Openbravo WebServices

External Point Of Sale

16 August 2006

Revision 1.1

Visit us at www.openbravo.com

Table of Contents

I. Introduction

Openbravo's webservices provide the interface that allows Openbravo's integration with other systems (for example with a Virtual Shop and a POS Terminal) and hereby share the management of products and orders obtaining this way a global integrated and more complete solution.

We will have to define the classes of the objects that will form a part of this interface (Products, Orders and Business Partner) as well as the methods that will be able to be invoked.

II. Product definition.

We define the Products at this way:

II.1 Product type

Field	*Type*	*Description*
id	*int*	*ERP Product's identifier.*
name	*Java.lang.String*	*Product 's name.*
number	*Java.lang.String*	*Product number.*
description	*Java.lang.String*	*Product's description.*
listPrice	*double*	*Product's price without taxes and discounts.*
purchasePrice	*double*	*Product price for sale.*
tax	*Tax*	*Tax assigned to this product.*
imageUrl	*Java.lang.String*	*URL to download product's image.*
ean	*Java.lang.String*	*Product's barcode.*
category	*Category*	*Product's category*

II.2 Category type

Field	*Type*	*Description*
id	*int*	*ERP category's identifier.*
name	*Java.lang.String*	*Category's name.*
description	*Java.lang.String*	*Category's description.*

II.3 Tax type

Field	*Type*	*Description*
id	*int*	*Tax's identifier.*
name	*Java.lang.String*	*Tax's name*
percentage	*double*	*Values between -100 and 100.*

III. Order definition.

We will define the Order type for the interface at this way:

III.1 Order type

Field	*Type*	*Description*
orderId	*OrderIdentifier*	*Order's identifier for Virtual Shop and POS Terminal.*
lines	*OrderLine []*	*OrderLine's array.*
state	*int*	*Order's state identifier.*
bpartner	*BPartner*	*Client that generates this order.*
payments	*Payment []*	*Payments array*

III.2 OrderIdentifier type

Field	*Type*	*Description*
document_no	*String*	*Order identifier for Virtual Shop and POS Terminal.*
dateNew	*Java.util.Date*	*Insert date.*

III.3 OrderLine type

Field	*Type*	*Description*
orderLineId	*int*	*OrderLine identifier for ERP.*
productId	*int*	*Product identifier.*
units	*double*	*Number of product for this line.*

Field	Type	Description
price	double	Price in PriceList for this product.
taxId	int	Tax applied (Tax applied identifier).

III.4 Payment type

Field	Type	Description
amount	double	Amount satisfied with the payment.
paymentType	String	Way of payment.

IV. Business Partner definition.

We will define the Business partner in BPartner type.

IV.1 BPartner type

Field	*Type*	*Description*
id	*String*	*Business Partner identifier for Virtual Shop.*
name	*String*	*Business Partner name.*
country	*String*	*Business Partner Country.*
region	*String*	*Business Partner region.*
city	*String*	*Business Partner city.*
postal	*String*	*Business Partner postal code.*
address1	*String*	*Business Partner address' first line.*
address2	*String*	*Business Partner address' second line.*

V. Security

For the implementation of the security a user's validation is realized in every request that the ERP receives from the webService, for what in every method we have to pass a user with the necessary permissions and his ciphered password (such it is stored in database).

This user has to have permission to be able to execute the task of the ExternalPOS (org.openbravo.erpCommon.ws.externalSales.ExternalSales) inside the ERP

VI. API External Sales

For Openbravo's integration with other systems we have developed an API named "ExternalSales" that takes charge providing the mechanisms necessary for the integration of Orders, Products and Business Partner.

The technology used on that the implementation of the API is based is AXIS that is WebServices's implementation developed by Apache of easy integration in Tomcat.

