JUNTA DE ANDALUCIA

Consejería de Justicia y Administración Pública

Bus de Conexión

Manual del Programador de Servicio Consulta Datos Ciudadanía

Versión: v01r03

Fecha: 04/12/2007

HOJA DE CONTROL

Título	Bus de Conexión		
Entregable	Manual del Programador de Servicio Consulta Datos Ciudadanía		
Nombre del Fichero	WAN013T_Manual_Programador_Servicio_Consulta_Datos_Ciudadano_0103.doc		
Autor	GFI Informática		
Versión / Edición	v01r03	**Fecha Versión**	04/12/2007
Aprobado por		**Fecha Aprobación**	DD/MM/AAAA
		Nº Total Páginas	017

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Área	Fecha del Cambio
0101	Version Inicial	GFI	GFI	14/11/07
0102	• Se modifica código de ejemplo	GFI	GFI	22/11/07
0103	• Se incluyen archivos de ejemplo • Se especifica el identificador de servicio • Se revisa la especificacion del objeto de salida.	GFI	GFI	04/12/07

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos	Cargo	Área	Nº Copias
Manuel Perera Dominguez	Jefe de Servicio	Consejería de Justicia y Admón. Pública.	1
Manuel Escobar Montes	Director de Proyecto	Consejería de Justicia y Admón. Pública.	1

ÍNDICE

1 INTRODUCCIÓN

1.1 Propósito

El objetivo de este documento es describir la forma de utilización del Servicio de Consulta de datos de ciudadano, en particular, para recuperar la información personal que los ciudadanos hayan introducido a través del alta en Clar@.

1.2 Alcance

Este documento esta dirigido a los equipos de desarrollo que pretendan utilizar los servicios que proporcionen información de datos de contacto de ciudadanos y empresas introducidos en TDC a través de Cl@ra.

2 Elemento BusObject

El elemento busObject va a ser necesario en la invocación de **todos los servicios** y esta estructurado de forma que viaje a traves de los servicios del Bus W@nda, recogiendo toda la información de control, invocacion y posibles errores.

Este objeto estará incluido como parte de todos los objetos que se utilicen en la Entrada/Salida de los servicios del Bus de forma que permita recibir distintos tipos de parámetros. De igual forma a la salida se se incluira el mismo objeto, pero con la posible información de los errores producidos durante la ejecución del servicio.

En el caso concreto del servicio de consulta de interesados el objeto no será necesario construirlo en entrada, pero si vendrá construido en la respuesta y en caso de aparecer errores se incluirá información al respecto (Ver apartado Control de Errores)

3 Funcionamiento general

El sistema Bus de Conexión W@nda va a proporcionar un interfaz de acceso mediante Web Services, la comunicación por tanto se establece mediante protocolo SOAP, actualmente muy difundido. Por otra parte, la conexión se realizará por aplicación, es decir, los clientes de los servicios del Bus serán aplicaciones desarrolladas de forma externa a éste.

El desarrollo de un nuevo sistema cliente que necesite utilizar los servicios del Bus W@nda debe ser capaz de implementar un interfaz de conexión mediante Web Services, para lo cual será necesario utilizar el fichero de descriptor del webservice, **WSDL** (Web Service Description Language) de cada servicio que necesite utilizar.

La forma de obtener el fichero WSDL de cada webservice será simplemente invocando al webservice desde un explorador de internet (Internet Explorer, Mozilla Firefox o similar) mediante la URL correspondiente al Web Service en cada caso.

Teniendo en cuenta que los Web Services en el Bus de Conexión W@nda se implementan mediante archivos con extensión .jpd, una llamada tipo para obtener el WSDL será de la siguiente forma:

```
http://<host>:<puerto>/.../nombreWS.jpd?WSDL=
```

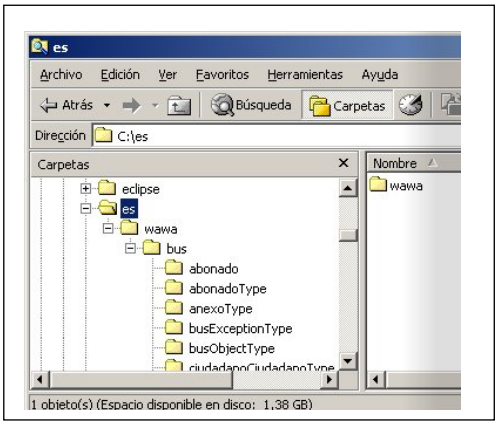
Posteriormente mediante una herramienta adecuada para la tecnología que utilice el cliente, se interpretará el archivo Wsdl, creando los objetos necesarios que permitan realizar una petición al Bus W@nda.

