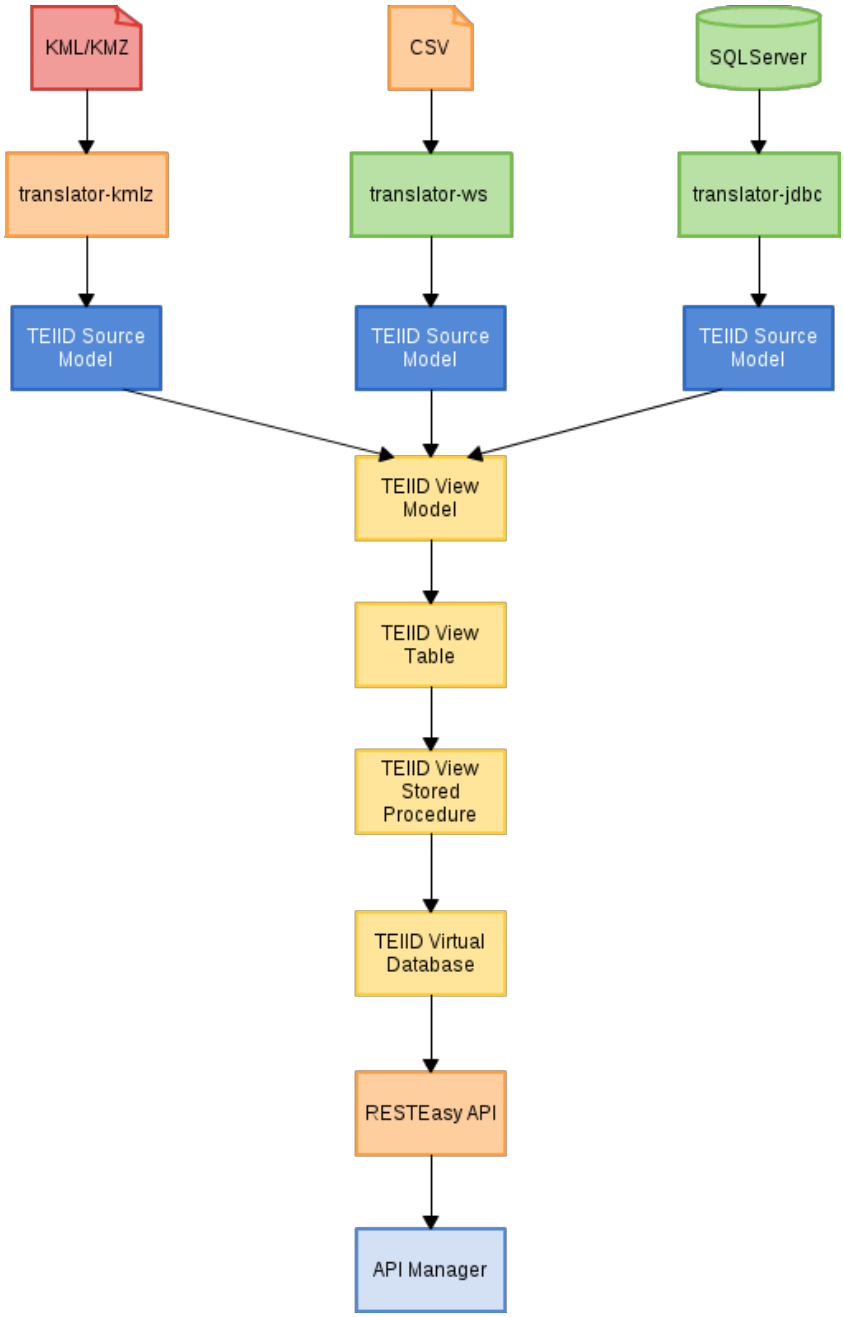
TEIID HOWTO & Backend development

Índice

Backend development workflow



Pasos a seguir

1. Traducción de la fuente de datos.
2. Creación del Source Model en TEIID.
3. Creación del View Model en TEIID.
    a. Crear el View Model.
    b. Crear tablas.
    c. Crear stored procedures.

4. Creación y deployment de la base de datos virtual.
5. Creación de la API REST que llamará a las stored procedures de 3.c.
6. Conexión de la API con API Manager.

Traducción de la fuente de datos

Mientras que para un SQLServer la traducción es automática, para los KMLs hay que usar un traductor custom (translator-kmlz en SVN).

Creación del Source Model en TEIID

SQLServer

Crear el Source Model en TEIID mediante el wizard de import JDBC source. El mismo wizard nos crea también el View Model y una tabla inicial si queremos.

CSVs remotos

Seguir el procedimiento que aparece más adelante en el documento: Ficheros CSV Remotos

KML/KMZs remotos

Seguir el procedimiento que aparece más adelante en el documento: Ficheros KML/KMZ remotos

Creación del View Model en TEIID

SQLServer

Se crea automáticamente en el paso anterior. Añadir las tablas y las stored procedures extra que se necesiten según la API a desarrollar.

CSVs remotos

Seguir el procedimiento que aparece más adelante en el documento: Ficheros CSV Remotos.

KML/KMZs remotos

Seguir el procedimiento que aparece más adelante en el documento: Ficheros KML/KMZ remotos.

Creación y deployment de la base de datos virtual

SQLServer

Simplemente añadir el view model creado en el punto anterior. Crear un datasource válido mediante la consola de JBOSS que apunte al servidor al que queremos conectar. Ver Bases de datos remotas (SQLServer)

CSVs remotos

Seguir el procedimiento que aparece más adelante en el documento: Ficheros CSV Remotos. En este caso modificar el translator a *ws* y utilizar un JNDI válido que apunte al fichero remoto (crear resource adaptor en standalone-teiid.xml).

KML/KMZs remotos

Seguir el procedimiento que aparece más adelante en el documento: Ficheros KML/KMZ remotos. En este caso modificar el translator a *kmlz* y utilizar cualquier JNDI existente, por ejemplo ExampleDS.

Creación de la API REST que llamará a las stored procedures de la VDB

Ampliar proyecto OpenDaiAPIsSpain del SVN (redmine del proyecto).

Conexión de la API con API Manager

Ver documento relacionado en SVN.

