Openbravo WebServices

External Point Of Sale

16 Agosto 2006

Revisión 1.1

openbravo™
opening ERP's future!

Visit us at www.openbravo.com

Tabla de Contenidos

I.Introducción ... 3
II.Definición del producto ... 4
II.1Tipo Product ... 4
II.2Tipo Category ... 4
II.3Tipo Tax ... 5
III.Definición del pedido ... 6
III.1Tipo Order ... 6
III.2Tipo OrderIdentifier ... 6
III.3Tipo OrderLine ... 6
III.4Tipo Payment ... 7
IV.Definición de tercero (BPartner) ... 8
IV.1Tipo BPartner ... 8
V.Seguridad ... 9
VI.API External Sales ... 10
VII.Métodos del interfaz ... 11
VII.1getProductsCatalog ... 11
VII.2uploadOrders ... 12
VII.3getOrders ... 14
VIII.Cliente ... 15

I. Introducción

El objetivo de este webservices es proporcionar el interfaz que permita la integración de Openbravo con otros sistemas (por ejemplo con una Tienda Virtual y un TPV) y de esta manera compartan la gestión de productos y pedidos obteniendo así una solución global integrada y más completa.

Deberemos definirnos las clases de los objetos que formarán parte de ese interfaz (Productos, Pedidos y Clientes) así como los métodos que podrán ser invocados.

II. Definición del producto.

Definimos el producto de la siguiente manera:

II.1 Tipo Product

Campo	*Tipo*	*Descripción*
id	*int*	*Identificador del producto en el ERP.*
name	*Java.lang.String*	*Nombre del producto.*
number	*Java.lang.String*	*Número de producto.*
description	*Java.lang.String*	*Descripción del producto.*
listPrice	*double*	*Precio del producto sin impuestos y sin tener en cuenta descuentos.*
purchasePrice	*double*	*Precio de compra del producto.*
tax	*Tax*	*Impuesto asignado a este producto.*
imageUrl	*Java.lang.String*	*URL desde la cual podemos descargar la imagen asociada a este producto.*
ean	*Java.lang.String*	*Código de barras del producto.*
category	*Category*	*Categoría a la que pertenece el producto.*

II.2 Tipo Category

Campo	*Tipo*	*Descripción*
id	*int*	*Identificador de la categoría en el ERP.*
name	*Java.lang.String*	*Nombre de la categoría.*

Campo	Tipo	Descripción
description	*Java.lang.String*	*Descripción de la categoría.*

II.3 Tipo Tax

Campo	Tipo	Descripción
id	*int*	*Identificador del impuesto.*
name	*Java.lang.String*	*Nombre del impuesto.*
percentage	*double*	*Valor entre -100 y 100.*

III. Definición del pedido.

Definiremos el Tipo Order del interfaz de integración de la siguiente manera:

III.1 Tipo Order

Campo	*Tipo*	*Descripción*
orderId	*OrderIdentifier*	*Identificador de la orden en la Tienda Virtual y el TPV.*
lines	*OrderLine []*	*Array de líneas de orden*
state	*int*	*Identificador del estado de la orden.*
bpartner	*BPartner*	*Cliente que ha generado la orden.*
payments	*Payment []*	*Array con los pagos realizados sobre la orden.*

III.2 Tipo OrderIdentifier

Campo	*Tipo*	*Descripción*
document_no	*String*	*Identificador de la orden en la Tienda Virtual y el TPV.*
dateNew	*Java.util.Date*	*Fecha de alta de la orden.*

III.3 Tipo OrderLine

Campo	*Tipo*	*Descripción*
orderLineId	*int*	*Identificador de la linea de orden en el ERP.*
productId	*int*	*Identificador del producto.*
units	*double*	*Cantidad de productos en la línea.*

Campo	Tipo	Descripción
price	double	Importe de lista del producto vendido.
taxId	int	Impuesto aplicado (Identificador del impuesto aplicado).

III.4 Tipo Payment

Campo	Tipo	Descripción
amount	double	Importe satisfecho con el pago.
paymentType	String	Forma de pago.

IV. Definición de tercero (BPartner).

El tercero siempre será de tipo BPartner.

IV.1 Tipo BPartner

Campo	*Tipo*	*Descripción*
id	*String*	*Identificador de cliente generado en la Tienda Virtual.*
name	*String*	*Nombre del cliente.*
country	*String*	*País al que pertenece el cliente.*
region	*String*	*Estado al que pertenece el cliente.*
city	*String*	*Población del cliente.*
postal	*String*	*Código postal del cliente.*
address1	*String*	*Primera linea de la dirección del cliente.*
address2	*String*	*Segunda línea de la dirección del cliente.*

V. Seguridad

Para la implementación de la seguridad se realiza una validación de usuario en cada petición que recibe el ERP desde el webService, por lo que en cada método se ha de pasar tanto un usuario con los permisos necesarios como su contraseña cifrada (tal y como se almacena en base de datos).

Este usuario ha de tener permiso para poder ejecutar la tarea del ExternalPOS (org.openbravo.erpCommon.ws.externalSales.ExternalSales) dentro del ERP.

VI. API External Sales

Para la posible integración de Openbravo con otros sistemas se ha desarrollado una API denominada “ExternalSales” que se encarga de proveer los mecanismos necesarios para la integración de Pedidos, Productos y Terceros.

