura el Servidor de Recursos(RS) al cual el cliente, haciendo uso del token de acceso obtenido de un AS, va a pedir un recurso, determinado por su ***scope***.

    - `RequestType` - puede tomar tres valores, que definen tres métodos de enviar el token de acceso (de tipo Bearer token) en la petición del recurso al RS:

        · ***HTTP_Authorization_Header*** : hace la petición del recurso metiendo el token de acceso en el campo *Authorization* de las cabeceras http.

        · ***Form-Encoded_Body_Parameter*** : hace la petición del recurso metiendo el token de acceso en el Body (POST) de la petición http.

        · ***URI_Query_Parameter*** : hace la petición del recurso metiendo el token de acceso en la URI de la petición HTTP (GET).

    - `ResServerURL` - la URL del server endpoint del Servidor de Recursos.

    - `ResponseFormats` - Define las clases que, por reflexión, darán formato al recurso devuelto por el RS. Cada **scope**, definido por su identificador **id**, debe tener un `FormatClass`, con el nombre de la clase que da formato a ese **scope**, y un `FormatFile` con el nombre del archivo que contiene la clase.

- `ClientConfig` : configuración del cliente.

    - `ClientID` - Identificador del cliente. Su valor debe ser suministrado por el AS en el momento del registro del cliente en él.

    - `ClientSecret` - Clave del cliente.

    - `ErrorResponseType` - Puede ser **HTML** o **JSON.**

    - `DefaultScopeStorageType` - El scope por defecto que usa el ciente para pedir un token/recurso.

    - `StorageType` - La forma en que guarda el cliente el token. Todas ellas usan el uid del cliente como clave de almacenamiento. Puede ser:

        · **sesión**, si se quiere guardar el token en la variable $_SESSION.

        · **db**, si se quiere guardar el token en una base de datos tipo dba.

        · **file**, si se quiere guardar el token de acceso en un fichero de texto plano.

    - `DebugActive` - Si está a **TRUE**, se guarda una traza de la petición.

SERVIDOR DE AUTORIZACIÓN (AS):

clientKeys.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Clients>
     <Client id="prueba_oauth">
          <TokenLifetime>3600</TokenLifetime>
          <Key>clave_de_prueba_oauth</Key>
          <GenerateAccessTokenMethod>AS</GenerateAccessTokenMethod>
          <AllowedScopes>
               <Scope id="scope_de_prueba"/>
               <Scope id="scope_de_prueba2"/>
          </AllowedScopes>
     </Client>
     <Client id="prueba_oauth_SAML">
          <TokenLifetime>3600</TokenLifetime>
          <Key>clave_de_prueba_oauth_SAML</Key>
          <GenerateAccessTokenMethod>AS</GenerateAccessTokenMethod>
          <AllowedScopes>
               <Scope id="scope_de_prueba_SAML"/>
          </AllowedScopes>
     </Client>
</Clients>
```

Contiene la configuración de todos los clientes registrados en el Servidor de Autorización:

- `id` del `Client` - Es el `ClientID` guardado en la configuración de ese cliente.

- `TokenLifetime` -Tiempo en segundos para el cual el token es válido.

- `Key` - El `ClientSecret` guardado en la configuración del cliente.

- `GenerateAccessTokenMethod` –

    · ***AS*** : si el token de acceso es generado por el AS.

    · ***STS*** : si el AS delega en un STS la generación del token.

- `AllowedScopes` – Los scopes ,para los cuales, ese cliente tiene registrados que se pueda emitir un token.

serverKeys.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<AuthServer id="authserver_example" url="url AS token endpoint">
     <ResourceServers>
          <ResourceServer id="prueba_oauth">
               <Scopes>
                    <Scope>scope_de_prueba</Scope>
                    <Scope>scope_de_prueba_SAML</Scope>
               </Scopes>
               <Key>oauth_server_key</Key>
          </ResourceServer>
          <ResourceServer id="prueba_oauth2">
               <Scopes>
                    <Scope>scope_de_prueba2</Scope>
               </Scopes>
               <Key>oauth_server_key2</Key>
          </ResourceServer>
          <ResourceServer id="prueba_oauth3">
               <Scopes>
                    <Scope>scope_de_prueba3</Scope>
               </Scopes>
               <Key>oauth_server_key3</Key>
          </ResourceServer>
     </ResourceServers>
</AuthServer>
```

Contiene la configuración de todos los Servidores de Recursos registrados en el Servidor de Autorización:

- `AuthServer` - contiene los atributos `id`, con el identificador del servidor de autorización, y `url`, con la URL del Servidor de Autorización.