[Conexión del BAM con API Manager para sacar estadísticas]

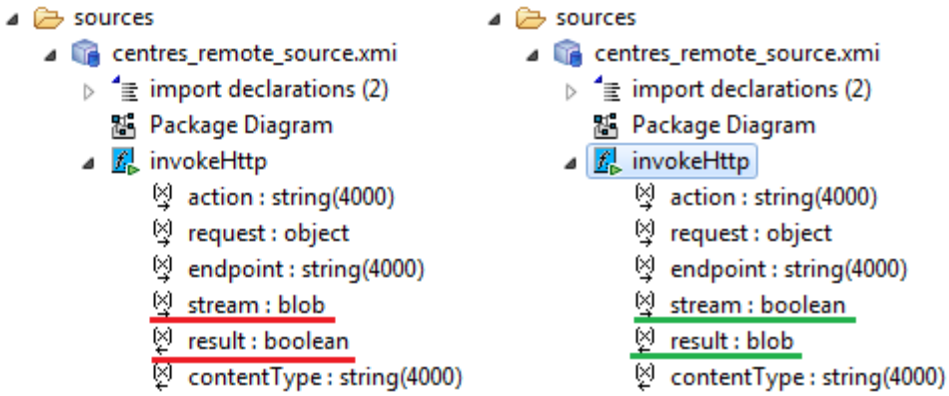
[TBD]

TEIID en la máquina local (devel)

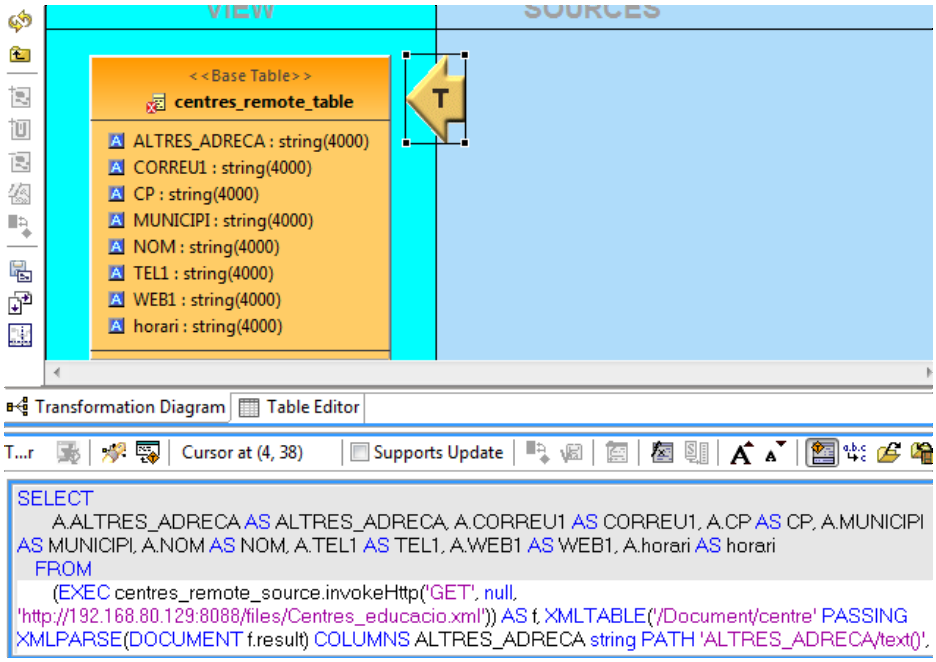
Ficheros XML Remotos

Problemas/inconvenientes que se van encontrando con TEIID 8.2/TEIID Designer 8.0, de momento en versión local.

- ~~No importa flat files remotos cuando se especifica una URI en vez de un fichero local. Parece que espera una URI dentro del directorio deployments del JBOSS. EDIT: Definitivamente no importa ficheros planos de manera remota, sólo XMLs..~~ **Sí que se pueden importar ficheros remotos, ver sección de CSVs.**
- ~~Bug al importar XML remotos en TEIID Designer 8.0 final: Los metadatos para las funciones invoke e invokeHttp utilizados para cargar ficheros remotos no están actualizados y no incluyen el parámetro opcional "stream", por lo que las funciones fallan y no se pueden importar ficheros XML remotos.~~
- El bug no es éste sino el siguiente: Los tipos para los parámetros *stream* y *result* están intercambiados, *stream* tiene que ser booleano y *result* blob, y no al revés como sale por defecto.



- Después modificaremos el diagrama de transformación: Seleccionamos Transformation Diagram en la tabla y doble click en la flecha de transformación (flecha naranja con una T) en el esquema.



- Hay que añadir el parámetro stream que por defecto no sale. Aprovechamos para usar *named parameters* para mayor claridad. La parte en el ejemplo `donde pone:`
  - `EXEC centres_remote_source.invokeHttp('GET', null, 'http://192.168.80.129:8088/files/Centres_educacio.xml')` se transforma en:
  - `EXEC centres_remote_source.invokeHttp(action=>'GET', endpoint=>'http://192.168.80.129:8088/files/Centres_educacio.xml', stream=>TRUE)`
  - A continuación click derecho en la vista -> Validate y la señal de error debería desaparecer, ya se puede crear la VDB. **Acordarse de salvar todos los modelos antes de crear la VDB o nos podemos volver locos al no encontrar las tablas en ésta!!!**