Como ejemplo, en el caso de utilizar el paquete Axis de Apache, existe una herramienta denominada *WSDL2Java* que permite generar un conjunto de clases Java con las cuales se podrá tanto construir los objetos del modelo del W@nda, como invocar los servicios.

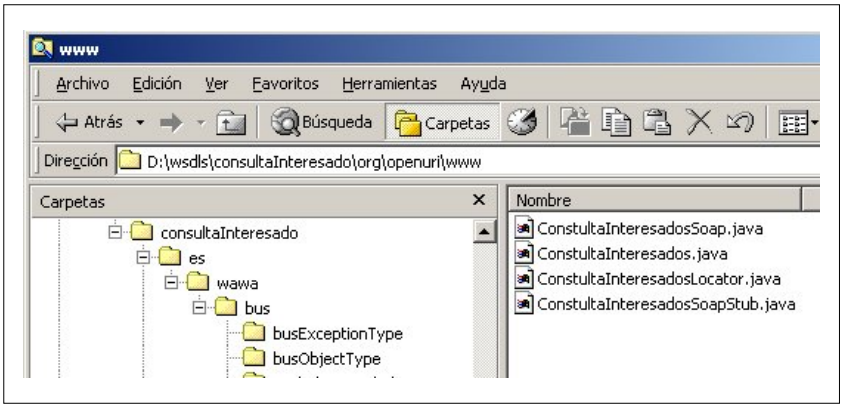
La forma de uso sería la siguiente

```
%>  java org.apache.axis.wsdl.WSDL2Java archivoWS.wsdl
```

Esto generará un directorio con un conjunto de clases correspondiente a los objetos del modelo de datos y otro con el conjunto de clases que permiten invocar a los servicios desde nuestra aplicación cliente en Java.



Conjunto de Clases del Modelo Común de Datos W@nda



Conjunto de Clases Cliente SOAP

4 Estructura de datos E/S

Los archivos de descripción de los webservices (Wsdl) contendrán más objetos de los que se van a utilizar en el servicio, ya que el modelo de datos se ha definido para poder dar cabida a una amplia variaedad de servicios, no solamente el de consulta de datos de la ciudadania.

- ✓ **Descripción de objetos**. En la descripción que se ofrece del servicio se especifica **únicamente**, tanto en entrada como en salida, los campos que deben ser considerados.
- ✓ **Respuesta de error**: En caso de que se produzca error en el proceso la respuesta de cada servicio seguirá devolviendo el mismo objeto, con la particularidad de que contendrá información relativa al error. El contenido de la respuesta se corresponderá con lo descrito en el apartado Control de Errores.

5 Servicio Consulta Datos Ciudadanía

5.1 Descripción del servicio

El servicio de consulta de datos de la ciudadanía, permitirá obtener, a partir de un documento identificativo (DNI, NIF, CIF, etc) los datos de la persona o entidad relacionada, tanto los datos identificativos como los de la dirección de contacto que el usuario, en particular la que se haya introducido al darse de alta en el sistema Cl@ra.

En caso de que el número del documento no se encontrara en el sistema la respuesta se devolverá vacía.

El servicio ofrece un acceso desde la siguiente URL:

http://<host>:<puerto>/agrupadores/tdc/consultaInteresados.jpd

El método a utilizar en el servicio es

WANBUS_ConsultaDatosCiudadano

5.2 Requisitos

Los requisitos principales de las peticiones hacia el servicio de consulta de datos de ciudadano son:

- ✓ El sistema cliente deberá disponer de un certificado de servidor
- ✓ El sistema cliente deberá estar dado de alta previamente en el Bus W@nda. En el alta se proporcionará un usuario y clave de acceso, que deberá incluirse con cada petición al servicio.

5.3 Información de Entrada

En la entrada se solicitará un objeto de tipo **SimpleObjectType** que contenga exclusivamente la información descrita en la siguiente tabla

Nombre	Etiqueta XML	Tipo	Descripción
Número documento	stringValue	Texto(15)	Contendrá el numero identificativo del documento por el que se quiere consultar los datos del ciudadano
Tipo Interesado	descripcion	Texto(1)	Tipo de interesado sobre el que se realiza la consulta ✓ **C =** Ciudadano ✓ **E =** Empresa

5.4 Información de Salida

En salida el servicio devolverá un objeto de tipo **IntereadoMultiple** conteniendo la siguiente información.