VII. Interface method's

VII.1 getProductsCatalog

Nombre función	***getProductsCatalog***	
Parameters	*entityId:int*	*Entity identifier for the Virtual Shop or the POS Terminal on the ERP.*
	OrganizationId: int	*Organization identifier for the Virtual Shop or the POS Terminal on the ERP.*
	salesChannel:int	*Sales channel.* *1. Virtual Shop = 1* *2. POS Terminal = 2*
	username:String	*ERP user.*
	password:String	*ERP user's ciphered password*
Return	*Product []*	*Products array.*
PreConditions	--	
PostConditions	*The Virtual Shop or the Terminal POS can update his catalog of products with the obtained one of the ERP.*	
Functionality	*It has to return the list of products according to the channel of sale and the consulted entity.* *Every product includes, specific information, the information of the category to which it belongs and the tax that partner has in the ERP. For more details on the Product Type see the paragraph "Product definition".*	

VII.2 uploadOrders

Function Name	***uploadOrders***	
Parameters	*entityId:int*	*Entity identifier for the Virtual Shop or the POS Terminal on the ERP.*
	organizationId: int	*Organization identifier for the Virtual Shop or the POS Terminal on the ERP.*
	salesChannel:int	*Sales channel.* *1. Virtual Shop = 1* *2. POS Terminal = 2*
	username:String	*ERP user.*
	password:String	*ERP user's ciphered password*
	newOrders:Order []	*Order's list to be uploaded on the ERP. They are all those new orders that have been generated in the Virtual Shop/POS Terminal from the last synchronization of orders.*
Return	*void*	*--*
PreConditions	*The product's catalog has been synchronized previously.*	
PostConditions	*ERP store new order's information.*	
Functionality	*Insert into ERP new order's generated in Virtual Shop or POS Terminal.* *It has to complete the next process:* *1. Validate that the client associated with the new Order exists (in case of the Terminal POS the client will be dummy).* ● *If the client doesn't exist it will be inserted.* *2. Save the Order.* *The order will be saved in the ERP with the specified values by the*	

Function Name	***uploadOrders***
	WebService with the object Order, more certain values that will be informed about automatic form. Order values will be: *1.* ***Entity****. WebService's param.* *2.* ***Organization****. WebService's param.* *3.* ***Doc. Num.*** *Openbravo generated.* *4.* ***Business Partner****. Order's client.* *5.* ***PriceList****. Assigned by ERP and will be the Entity/Organization's default priceList.* *6.* ***Currency****. Assigned by ERP and will be the Entity/Organization's default currency* *7.* ***Warehouse****. Assigned by ERP and will be the Entity/Organization's default warehouse* *8.* ***Order Date****. Order.dateNew.* *9.* ***Payment type****. Order.paymentType.* *For every OrderLine:* *1.* ***Product****. OrderLine.productId.* *2.* ***Unit****. OrderLine.units.* *3.* ***Tax****. OrderLine.tax.* *4.* ***Price****. OrderLine.price.* *5.* ***UOM****. Assigned by ERP and will be the Product's default unit of measure* *Others:* ● *There exist other values of the order that is preferable adapt to the suggestions proposed by Openbravo.*

VII.3 getOrders

Function Name	***getOrders***	
Parameters	*entityId:int*	*Entity identifier for the Virtual Shop or the POS Terminal on the ERP.*
	organizationId: int	*Organization identifier for the Virtual Shop or the POS Terminal on the ERP.*
	username:String	*ERP user.*
	password:String	*ERP user's ciphered password*
	orderIds: OrderIdentifier []	*OrderIdentifier's array for the orders that we want to consult.*
Return	*Order []*	*List of orders requested.*
Functionality	*It has to return an Order type array. With the information returned by this service, the state of the orders will be updated in the Terminal POS and the Virtual Shop.*	

VIII. Client

Once installed Openbravo application we can verify if webservices are working writing the next sentence into the browser:

http://localhost:8880/openbravo/services/ExternalSales?wsdl

if we suppose that we have installed it into a local host (localhost) and a context called openbravo.

With this command we will obtain as response the file WSDL of our webservice:

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions
targetNamespace="http://localhost:8880/openbravo/services/Externa
lSales" xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8880/openbravo/services/ExternalSale
s"
xmlns:intf="http://localhost:8880/openbravo/services/ExternalSale
s" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <!--
WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)
  -->
- <wsdl:types>
- <schema
targetNamespace="http://localhost:8880/openbravo/services/Externa
lSales" xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/"
/>
- <complexType name="Category">
- <sequence>
  <element name="description" nillable="true" type="xsd:string"
/>
  <element name="id" type="xsd:int" />
  <element name="name" nillable="true" type="xsd:string" />
  </sequence>
  </complexType>
- <complexType name="Tax">
- <sequence>
  <element name="id" type="xsd:int" />
  <element name="name" nillable="true" type="xsd:string" />
  <element name="percentage" type="xsd:double" />
  </sequence>
  </complexType>
- <complexType name="Product">
- <sequence>
  <element name="category" nillable="true" type="impl:Category"
/>
  <element name="description" nillable="true" type="xsd:string"
/>
  <element name="ean" nillable="true" type="xsd:string" />
  <element name="id" type="xsd:int" />
  <element name="imageUrl" nillable="true" type="xsd:string" />
  <element name="listPrice" type="xsd:double" />
```