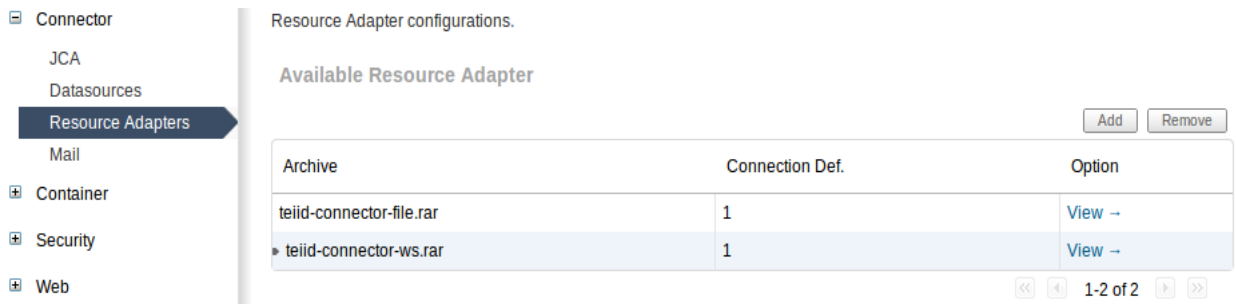
Ficheros CSV Remotos

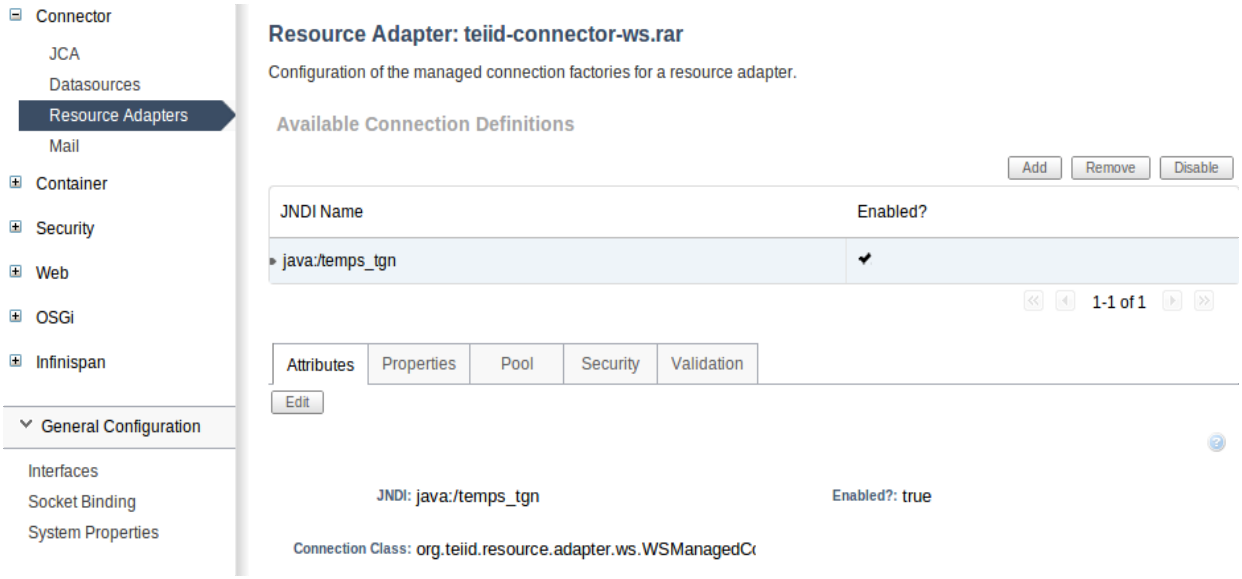
El método es sencillo, aunque no está documentado en ningún sitio. La idea es crear una conexión a un csv local y posteriormente substituir la función que lee del sistema de ficheros (gettextable) por la que lee remotamente (invokeHttp) que se utiliza para los xml. La ventaja de hacerlo de esta manera es que la consulta se genera automáticamente con el wizard.

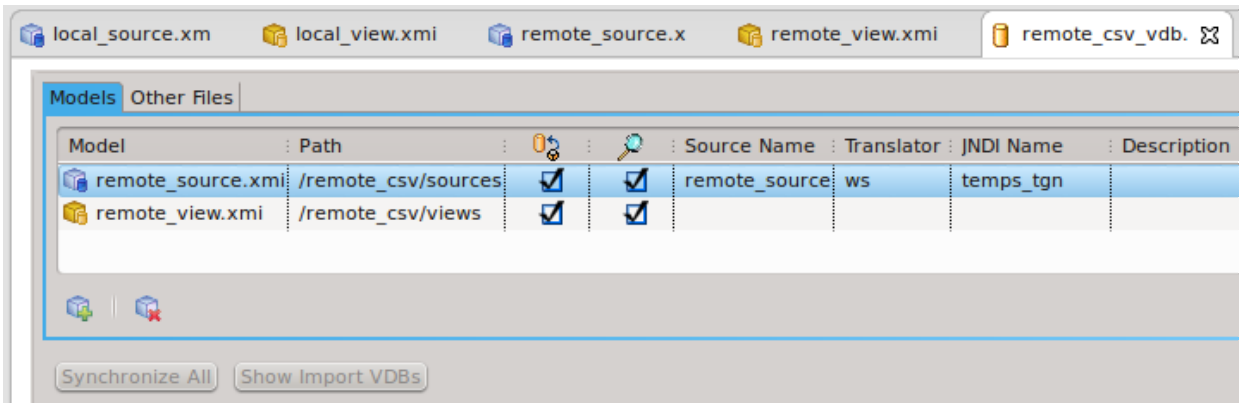
1. Obtener una copia local del csv remoto deseado.
2. Crear un datasource a partir del csv local, el wizard nos creará también la vista y tablas asociadas.
3. Crear un datasource vacío, seleccionar “Generate Web Service Translator Procedures” y elegir sólo invokeHttp.
    a. Inicializar el tamaño de los strings de invokeHttp a 4000.
4. Crear una view vacía y añadir una tabla.
5. Modificar el transformation diagram de la nueva tabla usando el del modelo local pero cambiando la conexión local al fichero por la remota:
    a. Cambiar el gettexttable por el invokeHttp siguiendo el modelo siguiente.
    b. Modificar el origen en TEXTTABLE para usar el result del invokeHttp convertido a clob con TO_CHARS. **Ojo al encoding!! Comprobar encoding de fichero.**

```
SELECT A.Subject, A."Start Date", A."Start Time", A."End Date", A."End Time", A."All
day event", A."Reminder on/off", A."Reminder Date", A."Reminder Time", A.Categories,
A.Description, A.Location, A.Private

FROM (EXEC temps_tgn.invokeHttp(action => 'GET', endpoint =>
'http://dadesobertes.gencat.cat/recursos/calendaris/calendari_laboral_festes_locals_
2012.csv', stream => TRUE)) AS f, TEXTTABLE(TO_CHARS(f.result, 'ISO-8859-1') COLUMNS
Subject string, "Start Date" string, "Start Time" string, "End Date" string, "End
Time" string, "All day event" string, "Reminder on/off" string, "Reminder Date"
string, "Reminder Time" string, Categories string, Description string, Location
string, Private string DELIMITER ';' HEADER) AS A
```