Nombre	Etiqueta XML	Tipo	Descripción
interesados El servicio devolverá un array con 0 ó 1 interesados (0 si el número del documento identificativo no se encuentre en el sistema). Si no existieran datos en clara, este objeto aparecerá vacío.			
interesados.interesado[0..1] Contendrá los datos identificativos de un interesado.			
Ciwa del ciudadano	ciwa	Texto(15)	Identificador en TDC del ciudadano relacionado
Tipo Identificador	tipoIdentificador	Texto (30)	Tipo de documento con el que se identifica al interesado (DNI, NIE, Pasaporte….)
Nif / Cif	nifCif	Texto (15)	Numero del documento que se utiliza para la consulta
interesado.interesado.ciudadano Contendrá los datos identificativos propios de la persona en caso de que la consulta se refiera a un ciudadano y cuyo documento identificativo se ha consultado			
Nombre o Razon Social	Nombre	Texto(50)	Nombre en caso de ser un ciudadano o Razón social en caso de ser empresa
Primer Apellido	apellidoCiudadano1	Texto (50)	Apellido 1 del ciudadano (Solo para el caso de ciudadanos)
Segundo Apellido	apellidoCiudadano2	Texto (50)	Apellido 2 del ciudadano (Solo para el caso de ciudadanos)
Fecha Nacimiento	fechaNacimiento	Fecha/Hora	Fecha de nacimiento del ciudadano (Solo para el caso de ciudadanos)
interesados.interesado.ciudadano.datosContacto (Cardinalidad 0:n) Contendrá un listado de campos de datos de contacto. Estos datos de contacto del incluirán dirección, teléfonos, dirección de correo electrónico etc.			
Tipo Vía	tipoVia	Texto (15)	Tipo de Vía : Calle, avenida, etc...
Vía	via	Texto (200)	Nombre de la vía.
Número	numero	Número (5)	Número de la vía
Letra	letra	Texto (2)	Letra de la vía
Piso	piso	Número (2)	Piso de la vivienda

Puerta	puerta	Número (2)	Puerta de la vivienda
Escalera	escalera	Texto (1)	Escalera de la vivienda
Localidad	localidad	Texto (50)	Localidad
Provincia	provincia	Texto (10)	Provincia
País	pais	Texto (30)	País
Código Postal	codigoPostal	Texto (10)	Código Postal
Teléfono Fijo	telefonoFijo	Texto (20)	Teléfono Fijo
Teléfono Móvil	telefonoMovil	Texto (20)	Teléfono Móvil
Email	Email	Texto (50)	Email
Fax	fax	Texto (20)	Fax

Ejemplos de Xml

Ejemplo Xml de Entrada

```
<WANBUS_ConsultaDatosCiudadano xmlns="http://www.openuri.org/"
xmlns:bus="http://wawa.es/bus/busObjectType" xmlns:bus1="http://wawa.es/bus/busExceptionType"
xmlns:com="http://wawa.es/bus/componenteType" xmlns:sim="http://wawa.es/bus/simpleObject"
xmlns:sim1="http://wawa.es/bus/simpleObjectType">
        <sim:simpleObject xmlns:sim="http://wawa.es/bus/simpleObject">
                <sim1:busObject xmlns:sim1="">
                        <bus:componenteDestino xmlns:bus="">
                                <com:idComponente xmlns:com="">xxxxx</com:idComponente>
                                <com:nombreComponente
xmlns:com="">yyyyy</com:nombreComponente>
                        </bus:componenteDestino>
                </sim1:busObject>
                <sim1:stringValue>99999999R</sim1:stringValue>
                <sim1:descripcion>C</sim1:descripcion>
        </sim:simpleObject>

</WANBUS_ConsultaDatosCiudadano>
```