- `ResourceServer` - contiene los `Scopes` soportados por cada Servidor de Recursos, junto con un `Key`, que servirá como clave para generar el valor cifrado del `id` del Servidor de Autorización

policies.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<AssertionList>
     <Assertion type="papi">
          <Policies scope="scope_de_prueba">
               <TokenFormat>
                    <format>%sHO%</format>
                    <format>%mail%</format>
                    <format>%uid%</format>
               </TokenFormat>
               <Policy>
                    <Attributes check="all">
                         <Attribute name="mail" value=""/>
                         <Attribute name="uid" value=""/>
                    </Attributes>
                    <Attributes check="any">
                         <Attribute name="sHO" value=""/>
                         <Attribute name="sHO" value=""/>
                    </Attributes>
                    <Attributes check="none">
                         <Attribute name="mail" value=""/>
                         <Attribute name="uid" value=""/>
                    </Attributes>
               </Policy>
          </Policies>
          <Policies scope="scope_de_prueba2">
               <TokenFormat>
                    <format>%sHO%</format>
               </TokenFormat>
               <Policy>
                    <Attributes check="any">
                         <Attribute name="sHO" value=""/>
                    </Attributes>
                    <Attributes check="all">
                         <Attribute name="mail" value=""/>
                    </Attributes>
               </Policy>
          </Policies>
     </Assertion>
```

```
39.      <Assertion type="saml2">
40.           <Policies scope="scope_de_prueba_SAML">
41.                <TokenFormat>
42.                     <format></format>
43.                     <format></format>
44.                </TokenFormat>
45.                <Policy>
46.                     <Attributes check="any" >
47.                          <Attribute name=" " value=""/>
48.                     </Attributes>
49.                     <Attributes check="all" >
50.                          <Attribute name="" value=""/>
51.                     </Attributes>
52.                     <Attributes check="none" >
53.                          <Attribute name="" value=""/>
54.                     </Attributes>
55.                </Policy>
56.           </Policies>
57.      </Assertion>
58. </AssertionList>
```

Contiene las políticas, según el scope, que se deben cumplir para que el Servidor de Autorización pueda emitir un token válido. Esta dividida por tipo de aserción(papi o saml2), debiendo de crearse dos políticas distintas para un mismo scope, una por cada tipo de aserción:

- `TokenFormat` - son atributos, extraidos de la aserción, necesarios para obtener el recurso asociado al scope. Estos atributos serán parte del token.

- `Policy` - es la política a seguir para el cada scope. Los atributos `check` pueden ser:

  - **any** : al menos debe de venir en la aserción un atributo `name` cuyo valor sea `value`.

  - **all** : todos los atributos que vengan en la aserción con nombre `name` deben de tener el valor `value`.

  - **none** : no puede venir en la aserción ningún atributo con nombre `name` cuyo valor sea `value`.

SERVIDOR DE RECURSOS(RS):

asKeys.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Registered Auth Servers-->
<AuthServers>
<AuthServer id="authserver_example" url="url token endpoint AS">
<Key>oauth_server_key</Key>
</AuthServer>
</AuthServers>
```

Contiene la información sobre los Servidores de Autorización registrados, para los cuales el Servidor de Recursos puede decodificar los tokens de acceso que le llegan de un cliente, pudiendo verificar si son válidos o no:

- `AuthServer` – contiene el identificador `id` del Servidor de Autorización y la `url` de su token endpoint. Además tiene una `key` que servirá para verificar que el token de acceso ha sido emitido por un Servidor de Autorización registrado.

resourceClasses.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Resources>
     <Scope id="scope_de_prueba">
          <TokenFormat>
               <format>%sHO%</format>
               <format>%mail%</format>
               <format>%uid%</format>
          </TokenFormat>
          <ResourceClass>Resource</ResourceClass>
          <ResourceFile>Resource.php</ResourceFile>
     </Scope>

     <Scope id="scope_de_prueba_SAML">
          <TokenFormat>
               <format></format>
               <format>format>
          </TokenFormat>
          <ResourceClass>Resource</ResourceClass>
          <ResourceFile>Resource.php</ResourceFile>
     </Scope>
</Resources>
```

Contiene los atributos, que deben venir en el token, para poder acceder al recurso definido por cada scope, además de la clase que se usará para dar formato a ese recurso.