6. Crear la VDB con la nueva vista remota.
    a. En el modelo source en la vista de VDB modificar el Translator, que estará vacío. Poner ‘ws’
    b. Necesitamos un jndi que exista:







Para crear un datasource válido podemos crear un modelo que apunte a un xml remoto. O añadir uno nuevo en el standalone-teiid.xml, por ejemplo:

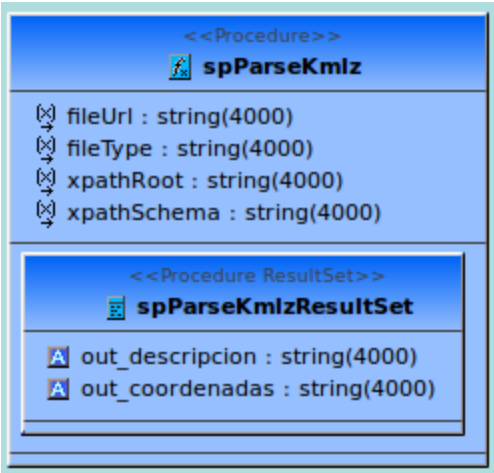
```
<resource-adapter>
    <archive>
       teiid-connector-ws.rar
    </archive>
    <transaction-support>NoTransaction</transaction-support>
    <connection-definitions>
       <connection-definition
class-name="org.teiid.resource.adapter.ws.WSManagedConnectionFactory"
jndi-name="java:/remote_source" enabled="true" pool-name="remote_source">
          <config-property name="EndPoint">

http://dadesobertes.gencat.cat/recursos/calendaris/calendari_laboral_festes_locals_
012.csv
          </config-property>
       </connection-definition>
    </connection-definitions>
</resource-adapter>
```

Se pueden eliminar los modelos locales, sólo los usamos para empezar a automatizar la transformación.

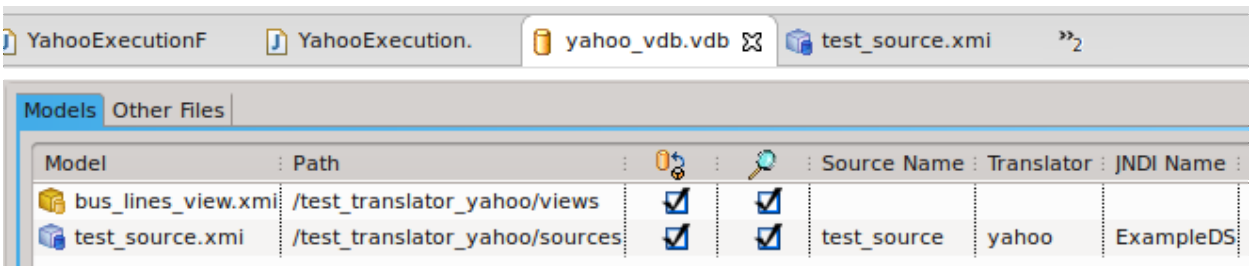
Ficheros KML/KMZ remotos

Utilizaremos un translator de TEIID programado para la ocasión. Se ha partido del translator de yahoo de los fuentes de TEIID. Código en el SVN y más adelante en el documento.

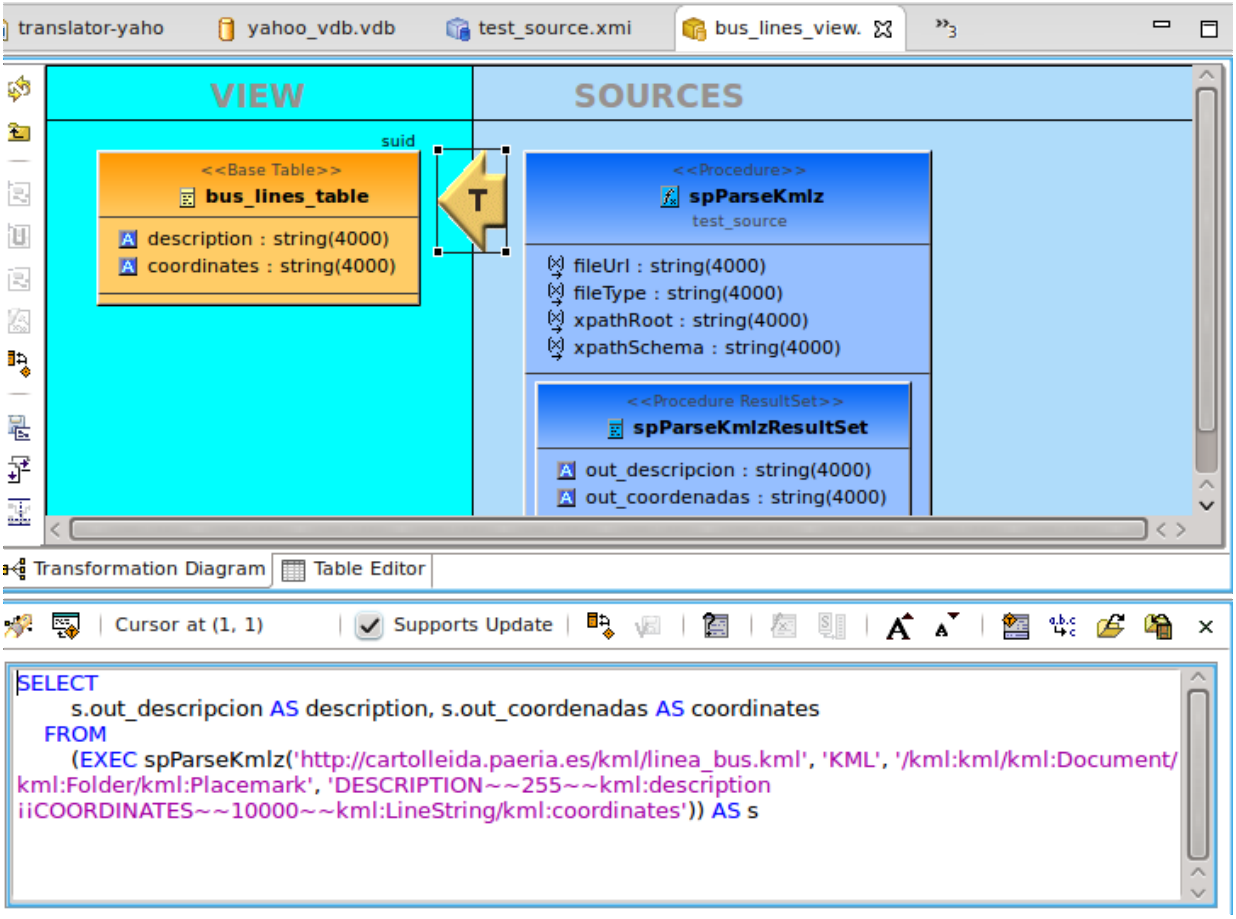
1. Crear un source vacío.
2. Mapear la SP creada en el translator creando una en TEIID como la siguiente (ejemplo):