Ejemplo Xml de Salida en caso de que el ciudadano esté dado de alta en Cl@ra

```
<ns:WANBUS_ConsultaDatosCiudadanoResponse xmlns:ns="http://www.openuri.org/">
	<int:interesadoMultiple xmlns:int="http://wawa.es/bus/interesado">
		<int:busObject/>
		<int:interesados>
			<int:interesado>
				<int1:ciwa>1-7FX8F</int1:ciwa>
				<int1:tipoIdentificador>N.I.F.</int1:tipoIdentificador>
				<int1:nifCif>99999999R</int1:nifCif>
				<int1:datosContacto>
					<dat:tipoVia>AV</dat:tipoVia>
					<dat:via>Rep. Argentina</dat:via>
					<dat:numero>27</dat:numero>
					<dat:letra />
					<dat:escalera />
					<dat:piso>0</dat:piso>
					<dat:puerta />
					<dat:telefonoFijo>+34954000111</dat:telefonoFijo>
					<dat:telefonoMovil />
					<dat:fax />
					<dat:email />
					<dat:localidad>41011</dat:localidad>
					<dat:provincia>41</dat:provincia>
					<dat:pais />
					<dat:codigoPostal>41011</dat:codigoPostal>
				</int1:datosContacto>
				<int1:ciudadano>
					<ciud:nombreCiudadano>FERNANDO</ciud:nombreCiudadano>
					<ciud:apellidoCiudadano1>INGLES</ciud:apellidoCiudadano1>
					<ciud:apellidoCiudadano2>INGLES</ciud:apellidoCiudadano2>
				</int1:ciudadano>
				<int1:tipoInteresado/>
			</int:interesado>
		</int:interesados>
	</int:interesadoMultiple>
</ns:WANBUS_ConsultaDatosCiudadanoResponse>
```

Xml de ejemplo en caso de que no existan datos del ciudadano en Cl@ra

```
<ns:WANBUS_ConsultaDatosCiudadanoResponse xmlns:ns="http://www.openuri.org/">
	<int:interesadoMultiple xmlns:int="http://wawa.es/bus/interesado">
		<int:busObject/>
		<int:interesados></int:interesados>
	</int:interesadoMultiple>
</ns:WANBUS_ConsultaDatosCiudadanoResponse>
```

6 Ejemplos de uso

En general, las herramientas de generación de código para Web Services, como es el caso de WSDL2Java, permiten abstraer el código de invocación cliente de la implementación de la lógica de negocio, es decir, a partir de las clases del modelo y de las de invocación, se podrán realizar peticiones SOAP desde un desarrollo abstrayendo la comunicación mediante de Web Services.

La lógica general a seguir será:

1) Construir la clase de invocación
2) Construir el objeto de entrada que requiera el servicio que vamos a utilizar.
3) Invocar al servicio con el objeto construido
4) Comprobar en la respuesta, la posible existencia de errores en el objeto BusObject.

6.1 Ejemplo invocación Consultar Datos Ciudadano

Se muestra a continuación un **ejemplo** de código, para el caso de utilizar tecnología Java con Axis

```
private String miURL = "http://<host>:<puerto>/agrupadores/tdc/constultaInteresados.jpd";
public InteresadoMultiple realizarConsultaCiudadano() throws Exception{
    ConsultaInteresadoSoap consultaInteresadoBus = crearAccesoConsultaInteresado();
     es.wawa.bus.simpleObjectType.SimpleObjectType  datosEntrada = crearObjetoConsulta();
    try{
      CiudadanoType ciudadanoRespuesta= null ;
      es.wawa.bus.interesado.InteresadoMultiple ciudadanoRespuesta =
                                  consultaInteresadosBus.WANBUS_ConsultaDatosCiudadano (datosEntrada);
      BusObjectType b0 = ciudadanoRespuesta.getBusObject();
      // Tras recibir la respuesta es necesario comprobar posibles errores
      // un ejemplo de comprobacion podria ser un metodo del tipo descrito a continuacion
      String codError = checkBusException(b0)
        if(!codError.equals("") ) throw new Exception("Error al realizar la peticion del certificado " + codError);
       }catch(Exception e){
          throw new Exception("Error al enviar la peticion");
       }
     //
   return ciudadanoRespuesta;
 }

private String checkBusException(BusobjectType b0){
      if( b0 != null && b0.getException() != null){
```

```
                    if( Integer.parseInt(b0. getException().getException().getErrorCodeFuncional() >0)
                        return b0. getException().getException().getErrorCodeFuncional();
                    else if (Integer.parseInt( b0. getException().getException().getErrorCodeTecnico())
                        return b0. getException().getException().getErrorCodeTecnico();
                    else if (!b0. getException().getException().getErrorDescTecnico().equals (“”) ||
                            !b0. getException().getException().getErrorDescFuncional().equals (“”)  )
                      return “”;
            }
                  return “”
           }
      private ConsultaInteresadoSoap crearAccesoConsultaInteresado (){
            return new ConsultaInteresadoLocator().getconstultaInteresadosSoap (new URL(miURL));
       }
         private SimpleObjectType crearObjetoConsulta(){
          SimpleObjectType datosEntrada = new SimpleObjectType();
           datosEntrada.setStringValue(“99999999R”)  ;
           datosEntrada.setDescripcion(“C”);
         }
}
static{
 // Bloque estatico para el caso de tener que configurar el SSL
            System.setProperty("javax.net.ssl.trustStore","C:\\j2sdk142_07\\jre\\lib\\security\\cacerts");
            System.setProperty("javax.net.ssl.trustStorePassword", "clave_keystore");
           System.setProperty("javax.net.ssl.keyStore","alias_del_certificado");
           System.setProperty("javax.net.ssl.keyStorePassword","clave_del_certificado");
           System.setProperty("java.protocol.handler.pkgs","com.sun.net.ssl.internal.www.protocol.https");
           System.setProperty("javax.net.ssl.keyStoreType", "pkcs12");
           java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
 }
}
```

