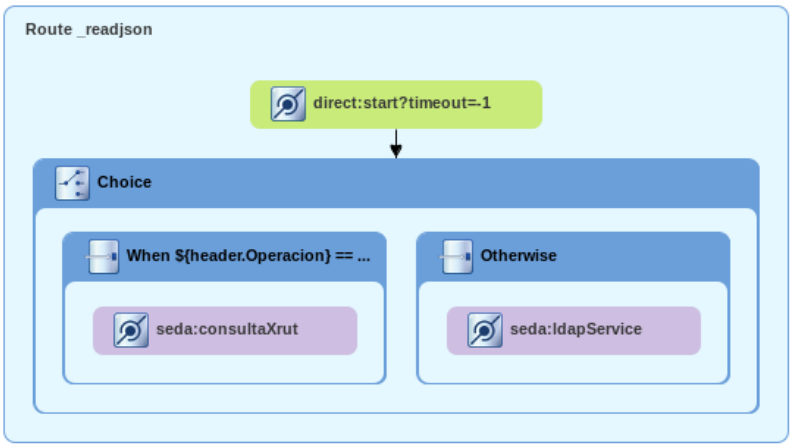

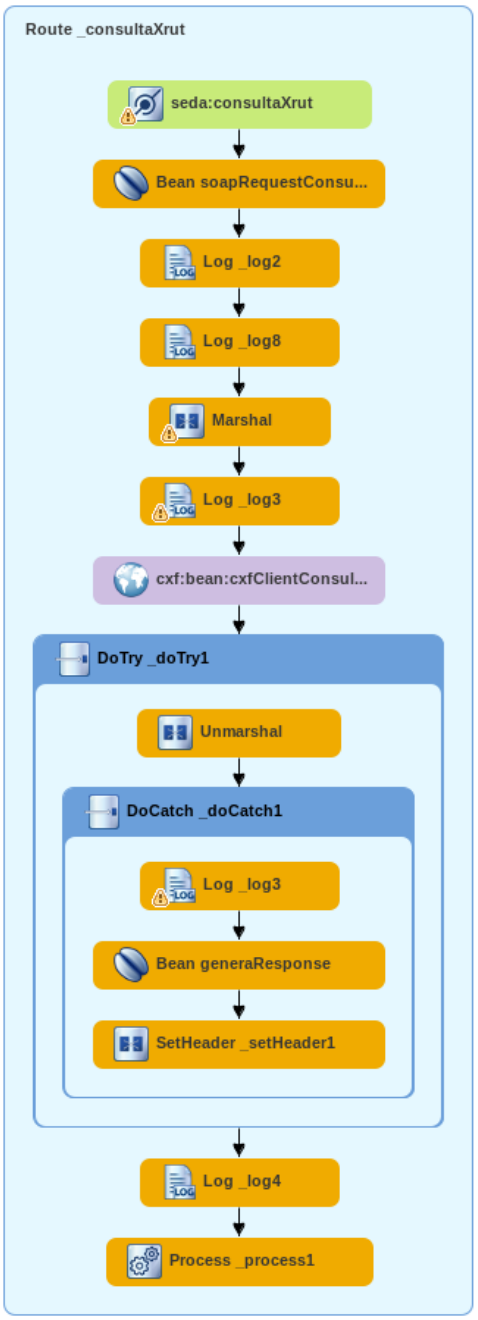

Servicios AD (Active Directory)

DOCUMENTACIÓN

uandes.servicios | servicios-ad-soap | 1.0.1

Descripción del servicio

Este servicio permite invocar a los servicios SOAP que acceden al Active Directory desde un API REST desplegado en el FUSE.

Hay 2 servicios SOAP usados:

- Consulta por RUT:
  https://webapp.uandes.cl:443/LDAP/src/wsLdapUandespanel.php
- Servicios de Active Directory (desarrollo FIR):
  http://10.1.1.49/LDAPServiceb/LDAPService.svc

El servicio ofrece 7 funciones a las cuales se accede por las siguientes URL:

base - https://panel.uandes.cl/cxf/ESB/panel/serviciosAD ....

1. / consultaXrut
2. / activarDesactivarUsuario
3. / actualizarUsuario
4. / crearUsuario
5. / desbloquearUsuario
6. / resetearPassword
7. / validarUsuario

REQUEST

Se usan 2 json:

1. ConsultaXrutRequest – para Consulta por RUT
2. ServiciosLDAPRequest – para lo servicios de Active Directory

Ejemplos:

Para consultar una cuenta por rut se usa este request:

```
{
  "rut": "[redacted]"
}
```

Para usar los servicios LDAP:

- Valida cuenta:

```
{
	"servicio": "ValidarUsuario",
	"datos-servicio": {
		"cuenta": "[redacted]"
	}
}
```

RESPONSE

El servicio responde con los siguientes responses:

1. ConsultaXrutResponse – servicio Consulta por RUT
2. ServiciosLDAPResponse – servicios Active Directory

Ejemplos:

Consulta por RUT

Si la cuenta no existe, se recibe esta respuesta:

```
{
    "cantidad": 0,
    "rut": "[redacted]",
    "usuario": "",
    "employeeid": "",
    "dn": "",
    "ou": "",
    "correo": "",
    "activa": "",
    "bloqueada": "",
    "estado": "ERROR: cuenta no existe",
    "codigo": 0,
    "mensaje": "OK"
}
```

Si la cuenta existe:


```
{
   "cantidad": 1,
   "rut": "[redacted]",
   "usuario": "[redacted]",
   "employeeid": "[redacted]",
   "dn": "CN=FERNANDO GERMAN IBARRA RODRIGUEZ,OU=BibliotecaUA,OU=Global
Users,DC=ua,DC=cl",
   "ou": "",
   "correo": "fgibarra@miuandes.cl",
   "activa": "SI",
   "bloqueada": "NO",
   "estado": "OK",
   "codigo": 0,
   "mensaje": "OK"
```


Servicios LDAP:

- Valida cuenta:


```
{
   "codigo": 0,
   "mensaje": "OK"
}
```


OBSERVACIONES DE IMPLEMENTACION

Como usar invocaciones SOAP desde Camel

- *En el pom*, del proyecto se incluye el plugin cxf-codegen-plugin de org.apache.cxf para generar el código a partir de las wsdl de los servicios SOAP. Uno para la wsdl wsLdapUandes.php.wsdl, el otro para LDAPServiceb.wsdl.

  Ambas wsdl se ubican en el path: src/main/resources/wsdl

- *En el blueprint.xlm* del proyecto se deben incluir los siguientas tags:

```
    <http:conduit name="*.http-conduit">
        <http:tlsClientParameters disableCNCheck="true">
            <sec:keyManagers keyPassword="Uandes2023">
                <sec:keyStore
                    file="/usr/local/fuse/etc/certs/server.keystore"
                    password="Uandes2023" type="JKS"/>
            </sec:keyManagers>
            <sec:trustManagers>
                <sec:keyStore
                    file="/usr/local/fuse/etc/certs/server.keystore"
                    password="Uandes2023" type="JKS"/>
            </sec:trustManagers>
            <sec:cipherSuitesFilter>
                <!-- these filters ensure that a ciphersuite with
                     export-suitable or null encryption is used, but exclude
                      anonymous Diffie-Hellman key change as this is
                      vulnerable to man-in-the-middle attacks
                -->
                <sec:include>.*_EXPORT_.*</sec:include>
                <sec:include>.*_EXPORT1024_.*</sec:include>
                <sec:include>.*_WITH_DES_.*</sec:include>
                <sec:include>.*_WITH_AES_.*</sec:include>
                <sec:include>.*_WITH_NULL_.*</sec:include>
                <sec:exclude>.*_DH_anon_.*</sec:exclude>
            </sec:cipherSuitesFilter>
        </http:tlsClientParameters>
        <http:client AutoRedirect="true" Connection="Keep-Alive"
            ConnectionTimeout="60000" ReceiveTimeout="60000"/>
    </http:conduit>
```

```
<cxf:cxfEndpoint address="{{consultaXrut.endpointAddress}}"
        id="cxfClientConsultaXrutEndpoint"
        serviceClass="wsldapuandes.WS_0020Consulta_0020LDAPPortType"
wsdlURL="{{consultaXrut.wsdlAddress}}">
<!-- The interceptors - needed to log the SOAP requests and responses -->
<!-- They can be removed, when no logging is needed -->
        <cxf:inInterceptors>
            <ref component-id="loggingInInterceptor"/>
        </cxf:inInterceptors>
        <cxf:outInterceptors>
            <ref component-id="loggingOutInterceptor"/>
            <ref component-id="headerOutInterceptorConsultaRutAD"/>
        </cxf:outInterceptors>
        <cxf:outFaultInterceptors>
            <ref component-id="loggingOutInterceptor"/>
        </cxf:outFaultInterceptors>
        <cxf:inFaultInterceptors>
            <ref component-id="loggingInInterceptor"/>
        </cxf:inFaultInterceptors>
</cxf:cxfEndpoint>
```

```
<camel:dataFormats>
  <camel:soapjaxb contextPath=
                      "cl.uandes.comun.ad.client.soap.cxf.consultaXrut.dto"
                  id="soapConsultaXrut"/>
  <camel:soapjaxb contextPath="org.tempuri" id="soapLdapService"/>
</camel:dataFormats>
```

```
    <!-- BEANS para manejo del cliente SOAP -->
    <!-- The interceptors bean definitions - used for logging SOAP requests. -->
    <!-- They can be removed, when no logging is needed -->
    <bean class="org.apache.cxf.interceptor.LoggingInInterceptor"
          id="loggingInInterceptor">
        <property name="prettyLogging" value="true"/>
    </bean>
    <bean class="org.apache.cxf.interceptor.LoggingOutInterceptor"
          id="loggingOutInterceptor">
        <property name="prettyLogging" value="true"/>
    </bean>
    <bean class="cl.uandes.comun.ad.client.soap.cxf.consultaXrut.CxfHeaderFilterStrategy"
          id="cxfConsultaXrutHeaderFilterStrategy"/>
    <bean class="cl.uandes.comun.ad.client.soap.cxf.consultaXrut.CxfHeaderOutInterceptor"
          id="headerOutInterceptorConsultaRutAD"/>
    <bean class="cl.uandes.comun.ad.client.soap.cxf.consultaXrut.CreateSoapRequest"
          id="soapRequestConsultaXrut">
        <property name="operationName" value="{{consultaXrut.operationName}}"/>
        <property name="soapAction" value="{{consultaXrut.soapAction}}"/>
    </bean>
    <bean class="cl.uandes.comun.ad.client.soap.cxf.ldapService.CxfHeaderFilterStrategy"
          id="cxfLdapServiceHeaderFilterStrategy"/>
    <bean class="cl.uandes.comun.ad.client.soap.cxf.ldapService.CxfHeaderOutInterceptor"
          id="headerOutInterceptorLdapService"/>
    <bean class="cl.uandes.comun.ad.client.soap.cxf.ldapService.CreateSoapRequest"
          id="soapRequestPojoLdapService">
        <property name="activarDesactivarUsuario"
                  value="{ {operacion.activarDesactivarUsuario}}"/>
        <property name="actualizarUsuario" value="{{operacion.actualizarUsuario}}"/>
        <property name="crearUsuario" value="{{operacion.crearUsuario}}"/>
        <property name="desbloquearUsuario" value="{{operacion.desbloquearUsuario}}"/>
        <property name="resetearPassword" value="{{operacion.resetearPassword}}"/>
        <property name="validarUsuario" value="{{operacion.validarUsuario}}"/>
    </bean>
    <!-- fin BEANS para manejo del cliente SOAP -->
```

```
<route id="_consultaXrut">
  <from id="_ruta1" uri="seda:consultaXrut"/>
  <bean id="_bean1" method="createSoapBody" ref="soapRequestConsultaXrut"/>
  <marshal id="_marshal1" ref="soapConsultaXrut"/>
  <to id="_to1"
       uri="cxf:bean:cxfClientConsultaXrutEndpoint?
       dataFormat=MESSAGE&amp;headerFilterStrategy=#cxfConsultaXrutHeaderFilterStrategy"/>
  <doTry id="_doTry1">
     <unmarshal id="_unmarshal1">
        <custom ref="soapConsultaXrut"/>
     </unmarshal>
     <camel:doCatch id="_doCatch1">
         <exception>javax.xml.bind.UnmarshalException</exception>
         <log id="_log3" message="atrapo un javax.xml.bind.UnmarshalException"/>
         <bean id="_bean3" method="unmarshalError" ref="generaResponse"/>
         <camel:setHeader headerName="Operacion" id="_etHeader1">
             <constant>Error</constant>
         </camel:setHeader>
     </camel:doCatch>
  </doTry>
  <process id="_process1" ref="generaResponse"/>
</route>
```

NOTAS:

1. El el tag `http:tlsClientParameters` del `http:conduit` se define la ubicación del keystore para poder usar https.

2. El tag `http:client` del `http:conduit` se define el timeout para la invocación del servicio.
3. En el atributo `serviceClass` del tag `cxf:cxfEndpoint` se define la Interface generada por el plugin org.apache.cxf. cxf-codegen-plugin. 3.5.1 donde se describe los Entrypoints del web service SOAP.
4. En el atributo `wsdlURL serviceClass` del tag `cxf:cxfEndpoint` se define la ubicación de la wsdl del servicio. Puede indicarse como una key del archivo de propiedades del contexto Camel.
5. El atributo `id` del tag `cxf:cxfEndpoint` es utilizado por el tag `<to uri="cxf:bean:cxfClientConsultaXrutEndpoint?dataFormat=MESSAGE&amp;headerFilterStrategy=#cxfConsultaXrutHeaderFilterStrategy"/>` dentro de un Process o ruta.
6. Se pueden definir Interceptors durante el proceso del message, pero en la practica sólo lo usé para Logging.
7. La función `<camel:dataFormats>` que se incluye dentro del blueprint después del `<propertyPlaceholder`, permite convertir XML a Objeto y viceversa. Se usa para colocar / sacar del Body el request y el response del webservice SOAP invocado. Dentro de este tag, se define `camel:soapjaxb,` atributo `contextPath` define el package generado por el plugin donde se encuentran las clases para instanciar el Request y Response; incluye también la clase ObjectFactory para instanciarlos.
8. En el tag <route> se incluye los tags <marshall> y <unmarshall> que hacen referencia al `camel:soapjaxb` incluido dentro del tag `<camel:dataFormats>`. Estas funciones serializan / de serializan los objetos incluidos en el Body del Message.

NOTA caso Consulta por RUT

La wsLdapUandes.php.wsdl está modificada a partir de la que entrega el servicio SOAP porque el plugin no soporta wsdl del tipo “rpc encoded”. Se convirtió al tipo literal. Sin embargo esta modificación no quedó del todo completa por lo que las clases que describen el Request y el Response no son del todo compatible con el servicio, por lo que hubo que generar un package cl.uandes.comun.ad.client.soap.cxf.consultaXrut.dto para alojar una adaptación de las clases que genera el plugin y usar estas clases en la creacion del SOAP Request. Este se crea en la clase CreateSoapRequest del package cl.uandes.comun.ad.client.soap.cxf.consultaXrut. En el atributo `contextPath` del tag `camel:soapjaxb,` se coloca el package generado e indicado previamente.