3. Añadir el source a una VDB y modificar el translator al nuestro. En este caso pone yahoo porque es el que hemos modificado. Queda pendiente actualizar el proyecto y cambiar los nombres.
4. Modificar el JNDI a cualquiera existente, ExampleDS ya nos vale.



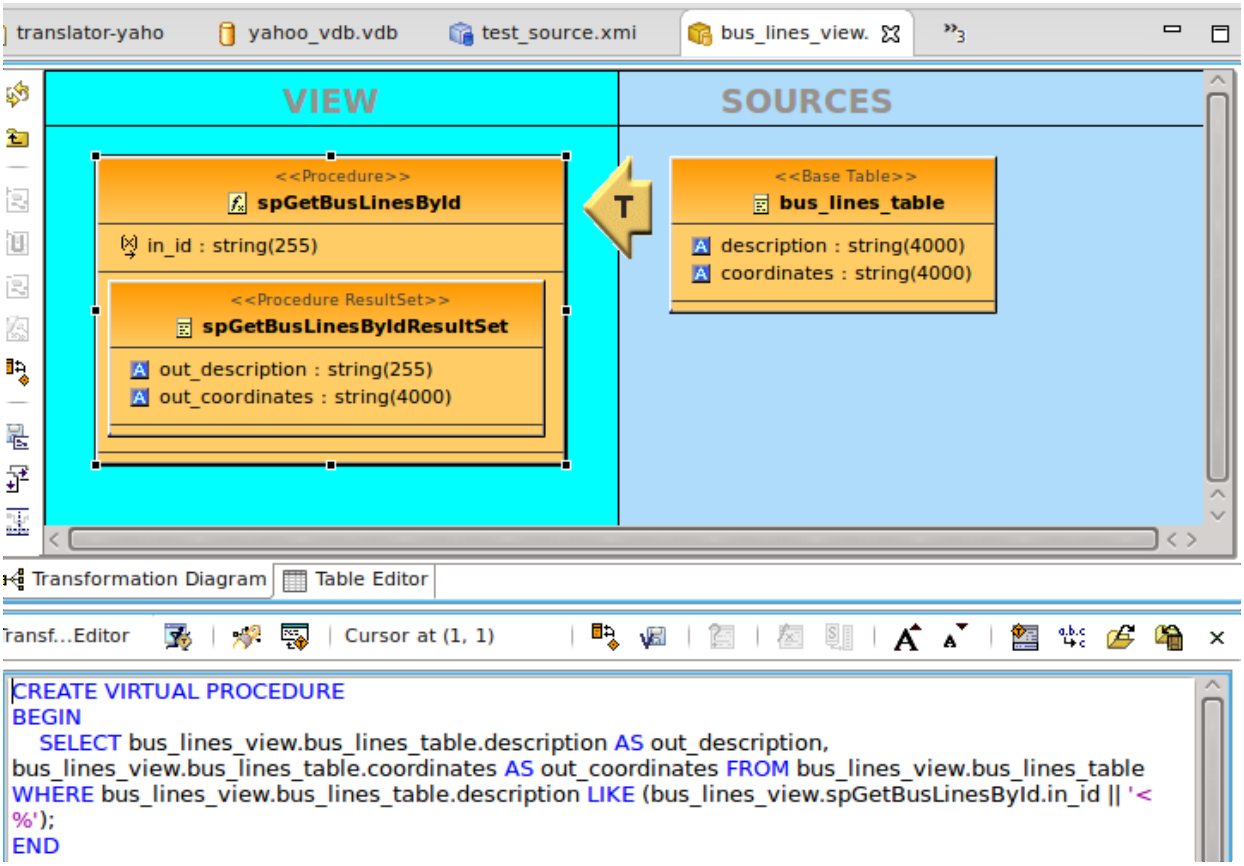
5. Comprobar que todo funciona mediante deploy de la vdb (sólo con el source) y llamando a la SP.
6. Crear una vista vacía.
7. Añadir una tabla a la vista mapeando la SP creada en el source:



8. El código del transformation diagram en este ejemplo es el siguiente:

```
SELECT s.out_descripcion AS description, s.out_coordenadas AS coordinates
FROM (EXEC spParseKmlz('http://cartolleida.paeria.es/kml/linea_bus.kml', 'KML',
'/kml:kml/kml:Document/kml:Folder/kml:Placemark',
'DESCRIPTION~~255~~kml:description¡¡COORDINATES~~10000~~kml:LineString/kml:coordina
es')) AS s
```

9. Crear las SPs sobre la tabla que necesitemos para la API REST. Normalmente un SELECT * y un SELECT parametrizado por ID. Por ejemplo:



Seguramente podríamos saltarnos el paso de crear la tabla en la vista, pero así queda más claro y es más flexible. No creo que afecte en nada al rendimiento.