7 Control de errores

En cualquier petición de servicio realizada al bus se podrán producir errores, tanto en el propio Bus de Conexión, como en los sistemas finales que le proporcionan la información.

Por tanto, será necesario que el sistema cliente del Bus compruebe la validez de la respuesta que le llega.

Para ello va a ser necesario comprobar el contenido del objeto BusObject que, como ya se ha explicado, es el objeto de control que viajará a través del Bus y mediante el cual vamos a ser capaces de controlar el funcionamiento y estado de los procesos.

En caso de ocurrir una excepción, el servicio devolverá en cada caso el objeto que tenga previamente definido, con la particularidad de que contendrá un objeto ***BusObject***, el cual contendrá un objeto ***exception***.

Como se aprecia en la tabla mostrada a continuación la definición del objeto ***exception*** es compleja, sin embargo los campos que será necesarios para que una aplicación cliente sea capaz de tratar un error devuelto por el Bus serán muchos menos:

1) **Código y descripción funcional del error.** Permitirán a la aplicación cliente determinar el tipo de error funcional que se ha producido, por ej. "*Error en el formato de datos enviados*", con código 1001. En general, **utilizando estos dos campos**, junto con la lista de descripción de errores del Bus W@nda, el sistema cliente debería poder ser capaz de tratar el error devuelto.

Adicionalmente, en casos excepcionales se podrán utilizar estos campos:

2) **Código y descripción técnica del error.** Se corresponderán con los valores generados por la propia excepción que podrá ser tanto del sistema del Bus como de la aplicación final que le da el servicio y servirán para el caso de que el tratamiento del error requiera de datos concretos de tipo técnico, p.ej. "*Error writing XML stream*".

3) **Mensaje de entrada.** Será básicamente el texto del mensaje devuelto por la excepción final

El resto de campos será necesario proporcionarlos en caso de que se produzcan incidencias sobre el funcionamiento del Bus, de forma que el error sea fácilmente traceable.

Nombre	Etiqueta XML	Tipo	Descripción
busObject.exception			
Código Funcional de Error	errorCodeFuncional	Texto (255)	Codigo de Error Funcional de la Excepción
Descripción Funcional de Error	errorDescFuncional	Texto (1000)	Descripción del error Funcional de la excepción
Codigo Técnico de Error	errorCodeTecnico	Texto (255)	Codigo de error técnico de la excepción
Descripción Técnica de Error	errorDescTecnico	Texto (1000)	Descripción del error técnico de la excepción
Mensaje de Entrada	mensajeEntrada	Texto (1000)	Mensaje de entrada del error

8 Convenciones

Los formatos de datos se asume que son los descritos en el Modelo Común de W@nda (ver Bibliografía ref 1) en concreto para

- ✓ Fechas : Los campos de tipo fecha serán String con el formato **dd/mm/aaaa hh:mm:ss**
- ✓ Booleanos: Formatos de valores booleanos "**S**" en mayúscula será verdadero y "**N**" en mayúscula será falso
- ✓ Base64Binary. Se define como un array de bytes codificado en formato binario en base 64.

9 Bibliografía

Referencia	Título	Código
Sistema de Gestión de Proyectos Frawa. Proyecto Wan-014 *WAN014I_ANA_ModeloDatosComun_xxyy.doc*	Modelo de datos Común	1