```
  <element name="name" nillable="true" type="xsd:string" />
  <element name="number" nillable="true" type="xsd:string" />
  <element name="purchasePrice" type="xsd:double" />
  <element name="tax" nillable="true" type="impl:Tax" />
  </sequence>
  </complexType>
- <complexType name="ArrayOfProduct">
- <complexContent>
- <restriction base="soapenc:Array">
  <attribute ref="soapenc:arrayType"
wsdl:arrayType="impl:Product[]" />
  </restriction>
  </complexContent>
  </complexType>
- <complexType name="BPartner">
- <sequence>
  <element name="address1" nillable="true" type="xsd:string" />
  <element name="address2" nillable="true" type="xsd:string" />
  <element name="city" nillable="true" type="xsd:string" />
  <element name="country" nillable="true" type="xsd:string" />
  <element name="id" nillable="true" type="xsd:string" />
  <element name="name" nillable="true" type="xsd:string" />
  <element name="postal" nillable="true" type="xsd:string" />
  <element name="region" nillable="true" type="xsd:string" />
  </sequence>
  </complexType>
- <complexType name="OrderLine">
- <sequence>
  <element name="orderLineId" type="xsd:int" />
  <element name="price" type="xsd:double" />
  <element name="productId" type="xsd:int" />
  <element name="taxId" type="xsd:int" />
  <element name="units" type="xsd:double" />
  </sequence>
  </complexType>
- <complexType name="ArrayOfOrderLine">
- <complexContent>
- <restriction base="soapenc:Array">
  <attribute ref="soapenc:arrayType"
wsdl:arrayType="impl:OrderLine[]" />
  </restriction>
  </complexContent>
  </complexType>
- <complexType name="OrderIdentifier">
- <sequence>
  <element name="dateNew" nillable="true" type="xsd:dateTime" />
  <element name="documentNo" nillable="true" type="xsd:string" />
  </sequence>
  </complexType>
- <complexType name="Payment">
- <sequence>
  <element name="amount" type="xsd:double" />
  <element name="paymentType" nillable="true" type="xsd:string"
/>
  </sequence>
  </complexType>
- <complexType name="ArrayOfPayment">
- <complexContent>
- <restriction base="soapenc:Array">
  <attribute ref="soapenc:arrayType"
wsdl:arrayType="impl:Payment[]" />
  </restriction>
```

```
  </complexContent>
  </complexType>
- <complexType name="Order">
- <sequence>
  <element name="BPartner" nillable="true" type="impl:BPartner"
/>
  <element name="lines" nillable="true"
type="impl:ArrayOfOrderLine" />
  <element name="orderId" nillable="true"
type="impl:OrderIdentifier" />
  <element name="payment" nillable="true"
type="impl:ArrayOfPayment" />
  <element name="state" type="xsd:int" />
  </sequence>
  </complexType>
- <complexType name="ArrayOfOrder">
- <complexContent>
- <restriction base="soapenc:Array">
  <attribute ref="soapenc:arrayType"
wsdl:arrayType="impl:Order[]" />
  </restriction>
  </complexContent>
  </complexType>
- <complexType name="ArrayOfOrderIdentifier">
- <complexContent>
- <restriction base="soapenc:Array">
  <attribute ref="soapenc:arrayType"
wsdl:arrayType="impl:OrderIdentifier[]" />
  </restriction>
  </complexContent>
  </complexType>
  </schema>
  </wsdl:types>
- <wsdl:message name="getOrdersResponse">
  <wsdl:part name="getOrdersReturn" type="impl:ArrayOfOrder" />
  </wsdl:message>
- <wsdl:message name="getOrdersRequest">
  <wsdl:part name="in0" type="xsd:int" />
  <wsdl:part name="in1" type="xsd:int" />
  <wsdl:part name="in2" type="impl:ArrayOfOrderIdentifier" />
  </wsdl:message>
- <wsdl:message name="uploadOrdersRequest">
  <wsdl:part name="in0" type="xsd:int" />
  <wsdl:part name="in1" type="xsd:int" />
  <wsdl:part name="in2" type="xsd:int" />
  <wsdl:part name="in3" type="impl:ArrayOfOrder" />
  </wsdl:message>
- <wsdl:message name="getProductsCatalogResponse">
  <wsdl:part name="getProductsCatalogReturn"
type="impl:ArrayOfProduct" />
  </wsdl:message>
  <wsdl:message name="uploadOrdersResponse" />
- <wsdl:message name="getProductsCatalogRequest">
  <wsdl:part name="in0" type="xsd:int" />
  <wsdl:part name="in1" type="xsd:int" />
  <wsdl:part name="in2" type="xsd:int" />
  </wsdl:message>
- <wsdl:portType name="ExternalSalesImpl">
- <wsdl:operation name="getProductsCatalog" parameterOrder="in0
in1 in2">
  <wsdl:input message="impl:getProductsCatalogRequest"
name="getProductsCatalogRequest" />
```