10. Llamar a las SP desde la API REST normalmente una vez deployed la VDB actualizada.

Código del translator

KmlzExecutionFactory.java

```
package org.teiid.translator.kmlz;

import org.teiid.language.Call;
import org.teiid.metadata.MetadataFactory;
import org.teiid.metadata.Procedure;
import org.teiid.metadata.RuntimeMetadata;
import org.teiid.metadata.ProcedureParameter.Type;
import org.teiid.translator.ExecutionContext;
import org.teiid.translator.ExecutionFactory;
import org.teiid.translator.ProcedureExecution;
import org.teiid.translator.Translator;
import org.teiid.translator.TranslatorException;
import org.teiid.translator.TypeFacility;

@Translator(name="kmlz", description="Translator for remote KML and KMZ files")
public class KmlzExecutionFactory extends ExecutionFactory<Object, Object> {
```

```
        public KmlzExecutionFactory() {
                setSourceRequiredForMetadata(false);
        }

    @Override
    public void start() throws TranslatorException {
    }

    @Override
    public ProcedureExecution createProcedureExecution(Call command, ExecutionContext
executionContext, RuntimeMetadata metadata, Object connectionFactory)
                throws TranslatorException {
//      System.out.println("COMANDO: "+command.toString());
        return new KmlzProcedureExecution(command);
    }


    public boolean supportsCompareCriteriaEquals() {
        return true;
    }

    public boolean supportsInCriteria() {
        return true;
    }

    @Override
    public boolean isSourceRequired() {
        return false;
    }

        @Override
        public void getMetadata(MetadataFactory metadataFactory, Object connection) throws
TranslatorException {
                Procedure spParseKmlz = metadataFactory.addProcedure("spParseKmlz");
                metadataFactory.addProcedureParameter("fileUrl", TypeFacility.RUNTIME_NAMES.STRING,
Type.In, spParseKmlz);
                metadataFactory.addProcedureParameter("fileType", TypeFacility.RUNTIME_NAMES.STRING,
Type.In, spParseKmlz);
                metadataFactory.addProcedureParameter("xpathRoot",
TypeFacility.RUNTIME_NAMES.STRING, Type.In, spParseKmlz);
                metadataFactory.addProcedureParameter("xpathSchema",
TypeFacility.RUNTIME_NAMES.STRING, Type.In, spParseKmlz);
        }

        @Override
        public boolean supportsOnlyLiteralComparison() {
                return true;
        }

}
```

KmlzProcedureExecution.java

```
package org.teiid.translator.kmlz;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import org.teiid.language.Call;
import org.teiid.translator.DataNotAvailableException;
import org.teiid.translator.ProcedureExecution;
import org.teiid.translator.TranslatorException;

//XPath part
//import java.sql.ResultSet;
//import java.sql.Types;
//import org.h2.tools.SimpleResultSet;

import java.io.*;
import java.net.*;
import java.util.zip.*;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpression;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;


public class KmlzProcedureExecution implements ProcedureExecution {

	//private Call command;
                                                                //Lladada de ejemplo desde
la BD:
                                                                //      call
spParseKmlz('http://cartolleida.paeria.es/kml/linea_bus.kml', 'KML',
'/kml:kml/kml:Document/kml:Folder/kml:Placemark',
'DESCRIPTION~~255~~kml:description¡¡COORDINATES~~10000~~kml:LineString/kml:coordinates');

	private String xpathRoot;		//raíz del documento a parsear, ej:
"/kml:kml/kml:Document/kml:Folder/kml:Placemark"
	private String xpathSchema;		//esquema custom:
"campo1~~length1~~xpath1¡¡campo2~~length2~~xpath2" -> ej:
"DESCRIPTION~~255~~kml:description¡¡COORDINATES~~10000~~kml:LineString/kml:coordinates"
                                                                //de momento limitaremos a
sólo dos columnas de retorno: descripcion y coordenadas, mientras no sepamos si se pueden devolver
"recordsets dinámicos"
```

```
        private String fileType;                //'KML' o 'KMZ'
        private String fileUrl;                 //url al kml o kmz remoto, ej:
http://cartolleida.paeria.es/kml/linea_bus.kml
        Iterator<List<?>> results;

    public KmlzProcedureExecution(Call command) {
        //this.command = command;
        this.fileUrl = (String)command.getArguments().get(0).getArgumentValue().getValue();
        this.fileType = (String)command.getArguments().get(1).getArgumentValue().getValue();
        this.xpathRoot = (String)command.getArguments().get(2).getArgumentValue().getValue();
        this.xpathSchema = (String)command.getArguments().get(3).getArgumentValue().getValue();
//        System.out.println("PARAMS:
\n\t"+this.fileUrl+"\n\t"+this.fileType+"\n\t"+this.xpathRoot+"\n\t"+this.xpathSchema);
    }

        @Override
        public void execute() throws TranslatorException {
        List<List<?>> rows = new ArrayList<List<?>>();

        try{
                        URL url = new URL(this.fileUrl);
                        URLConnection connection = url.openConnection();
                        String kml = "";

                        if (this.fileType.equals("KML")) {
                                InputStreamReader stream = new InputStreamReader(
                                                connection.getInputStream());
                                BufferedReader reader = new BufferedReader(stream);
                                String line;
                                while ((line = reader.readLine()) != null) {
                                        kml += line + "\n";
                                }
                        }

                        else if (this.fileType.equals("KMZ")) {
                                ZipInputStream zip = new
ZipInputStream(connection.getInputStream());
                                ZipEntry ze;
                                while ((ze = zip.getNextEntry()) != null) {
                                        if (ze.getName().equals("doc.kml")) {
                                                ByteArrayOutputStream baos = new
ByteArrayOutputStream();

                                                byte[] buffer = new byte[1024];
                                                int count;
                                                while ((count = zip.read(buffer)) != -1) {
                                                        baos.write(buffer, 0, count);
                                                }
                                                byte[] bytes = baos.toByteArray();
                                                kml = new String(bytes, "UTF-8");
                                        }
                                }
                        }
                        //System.out.println(kml);
```

```
                        XPathFactory xpFactory = XPathFactory.newInstance();
                        XPath xpath = xpFactory.newXPath();
                        xpath.setNamespaceContext(new KmlNamespaceContext());

                        XPathExpression expr = xpath.compile(this.xpathRoot);

                        //Para evitar cast exception no pasar a la expresión el input reader hay que
crear un documento
                        DocumentBuilderFactory builderFactory =
DocumentBuilderFactory.newInstance();
                        DocumentBuilder documentBuilder = builderFactory.newDocumentBuilder();
                        Document someXML = documentBuilder.parse(new InputSource(new
StringReader(kml)));
                        Object result = expr.evaluate(someXML, XPathConstants.NODESET);
                        NodeList places = (NodeList) result;

                        String[] rowSchema=this.xpathSchema.split("¡¡");

                        //SimpleResultSet rs = new SimpleResultSet();

                        List<String> fieldsSchemaList=new ArrayList<String>();

                        for(int i=0; i<rowSchema.length; i++){
                                String[] fieldSchema=rowSchema[i].split("~~");
                                fieldsSchemaList.add(fieldSchema[2]);
                                //rs.addColumn(fieldSchema[0], Types.VARCHAR,
Integer.parseInt(fieldSchema[1]), 0);          //Parte dinámica del código legacy, no sé si se
puede trasladar
                        }

                        for (int i = 0; i < places.getLength(); i++) {
                                Node place = places.item(i);
                                List<String> rowList=new ArrayList<String>();

                                for(int j=0; j<fieldsSchemaList.size(); j++){
                                        String tmp=((Node) xpath.evaluate(fieldsSchemaList.get(j),
place,
                                                        XPathConstants.NODE)).getTextContent();
                                        rowList.add(tmp);
                                }

                                //String[] row=rowList.toArray(new String[rowList.size()]);
                                //rs.addRow((Object[]) row);

                                rows.add(rowList);
                        }
                } catch(MalformedURLException mue){
                    throw new TranslatorException(mue, mue.getMessage());
                } catch(IOException e) {
                    throw new TranslatorException(e, e.getMessage());
                } catch(XPathExpressionException xpe){
                        throw new TranslatorException(xpe, xpe.getMessage());
                } catch (ParserConfigurationException pce) {
                        throw new TranslatorException(pce, pce.getMessage());
```

```
                } catch (SAXException sxe) {
                        throw new TranslatorException(sxe, sxe.getMessage());
                }

                //return rs;
                this.results=rows.iterator();

        }

        @Override
        public List<?> next() throws TranslatorException, DataNotAvailableException {
        if (results.hasNext()) {
            return results.next();
        }
        return null;
        }

        @Override
        public void cancel() throws TranslatorException {
                // TODO Auto-generated method stub

        }

        @Override
        public void close() {
                // TODO Auto-generated method stub

        }

        @Override
        public List<?> getOutputParameterValues() throws TranslatorException {
                // TODO Auto-generated method stub
                return null;
        }

}
```