La tecnología empleada sobre la que se basa la implementación de la API es AXIS que es una implementación de WebServices desarrollada por Apache de fácil integración en Tomcat.

VII. Métodos del interfaz

VII.1 getProductsCatalog

Nombre función	***getProductsCatalog***	
Parámetros	*entityId:int*	*Identificador de la entidad a la que pertenece la Tienda o TPV en el ERP.*
	organizationId: int	*Identificador de la Tienda o TPV en el ERP.*
	salesChannel:int	*Canal de venta.* *1. Tienda Virtual = 1* *2. TPV = 2*
	username:String	
	password:String	
Retorno	*Product []*	*Array de productos.*
PreCondiciones	--	
PostCondiciones	*La Tienda Virtual o el TPV pueden actualizar su catálogo de productos con el obtenido del ERP.*	
Funcionalidad	*Ha de retornar la lista de productos según el canal de venta y la entidad consultada.* *Cada producto incluye, a parte de su información específica, la información de la categoría a la que pertenece y el impuesto que tiene asociado en el ERP. Para más detalles sobre el tipo Producto ver el apartado "Definición de producto".*	

VII.2 uploadOrders

Nombre función	***uploadOrders***	
Parámetros	*entityId:int*	*Identificador de la entidad a la que pertenece la Tienda o TPV en el ERP.*
	organizationId: int	*Identificador de la Tienda o TPV en el ERP.*
	salesChannel:int	*Canal de venta.* *1. Tienda Virtual = 1* *2. TPV = 2*
	username:String	
	password:String	
	newOrders:Order []	*Lista de órdenes que se han de dar de alta en el ERP. Son todas aquellas nuevas órdenes que se han generado en la tienda virtual/TPV desde la última sincronización de órdenes.*
Retorno	*void*	--
PreCondiciones	*Se ha sincronizado el catálogo de productos con anterioridad.*	
PostCondiciones	*El ERP almacena la información de las nuevas órdenes.*	
Funcionalidad	*Dar de alta las nuevas órdenes generadas en la Tienda Virtual o TPV en el ERP. Para ello se ha de completar la siguiente secuencia:* *1. Validar que el cliente asociado a la nueva Orden existe (en el caso del TPV el cliente será dummy).* ● *Si no existe se ha de dar de alta el cliente.* *2. Guardar la orden.* *La orden se ha de guardar en el ERP con los valores especificados vía el WebService con el objeto Order, más ciertos valores que se informarán de*	

Nombre función	***uploadOrders***
	forma automática. Los valores de la orden serán: *1. **Entidad**. El valor del parámetro recibido en el WebService.* *2. **Organización**. El valor del parámetro recibido en el WebService.* *3. **Num. Doc**. Generado por Openbravo.* *4. **Tercero**. Es el cliente de asociado a la orden enviada.* *5. **Tarifa**. La tiene que asignar el ERP y debería ser la de defecto para esa Entidad/Organización.* *6. **Moneda**. La asigna el ERP por defecto en función de la Entidad/Organización.* *7. **Almacén**. Lo asigna el ERP en función de la Entidad/Organización.* *8. **Fecha pedido**. Orden.dateNew.* *9. **Forma de Pago**. Orden.paymentType.* *Por cada Linea de la Orden:* *1. **Producto**. OrderLine.productId.* *2. **Cantidad**. OrderLine.units.* *3. **Impuesto**. OrderLine.tax.* *4. **Precio**. OrderLine.price.* *5. **Unidad**. Lo asigna el ERP en función del producto.* *Otros valores:* *● Existen otros valores de la orden que es preferible se adapten a la recomendaciones propuestas por Openbravo.*

VII.3 getOrders

Nombre función	***getOrders***	
Parámetros	*entityId:int*	*Identificador de la entidad a la que pertenece la Tienda o TPV en el ERP.*
	organizationId: int	*Identificador de la Tienda o TPV en el ERP.*
	username:String	
	password:String	
	orderIds: OrderIdentifier []	*Array de enteros que representan los identificadores de las órdenes que queremos consultar.*
Retorno	*Order []*	*Lista de órdenes solicitada.*
Funcionalidad	*Ha de retornar los objetos de tipo Order solicitados vía el parámetro orderIds. Con la información devuelta por este servicio se actualizará el estado de las órdenes en el TPV y la Tienda Virtual.*	

VIII. Cliente

Una vez instalada la aplicación Openbravo podemos comprobar si los webservices están funcionando correctamente escribiendo la siguiente sentencia en nuestro navegador:

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

suponiendo que la tengamos instalada sobre un servidor local (localhost) y en un contexto de nombre openbravo.

Con este comando obtendremos como respuesta el fichero WSDL de nuestro 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>
```

En este momento tenemos dos opciones para la utilización de los webservices:

I. Descargar el cliente desde la página de Openbravo y configurar los ficheros con las características de nuestro servidor web (reemplazando el nombre y puerto del servidor con los propios de nuestra instalación).

II. Generar los ficheros del cliente mediante la task de ant “wsdl2java” que añadiremos al build.xml de nuestro proyecto.

```
        <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>
```

Con lo que nos generará todos los ficheros necesarios.

Con esto ya podemos empezar a utilizar los servicios web, por ejemplo:

```
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);
```