Modulo conexión Cliente WS – DGI

El desarrollo fue echo con eclipse (eclipse-jee-kepler) utilizando herramientas Apache. De la misma forma puede hacerse con otro lenguaje que acepte estas librerias porque la conexión es más bien un tema de configuración.

Librerías para conexión:

- Axis2 (axis2-1.6.2)
- Rampart (rampart-1.6.2)

Librerías para seguimiento (Anexo):

- stunnel-5.03-installer (intermediario para visualizar mensajes Soap)
- tcp/ip monitor (intermediario para visualizar mensajes Soap)

Los pasos seguidos son:

- Armo un nuevo Cliente de Web Service

🔘 Web Service Client	
Web Services Select a service definition and move the slider to set the	e level of client generation.
Service definition: https://efactura.dgi.gub.uy:6443/	ePrueba/ws_eprueba?w: 🔽 Browse)
Client type: Java Proxy	✓
Develop client	Configuration: <u>Server runtime: Tomcat v7.0 Server</u> <u>Web service runtime: Apache Axis2</u> <u>Client project: Modulo_conexion</u>
Monitor the Web service	
✓ Overwrite files without warning	
Reck	ext > Finish Cancel

- Le doy siguiente y no hago nada.....

💭 Web Service Clie	nt					
Axis2 Client Web Se	ervice Configuration					
Select the appropriate c	appropriate code generation settings					
			_			
			_			
Service Name	WS_eFactura		*			
Port Name	WS_eFacturaSoapPort					
Databinding	ADB					
Custom package name	uy.gub.dgi					
Client mode						
Generate a client w	hich supports both synchronous	s and asynchronous invocation				
Generate a synchro	nous dient					
Generate all types fo	or all elements referred to by sc	hemas	7			
http://www.w2.org/oc	(we-policy	ora w2 www.pc.wc.policy	-			
http://www.wo.org/ns	n.ora/wsdl/	org.xmlsoan.schemas.wsdl	-			
http://schemas.xmlsoa	p.org/wsdl/http/	org.xmlsoap.schemas.wsdl.http	-			
http://dgi.gub.uy		uy,gub,dgi				
http://www.w3.org/20	01/XMLSchema	org.w3.www2001.xmlschema				
http://schemas.xmlsoa	p.org/ws/2004/09/policy	org.xmlsoap.schemas.ws20040				
http://schemas.xmlsoa	p.org/wsdl/soap/	org.xmlsoap.schemas.wsdl.soap	_			
http://schemas.xmlsoa	org.xmlsoap.schemas.wsdl.mime	_				
http://www.w3.org/20	06/07/ws-policy org.w3.www200607.ws_policy					
_nttp://schemas.xmlsoa	p.org/wsdl/soap12/	org.xmisoap.schemas.wsdi.soap12	-			
0		Rack Nevt S Finish Cance	_			

- Finalizo y automaticamente quedan armadas estas carpetas y archivos:



Luego de esto debo configurar el Rampart:

Dentro de este directorio (que se ve en la figura de arriba) **WebContent/WEB-INF/lib** pego los archivos de librería que se encuentran en **Rampart-1.6.2/lib** (que están dentro de lo que baje de Rampart)

Y dentro del directorio **WebContent/WEB-INF/modules** pego los modulos que están dentro de la carpeta **rampart-1.6.2/modules** (que son rahas-1.6.2 y rampart-1.6.2). Luego de esto agregarlos al listado Modules.list que esta en **WebContent/WEB-INF/modules**

Por algún error del Eclipse dos archivos de la librería de Axis2 no los trae y los tengo que incluir en WebContent/WEB-INF/lib pegandolos desde lo que baje de Axis2 dentro de la carpeta axis2-1.6.2/lib , los archivos son jaxen-1.1.1.jar y mex-1.6.2-impl.jar

Más adelante, en el Anexo muestro como configurar el visor de los mensajes SOAP, que lo armo Maurizio, y que es vital para este desarrollo, ya que sino trabajamos a ciegas

Una vez se tiene todo cargado se cambia el archivo de configuración **Axis2.xml** que viene por defecto por el que adjunto con el pdf (se ve en figura de arriba)

Luego de pegado el archivo Axis2.xml hay que abrirlo y setear con sus datos temas de seguridad

De aca en adelante pongo entre [] (no van los parentesis) lo que hay que completar:

sec.properties - Crypto properties para enviar (va en carpeta WebContent/WEB-INF/classes)
recibe.properties - Crypto properties para recibir (va en carpeta WebContent/WEB-INF/classes)
uy.conexion.Psw cb - Clase que viene más abajo

Al Proyecto le agrego los archivos en el paquete creado "uy.conexion" y las properties que menciono arriba :



- Conexion.java es la que realizara la conexión, paso codigo en detalle a continuación
- Convertidor.java convierte para pasar sobre de ejemplo, paso codigo en detalle a continuación
- Psw_cb.java es la que maneja la contraseña, paso código también
- politica.xml es la politica que le paso al Rampart para que funcione, De aqui toma la configuración para hablar con el certificado y el Security Binding (es lo que esta entre etiquetas AsymmetricBinding maneja los token)
- prueba.xml es el sobre de ejemplo para pasar a DGI con los comprobantes

- La clase WS_eFacturaStub.java que esta en el paquete uy.gub.dgi que aparece arriba (Maneja toda la conexión y es armada automaticamente con la definicion del servicio que hicimos en el primer paso)
- sec.properties archivo que tienen que llenar con sus datos (Hay que ubicarlo en WEB-INF/classes como se ve arriba). El contenido es:

```
org.apache.ws.security.crypto.provider=
org.apache.ws.security.components.crypto.Merlin
org.apache.ws.security.crypto.merlin.keystore.type=JKS
org.apache.ws.security.crypto.merlin.keystore.password=[contraseña]
org.apache.ws.security.crypto.merlin.keystore.alias=[alias certif nuestro]
org.apache.ws.security.crypto.merlin.keystore.file=C:\\..\\archivo.jks
```

- recibe.properties – archivo que tienen que llenar con sus datos (Hay que ubicarlo en WEB-INF/classes como se ve arriba). El contenido es: org.apache.ws.security.crypto.provider= org.apache.ws.security.components.crypto.Merlin org.apache.ws.security.crypto.merlin.keystore.type=JKS org.apache.ws.security.crypto.merlin.keystore.password=[contraseña] org.apache.ws.security.crypto.merlin.keystore.alias=[alias certif DGI] org.apache.ws.security.crypto.merlin.keystore.file=C:\\..\\archivo.jks

La parte de código:

Conexion.java

package uy.conexion;

```
import java.io.File;
import java.io.IOException;
import java.rmi.RemoteException;
import org.apache.axiom.om.impl.builder.StAXOMBuilder;
import org.apache.axis2.client.Options;
import org.apache.axis2.context.ConfigurationContext;
import org.apache.axis2.context.ConfigurationContextFactory;
import org.apache.neethi.Policy;
import org.apache.neethi.PolicyEngine;
import org.apache.rampart.RampartMessageData;
```

import uy.gub.dgi.WS eFacturaStub;

```
public class Conexion {
```

```
public static void main(String[] args) throws
javax.xml.stream.XMLStreamException, IOException {
```

// Configuramos el contexto
final ConfigurationContext ctx = ConfigurationContextFactory.
 createConfigurationContextFromFileSystem(
 "WebContent/WEB-INF", // Donde están Axis2 y Rampart
 "WebContent/WEB-INF/conf/axis2.xml"); // Ubicacion exacta del
 archivo axis2.xml que maneja la conexión