KmlNamespaceContext.java

```
package org.teiid.translator.kmlz;

import java.util.Iterator;
import javax.xml.*;
import javax.xml.namespace.NamespaceContext;

/**
* A class to return the appropriate Namespace context for xPath execution
* against KML files
*/
public class KmlNamespaceContext implements NamespaceContext {
        /**
         * A method to return the Namespace URI of a given namespace prefix
         *
```

```
         * @param prefix
         *            the prefix to math
         *
         * @return the matched namespace URI
         */
        @Override
        public String getNamespaceURI(String prefix) {
                if (prefix == null) {
                        throw new NullPointerException("Null prefix");
                } else if ("kml".equals(prefix)) {
                        return "http://www.opengis.net/kml/2.2";
                } else if ("atom".equals(prefix)) {
                        return "http://www.w3.org/2005/Atom";
                } else if ("xml".equals(prefix)) {
                        return XMLConstants.XML_NS_URI;
                } else {
                        return XMLConstants.XML_NS_URI;
                }
        }

        /**
         * This method isn't necessary for XPath processing.
         */
        @Override
        public String getPrefix(String uri) {
                throw new UnsupportedOperationException();
        }

        /**
         * This method isn't necessary for XPath processing.
         */
        @SuppressWarnings("rawtypes")
    @Override
        public Iterator getPrefixes(String uri) {
                throw new UnsupportedOperationException();
        }
}
```

---

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <artifactId>translator-kmlz</artifactId>
    <groupId>org.jboss.teiid.connectors</groupId>
    <name>KML/KMZ Translator</name>
    <description>Translator to parse KML and KMZ files</description>
        <version>0.0.1</version>
        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
           <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
```

```
</properties>
<dependencies>
    <dependency>
        <groupId>org.jboss.teiid</groupId>
        <artifactId>teiid-api</artifactId>
        <scope>provided</scope>
                    <version>8.2.0</version>
    </dependency>
    <dependency>
        <groupId>org.jboss.teiid</groupId>
        <artifactId>teiid-common-core</artifactId>
        <scope>provided</scope>
                    <version>8.2.0</version>
    </dependency>
    <dependency>
        <groupId>javax.resource</groupId>
        <artifactId>connector-api</artifactId>
        <scope>provided</scope>
                    <version>1.5</version>
            </dependency>
            <dependency>
                    <groupId>junit</groupId>
                    <artifactId>junit</artifactId>
                    <version>3.8.1</version>
                    <scope>test</scope>
            </dependency>
            <dependency>
                    <groupId>javax.xml</groupId>
                    <artifactId>jaxp-api</artifactId>
                    <version>1.4.2</version>
            </dependency>
    </dependencies>