```
  <wsdl:output message="impl:getProductsCatalogResponse"
name="getProductsCatalogResponse" />
  </wsdl:operation>
- <wsdl:operation name="uploadOrders" parameterOrder="in0 in1 in2
in3">
  <wsdl:input message="impl:uploadOrdersRequest"
name="uploadOrdersRequest" />
  <wsdl:output message="impl:uploadOrdersResponse"
name="uploadOrdersResponse" />
  </wsdl:operation>
- <wsdl:operation name="getOrders" parameterOrder="in0 in1 in2">
  <wsdl:input message="impl:getOrdersRequest"
name="getOrdersRequest" />
  <wsdl:output message="impl:getOrdersResponse"
name="getOrdersResponse" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="ExternalSalesSoapBinding"
type="impl:ExternalSalesImpl">
  <wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="getProductsCatalog">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="getProductsCatalogRequest">
  <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://externalSales.ws.erpCommon.openbravo.org"
use="encoded" />
  </wsdl:input>
- <wsdl:output name="getProductsCatalogResponse">
  <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8880/openbravo/services/ExternalSales
" use="encoded" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="uploadOrders">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="uploadOrdersRequest">
  <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://externalSales.ws.erpCommon.openbravo.org"
use="encoded" />
  </wsdl:input>
- <wsdl:output name="uploadOrdersResponse">
  <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8880/openbravo/services/ExternalSales
" use="encoded" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getOrders">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="getOrdersRequest">
  <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://externalSales.ws.erpCommon.openbravo.org"
use="encoded" />
  </wsdl:input>
- <wsdl:output name="getOrdersResponse">
  <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
namespace="http://localhost:8880/openbravo/services/ExternalSales
" use="encoded" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="ExternalSalesImplService">
- <wsdl:port binding="impl:ExternalSalesSoapBinding"
name="ExternalSales">
  <wsdlsoap:address
location="http://localhost:8880/openbravo/services/ExternalSales"
/>
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

At this moment we have two options for the utilization of the webservices:

I. Download the client from Openbravo's page and the files form with the characteristics of our server web (replacing the name and port of the server with the own ones of our installation).

II. Generate the files of the client by means of the task of ant "wsdl2java" that we will add to the build.xml of our project.

```
        <property file="build.properties"/>
        <path id="axis.classpath">
          <fileset dir="%AXIS_LIB%/lib/">
            <include name="**/*.jar" />
          </fileset>
        </path>
        <taskdef resource="axis-tasks.properties"
        classpathref="axis.classpath" />
        <target name="wsdl2java">
          <axis-wsdl2java url="${wsdl_definitions_url}"
        output="${generated.dir}" >
            <mapping namespace="${web_service_url}"
        package="${package_name}"/>
          </axis-wsdl2java>
        </target>
```

This task will generate all needed files with the correct configuration.

Now we can begin to use the Openbravo's webservice:

```
ExternalSalesImpl externalSales = new
ExternalSalesImplServiceLocator().getExternalSales();

Product[] products =
externalSales.getProductsCatalog(cliente,organizacion,pos);

...
externalSales.uploadOrders(cliente,organizacion,pos,
orders_to_upload);

OrderIdentifier[] orderIdentifiers = new OrderIdentifier[1];
orderIdentifiers[0] = createOrderIdentifier("12345678”);
Order[] orders =
externalSales.getOrders(cliente,organizacion,orderIdentifiers);
```