final WS_eFacturaStub wse_conexion = new WS_eFacturaStub(ctx);
//Genero la conexion con la clase de la DGI y el contexto de arriba

```
// Aca armo para cargar la politica falsa para que funcione Rampart
    final Options options = wse conexion. getServiceClient().getOptions();
    // Cargamos el archivo de la politica Rampart
    final StAXOMBuilder builder =
      new StAXOMBuilder("src/uy/conexion" + File.separator
        + "politica.xml");
    final Policy policy =
     PolicyEngine.getPolicy(builder.getDocumentElement());
     options.setProperty(RampartMessageData.KEY RAMPART POLICY,
     policy);
    // Aca armo el envío seteando la clase WS eFacturaStub
    uy.gub.dgi.WS eFacturaStub.WS eFacturaEFACRECEPCIONSOBRE sobre = new
    uy.gub.dgi.WS eFacturaStub.WS eFacturaEFACRECEPCIONSOBRE();
    WS eFacturaStub.Data xml sobre= new WS eFacturaStub.Data();
    // Aca convierto para adjuntar el xml
     Convertidor convertidor = new Convertidor();
xml sobre.setXmlData(convertidor.convertToXMLData(
"src/uy/conexion/prueba.xml"));
     sobre.setDatain(xml sobre);
    // Aca invoco al Web Service
     try {
uy.gub.dgi.WS eFacturaStub.WS eFacturaEFACRECEPCIONSOBREResponse
respSobre= wse conexion.eFACRECEPCIONSOBRE(sobre);
           System.out.println(respSobre.getDataout().getXmlData());
      } catch (RemoteException e) {
           // TODO Auto-generated catch block
           e.printStackTrace();
      }
  }
```

Convertidor.java

}

```
package uy.conexion;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
public class Convertidor {
    public Convertidor() { }
    public String convertToXMLData(String fileName) throws
        FileNotFoundException, IOException {
        BufferedReader br;
        br = new BufferedReader(new FileReader(fileName));
        int c;
        StringBuilder response= new StringBuilder();
        while ((c = br.read()) != -1) {
            response.append( (char)c ) ;
        }
    };
    };
}
```

```
}
String xmlData = response.toString();
br.close();
return xmlData;
}
```

Psw cb.java

```
package uy.conexion;
import java.io.IOException;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;
import org.apache.ws.security.WSPasswordCallback;
public class Psw cb implements CallbackHandler {
public void handle(Callback[] callbacks) throws IOException,
UnsupportedCallbackException {
            // Con esta Clase manejo el acceso al certificado (Hay que mejorar
            el tema seguridad con Digest, etc)
        WSPasswordCallback pwcb = (WSPasswordCallback)callbacks[0];
        String id = pwcb.getIdentifier();
        int usage = pwcb.getUsage();
        if (usage == WSPasswordCallback.SIGNATURE || usage ==
           WSPasswordCallback.DECRYPT ) {
            if ("[Usuario del Certificado]".equals(id)) {
                 pwcb.setPassword("[Contraseña del Certificado]");
            }
        }
    }
}
```

Con estas 3 clases y los archivos mencionados queda armada la conexión y el web service acepta el sobre eviado....

```
<ACKSobre xmlns="http://cfe.dgi.gub.uy" version="1.0">
...
<CantidadCFE>50</CantidadCFE>
<Tmst>2014-09-24T11:36:21-03:00</Tmst></Caratula>
<Detalle><Estado>AS</Estado>
...
```

<u>ANEXO</u>

Seguimiento mensajes SOAP

Lo último para configurar es la visualización de los mensajes Soap. Sin esta herramienta estaríamos trabajando a ciegas.

Basicamente con el TCP/IP Monitor nos conectamos al Stunnel que se conecta a la DGI con lo cual utilizando este elemento en el medio podemos ver los mensajes.

TCP Monitor se conecta al puerto 8081 y monitorea el puerto 8080 Stunnel recibe del 8081 y se conecta a efactura.dgi.gub.uy:6443

Las configuraciones necesarias son:

En el stunnel config

client=yes verify=0 debug=7

[my-https] accept = 8081 connect = efactura.dgi.gub.uy:6443 TIMEOUTclose = 0

En el TCP/IP Monitor

Preferences							
type filter text		TCP/IP Ma	nitor			<	↓ • ⇒ • •
 General Ant Data Management GlassFish Preferences Help 		Configure TC	P/IP monitors on loca TCP/IP Monitor view v prs:	l and remote p vhen there is a	oorts. activity		
⊡ Install/Update		Status	Host name	Туре	Local P	Auto-start	Add
⊕ Java ⊕ Java EE		🖳 Started	127.0.0.1:8081	TCP/IP	8080	Yes	Edit
■ JavaScript ■ Maven							Remove
. • • • • • • • • • • • • • • • • • • •	=						Start
🖅 Plug-in Development							Shop
Remote Systems							
···· Console ···· External Tools							Columns
Launching Perspectives							
TCP/IP Monitor							
•••• Server							
Terminal Validation	~						

Dentro de la clase de la DGI WS_eFacturaStub.java cambiamos momentaneamente lo siguiente para que el monitoreo funcione:



}

Con esto queda listo y arroja lo siguiente:

🕴 Servers 📮 Console 🔫 Progress 💂 TCP/IP Monitor 🙁 🕸 Debug	JO 🖉 🛃 🎽 🎽 🖛 e
□ ☐ localhost:8080 □	Time of request: 10:43.6.156 PM Response Time: 969 ms Type: TCP/IP
Request viewer type: Byte	Response viewer type: Byte
Request: localhost:8080 Size: 8014 (8014) bytes	Response: 127.0.0.1:8081 Size: 8790 (8790) bytes
<pre>POST /ePrueba/ws_eprueba HTTP/1.1 Content-Type: text/xml; charset=UTF-8 SOAPAction: "http://dgi.gub.uyaction/AWS_EFACTURA.EFACR User-Agent: Axis2 Host: localhost:8080 </pre>	HTTP/1.1 200 OK X-Backside-Transport: OK OK Connection: Keep-Alive Transfer-Encoding: chunked Date: Mon, 01 Sep 2014 22:43:34 GMT
Encoding: <pre></pre>	Encoding: None>
POST /ePrueba/ws_eprueba HTTP/1.1 Content-Type: text/xml; charset=UTF-8 SOAPAction: "http://dgi.gub.uyaction/AWS_EFACTURA.EFACR User-Agent: Axis2 Host: localhost:8080 Content-Length: 7804 xml version='1.0' encoding='UTF-8'? <soapenv:envelope QwMjExMjEzNTEOWhcNMTUwMjExMjEzNTEOWjB3MSUwIwYJKo2IhvcNA BgNhLhluzIG3IA+vUKdObgMJ9Ey10i99I03Ej7ebqODbXWStuL3bTZM</soapenv:envelope 	HTTP/1.1 200 OK X-Backside-Transport: OK OK Connection: Keep-Alive Transfer-Encoding: chunked Date: Mon, 01 Sep 2014 22:43:34 GMT Server: IBM_HTTP_Server Pragma: no-cache Expires: Mon, 01 Sep 2014 22:43:34 GMT Last-Modified: Mon, 01 Sep 2014 22:43:34 GMT Cache-Control: max-age=0, no-cache, no-store, must-reva.

Response viewer type: Byte
Response: 127.0.0.1:8081 Size: 8790 (8790) bytes
HTTP/1.1 200 OK
Encoding:
<soap-env:envelope :<="" http:="" td="" www.w3.org="" xmlns:soap-env="http://schemas.xmlsoa</td></tr><tr><td>MTQyWhcNMTUwMjI1MTYyMTQyWjCB9DEiMCAGCSqGSIb3DQEJARYTZ2R</td></tr><tr><td>Q29ycmVvQ2VydC9hbmMuY3JsMAOGCSqGSIb3DQEBBQUAA4ICAQAc9g71</td></tr><tr><td><SignedInfo></td></tr><tr><td><CanonicalizationMethod Algorithm="></soap-env:envelope>
<signaturemethod #body-5c5e0269-f2fc-449b-aa93-161105a;<="" algorithm="http://www.w3.org/2000/09,</td></tr><tr><td><Reference URI=" td=""></signaturemethod>
<transforms></transforms>
<transform 09="" 2000="" :<="" algorithm="http://www.w3.org/2001/10/xm</td></tr><tr><td></Transforms></td></tr><tr><td><DigestMethod Algorithm=" http:="" td="" www.w3.org=""></transform>
<digestvalue>Pc/TsDfdITCCk3S83VcDOfOUxQ8=</digestvalue>
<signaturevalue>ATZAUgO46jAQDzvaN/zi5MEjdsOhL1IWSMI</signaturevalue>
£89