<build>
    <outputDirectory>target/classes</outputDirectory>
    <resources>
        <resource>
            <directory>src/main/resources</directory>
            <filtering>true</filtering>
            <includes>
                <include>**/*.xml</include>
                <include>**/*.properties</include>
            </includes>
        </resource>
        <resource>
            <directory>src/main/resources</directory>
            <filtering>false</filtering>
            <excludes>
                <exclude>**/*.xml</exclude>
                <exclude>**/*.properties</exclude>
            </excludes>
        </resource>
    </resources>
            <plugins>
```

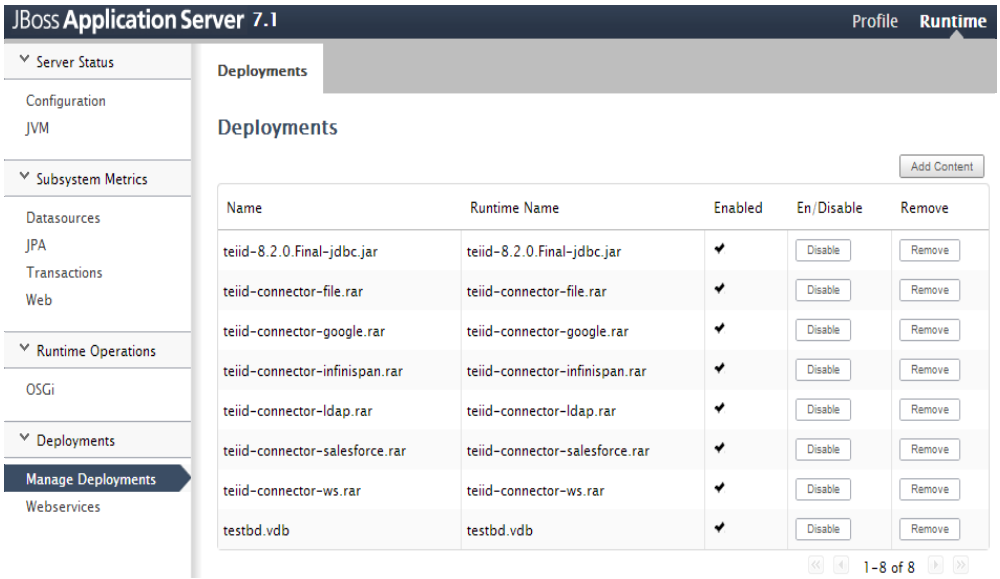
```
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <configuration>
                        <source>1.7</source>
                        <target>1.7</target>
                    </configuration>
                    <version>3.0</version>
                </plugin>
            </plugins>
    </build>
</project>
```

TEIID en máquina remota

- Instalar JBOSS (descomprimir tar.gz a /opt o /usr/local y dar permisos a usuario regular).
- Instalar TEIID (descomprimir zip en el directorio de JBOSS).
- Crear script de ejecución en directorio bin/ de JBOSS_HOME standalone_teiid.sh que simplemente hace: `./standalone.sh --server-config standalone-teiid.xml`
- Probar a arrancar servidor.
- Deployment de las VDB's a través de la admin console de JBOSS (http://ip_addr:9990).
    - Hay que permitir acceso a la management console desde el exterior, para ello en el standalone-teiid.xml modificaremos lo siguiente:

```
<interfaces>
        <interface name="management">
            <inet-address value="${jboss.bind.address.management:0.0.0.0}"/>
        </interface>
        <interface name="public">
            <inet-address value="${jboss.bind.address:0.0.0.0}"/>
        </interface>
        <interface name="unsecure">
            <inet-address value="${jboss.bind.address.unsecure:0.0.0.0}"/>
        </interface>
</interfaces>
```

    - Crear usuario válido para la management console mediante el script de jboss bin/add-user.sh (jboss/jb0ss).
    - Deploy de la vdb mediante la interface de la management console:



Ficheros XML remotos

- Añadir los datasources al fichero standalone-teiid.xml (copiar los nombres de los recursos de las sources en el TEIID Designer. Ejemplo:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.0">
        <resource-adapters>
                <resource-adapter>
                        <archive>
                                teiid-connector-file.rar
                        </archive>
                        <transaction-support>NoTransaction</transaction-support>
                        <connection-definitions>
                                <connection-definition
class-name="org.teiid.resource.adapter.file.FileManagedConnectionFactory"
jndi-name="java:/bus_lines_source" enabled="true" pool-name="bus_lines_source">
                                        <config-property name="ParentDirectory">
                                                C:/Temp
                                        </config-property>
                                </connection-definition>

                        </connection-definitions>
                </resource-adapter>
                <resource-adapter>
                        <archive>
                                teiid-connector-ws.rar
                        </archive>
                        <transaction-support>NoTransaction</transaction-support>
                        <connection-definitions>
                                <connection-definition
class-name="org.teiid.resource.adapter.ws.WSManagedConnectionFactory"
jndi-name="java:/centres_source_2" enabled="true" pool-name="centres_source_2">
                                        <config-property name="EndPoint">

http://192.168.80.129:8088/files/Centres_educacio.xml
                                        </config-property>
                                </connection-definition>
```

```
                        </connection-definitions>
                </resource-adapter>
                <resource-adapter>
                        <archive>
                                teiid-connector-ws.rar
                        </archive>
                        <transaction-support>NoTransaction</transaction-support>
                        <connection-definitions>
                                <connection-definition
class-name="org.teiid.resource.adapter.ws.WSManagedConnectionFactory"
jndi-name="java:/PREVIEW_226fecaa-31cc-4f01-86c6-c3ae3f73a3c8_test_soap_sources_CountryInfoService
" enabled="true"
pool-name="PREVIEW_226fecaa-31cc-4f01-86c6-c3ae3f73a3c8_test_soap_sources_CountryInfoService">
                                        <config-property name="SecurityType">
                                                None
                                        </config-property>
                                        <config-property name="EndPoint">

http://www.oorsprong.org/websamples.countryinfo/CountryInfoService.wso
                                        </config-property>
                                </connection-definition>
                        </connection-definitions>
                </resource-adapter>
        </resource-adapters>
</subsystem>
```

Ficheros CSV remotos

Debería ser igual que en local.

Bases de datos remotas (SQLServer)

- Copiar driver a la instalación de JBOSS: Por ejemplo fichero *sqljdbc4.jar* a carpeta `JBOSS_HOME/modules/com/microsoft/sqlserver/jdbc/main` (crear directorios necesarios).
- Añadir fichero *module.xml* al mismo directorio con el contenido:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="com.microsoft.sqlserver.jdbc">

    <resources>
        <resource-root path="sqljdbc4.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
        <module name="javax.servlet.api" optional="true"/>
    </dependencies>
</module>
```

- Modificar el standalone-teiid.xml para añadir el datasource que toque y el driver:

```
<datasource jndi-name="java:/SQL_Server_Lleida" pool-name="SQL_Server_Lleida"
```

```
enabled="true">
        <connection-url>jdbc:sqlserver://194.224.186.141:1433;databaseName=opendai</con
nection-url>
        <driver>sqljdbc</driver>
        <security>
                <user-name>opendai_external_user</user-name>
                <password>opendaiLleida</password>
        </security>
</datasource>
```

driver:

```
<drivers>
...
        <driver name="sqljdbc" module="com.microsoft.sqlserver.jdbc">

<driver-class>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver-class>
        </driver>
</drivers>
```

- Reiniciar TEIID.
- Conectar de manera remota al servidor para probar la base de datos: