Web Services PortalTickets

Guía de usuario

Título del documento:	Web Services PortalTickets
Nombre del fichero:	ES WebService PortalTickets Guía de usuario V1.2.odt
Versión:	1.2
Estado:	VIGENTE
Fecha:	06/03/2012
Autor:	Oscar Albert Arcas

Revisión, Aprobación		
Revisado por:	Francisco Belda	Fecha: 27/03/2012
Aprobado por:	Oscar Albert Arcas.	Fecha: 27/03/2012

Historial de cambios			
Versión	Fecha	Descripción de la acción	Páginas
1.0	06/03/2012	Versión inicial WS PortalTickets	Todas
1.1	27/03/2012	Se modifican los códigos de error a tres dígitos. Se añade código de error 131.	17
1.2	26/04/2012	Añadidos requerimientos de uso de las funciones publishDocument y getCustomData.	10, 17

Índice de contenido

1 WEBSERVICE PORTALTICKETS

1.1 INTRODUCCIÓN

PortalTickets es la plataforma webservice de EDICOM para acceder a la funcionalidad de portales de tickets definida por EDICOM. Esta plataforma implementa las funcionalidades más comunes de este tipo de portales para poder acceder a este servicio vía webservice.

Este documento ofrece la WSDL del servicio y la información necesaria para que para interactuar con el servicio PortalTicket y desarrollar aplicaciones compatibles con el servicio de Tickets

1.1.1 Acceso al servicio WebService PortalTickets de Edicom

La URL de acceso al webservice PortalTickets de EDICOM es la siguiente:

https://web.sedeb2b.com/EdiwinWS/services/PortalTickets?wsdl

La implementación del servicio se realiza a través del protocolo https. De esta forma, se utiliza un canal de comunicaciones seguro con el servicio del Portal de Tickets de EDICOM para poder operar de forma segura.

1.1.2 Funciones Del Webservice PortalTickets de Edicom

El Webservice PortalTickets de Edicom permite acceder a la plataforma de gestión del Portal de tickets de Edicom. Las funciones disponibles son:

Función	Descripción	ver.
publishDocument	**Publicación de documentos.** Permite publicar documentos dentro del portal con los datos iniciales del documento.	1.0
getDocument	**Obtención de documentos.** Permite obtener el o los documentos buscados dentro del portal en distintos formatos.	1.0
getCustomData	**Obtención de datos personalizados.** Permite ejecutar funciones personalizadas dentro del portal para obtener los resultados personalizados en función de la función.	1.0

2 FUNCIONES

2.1 PUBLICACIÓN DE DOCUMENTOS.

```
publishDocument
```

Permite publicar documentos en el Portal de Tickets.

Para poder implementar la llamada, el usuario debe autentificarse como usuario del servicio del portal de Tickets y especificar el documento que quiere importar.

2.1.1 LLAMADA:

```
public byte[] publishDocument (String user, String password, String
domain, int publishType, byte[] message, String process, boolean
sendDocument, boolean returnData, String returnDataProcess) throws
PortalTicketsException;
```

2.1.2 PARÁMETROS:

- **user**: Usuario que va realizar la conexión.
- **password**: Contraseña del usuario que va realizar la conexión.
- **domain**: Dominio del portal en el que se va a realizar la conexión.
- **publishType**: Tipo de publicación del documento.
  - 1: Documento de entrada.
  - 2: Documento de salida.
- **message**: Fichero con los datos del documentos a importar. Se trata del propio fichero que se quiere importar al portal o los datos iniciales con los que definitivamente se generará el documento final que se importará en el portal. Este fichero estará comprimido en formato ZIP. Dentro del mensaje SOAP, se enviarán los datos del fichero ZIP codificados en Base64 dentro del sobre de la llamada SOAP.
- **process:** Proceso de transformación que se aplicará al documento de entrada para generar el documento final que se importará.
- **sendDocument:** Indica si el documento hay que enviarlo o no una vez publicado.
- **returnData:** Indica si hay que devolver un fichero como resultado de la ejecución de la publicación. Por defecto, si el parámetro es true, devolverá el listado del documento.

- **returnDataProcess**: En caso de que lo que se quiera devolver sea una transformación del documento original, indicaremos aquí el proceso que hay que ejecutar para hacer la transformación. Se utiliza para especificar el proceso de transformación que hay que hacer para devolver el fichero del ticket en modo texto.

Proceso que contiene el servicio

- Verifica que los datos enviados al servicio son correctos.
- Verifica que los datos de acceso al portal indicado son correctos.
- Si se ha solicitado una transformación de los datos, se publica en entrada el documento original.
- Publica el documento en entrada o salida según especifique el parámetro publishType. Se publica el documento si process es nulo o blanco, y si no, se publica la transformación del documento según ese proceso.
- Si el parámetro sendDocument es true, se deposita el documento.
- Si el parámetro returnData es false, el proceso termina devolviendo una respuesta vacía.
- Si el parámetro returnData es true, y el parámetro returnDataProcess es nulo o blanco, el proceso devolverá el listado del documento importado.
- Si el parámetro returnData es true, y el parámetro returnDataProcess es un proceso, el proceso devolverá la transformación del documento importado según el proceso indicado.

2.1.3 RESPUESTA:

El resultado de la función dependerá de lo que contenga el parámetro returnData que se envía al webservice. En caso de que el parámetro returnData sea false, no se devolverá nada, símplemente una respuesta sin datos. En caso de que el parámetro returnData sea true, devolverá un fichero zip que dentro contendrá el documento especificado. Este documento puede ser o bien un **Listado** del comprobante, o bien el **Ticket** del comprobante.

- **Respuesta satisfactoria**. Una vez acabado todo el proceso, el webservice devolverá una respuesta que podrá estar vacía o contener un fichero en formato formato zip con el resultado de la respuesta seleccionada (nforme o ticket).
- **Respuesta con error**. En caso de que se produzca algún error durante el proceso de publicación, el webservice devolverá una una excepción con información acerca del error que se haya producido. El error tendrá un código de error y un texto genérico del servicio del portal, y a continuación una serie de errores específicos del portal en concreto. Ejemplos de este tipo de excepciones genéricas sería por ejemplo que el usuario no se haya autentificado correctamente en el sistema, o que no se haya enviado el fichero ZIP con el documento.

2.1.4 EJEMPLOS

Código del cliente:

```
public byte[] publishDocument(String user, String password,
String domain, int publishType, byte[] message, String process,
boolean sendDocument, boolean returnData, String
returnDataProcess) {
     byte[] data = null;
     try {
           data = portalTicketsService.publishDocument(user,
password, domain, publishType, message, process, sendDocument,
returnData, returnDataProcess);
     } catch ( PortalTicketsException PTe) {
           PTe.printStackTrace();
           System.out.println("PortalTicketsExcepcion: " +
PTe.getText());
     } catch (RemoteException Re) {
           Re.printStackTrace();
           System.out.println("RemoteExcepcion: " +
Re.toString());
     } catch (Exception e) {
           e.printStackTrace();
           System.out.println("Exception: " + e.toString());
     }
}
```

Petición SOAP:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tic="http://tickets.service.ediwinws.edicom.com">
   <soapenv:Header/>
   <soapenv:Body>
      <tic:publishDocument>
         <tic:user>xxxxxx</tic:user>
         <tic:password>xxxxxx</tic:password>
         <tic:domain>xxxxxx</tic:domain>
         <tic:publishType>2</tic:publishType>
         <tic:message>UEsDBBQAAAAIAOJkmD0CMccqxQcAAFUSAAAfAAAAc2
luX3RpbWJyYXJfX1hN+NAgoBzSJWgV4cbALEKkAC6Q/8Kf6O+2MupJ4ZzWI7/OoY
RaCSKrOyTh5OVeZvfwxFjp2jpk2r8m1CToky4Ti037j1s2nVxDhFECTuyJIRJFHh
ffVN/7vzp7RsO68MogkmV2UUem8T2VEmmJnWFeMVflqhZCZYEDV
[…]
C2Qy59xh/9sq4/ZB2RfEvjSuv3+L1BLAQIUCxQAAAAIAOJkmD0CMccqxQcAAFUSA
AAfAAAAAAAAAAEAIAAAAAAAAABzaW5fdGltYnJhcl9fWE1MSW1wXzQ3MzIwOS54b
Wx+UEsFBgAAAAABAAEATQAAAAIIAAAAAA==</tic:message>
         <tic:process/>
         <tic:sendDocument>false</tic:sendDocument>
```

```
         <tic:returnData>false</tic:returnData>
         <tic:returnDataProcess/>
      </tic:publishDocument>
   </soapenv:Body>
</soapenv:Envelope>
```

Nota*: Los datos que aparecen en este ejemplo no son reales. El código del fichero en base64 ha sido resumido [...]*

Respuesta SOAP:

El siguiente ejemplo es una respuesta de un **envío satisfactorio** sin datos de respuesta.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <publishDocumentResponse
xmlns="http://tickets.service.ediwinws.edicom.com">
         <publishDocumentReturn xsi:nil="true"/>
      </publishDocumentResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

El siguiente ejemplo es una respuesta de un **envío satisfactorio** con datos de respuesta.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <publishDocumentResponse
xmlns="http://tickets.service.ediwinws.edicom.com">
         <publishDocumentReturn>UEsDBBQACAAIABaGZkAAAAAAAAAAAAAA
AAAMAAAAY29tcG9zZXIucGRm3LvZtrrakjd4zxjnHZaCIiAqrYKKIp0tNtj3iCAI
SqOezPNdfK9YF/UGVS9SrP9eZ+99qnJnZdb4rmpdALOJmL+IGREzpiNWbiKrOFGi
/wbk/o//83/73/8G/A0gvipf4eX+N+Dr66vR+Pr1Tv/KUmKb7zD
[...]
dMd8lN8wvT+iPpF/D4wDodI1Y/eIVqBe7j/8eK9yfz2i905Xb5rTAGSBW8Yjj/11
DvkT1BLBwgT+YmIPjsAADVUAABQSwECFAAUAAgACAAWhmZAE/mJiD47AAA1VAAAD
AAAAAAAAAAAAAAAAAAAAAAAY29tcG9zZXIucGRmUEsFBgAAAAABAAEAOgAAAHg7A
AAAAA==</publishDocumentReturn>
      </publishDocumentResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Nota*: Los datos que aparecen en este ejemplo no son reales. El código del fichero en base64 ha sido resumido [...]*

El siguiente ejemplo es una respuesta de un **envío con errores.**

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server.userException</faultcode>
         <faultstring><![CDATA[Errores del documento <Id:114079>
<Referencia:PR1> <Origen:8426616000005> <Destino:8426616900008>
<Fichero:D114079.DOE>~VAL=<<1>>10: LA VERSIÓN DEL COMPROBANTE NO
ES VÁLIDA [3.0]~POS=No existe el Punto Operacional Origen
<8426616000005>~POS=No existe el Punto Operacional Destino
<8426616900008>~|]]></faultstring>
         <detail>
            <ns1:fault
xmlns:ns1="http://tickets.service.ediwinws.edicom.com">
               <ns1:cod>130</ns1:cod>
               <ns1:errors>
                  <ns1:errors>
                     <ns1:cod>100</ns1:cod>
                     <ns1:text>LA VERSIÓN DEL COMPROBANTE NO ES
VÁLIDA [3.0]</ns1:text>
                  </ns1:errors>
               </ns1:errors>
               <ns1:text><![CDATA[Errores del documento
<Id:114079> <Referencia:PR1> <Origen:8426616000005>
<Destino:8426616900008> <Fichero:D114079.DOE>~VAL=<<1>>LA
VERSIÓN DEL COMPROBANTE NO ES VÁLIDA [3.0]~POS=No existe el
Punto Operacional Origen <8426616000005>~POS=No existe el Punto
Operacional Destino <8426616900008>~|]]></ns1:text>
            </ns1:fault>
            <ns2:hostname
xmlns:ns2="http://xml.apache.org/axis/">ed267</ns2:hostname>
         </detail>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

Nota:*En este ejemplo concreto se presenta un error **301 (XML mal formado)** indicando que el documento no es correcto sintácticamente.*

Para ver una lista completa de los posibles códigos de error y respuestas asociadas consultar el punto Lista de Códigos de Error.

2.1.5 REQUERIMIENTOS

Los errores personalizados tienen que devolverse a partir de validadores personalizados que se ejecutarán a partir de funciones de la guía que ejecutarán un validador.

Para que este validador se pueda ejecutar, el interfaz destino ha de ser un XML obligatoriamente. El resultado de la importación de un portal de tickets es un Comprobante, por lo que el fichero final será un XML.

Si se quiere hacer una validación del fichero original antes de hacer la transformación para obtener el comprobante final, el interfaz del documento origen tiene que ser también un XML para poder ejecutar el validador desde las funciones de la guía del interfaz origen. Si se tiene cualquier otro tipo de interfaz intermedio, no se podrá utilizar un validador para el fichero origen y las validaciones se harán exclusivamente sobre el comprobante final.

2.2 OBTENCIÓN DEL DOCUMENTO.

```
getDocument
```

Permite obtener el documento que se especifique con los parámetros especificados en la llamada a la función del webservice.

Para poder implementar la llamada, el usuario debe autentificarse como usuario del servicio del portal de Tickets y especificar el documento que quiere exportar y el tipo de exportación.

2.2.1 LLAMADA:

```
public byte[] getDocument(String user, String password, String domain,
int exportType, Param[] parameters) throws PortalTicketsException;
```

2.2.2 PARÁMETROS:

- **user**: Usuario que va realizar la conexión.
- **password**: Contraseña del usuario que va realizar la conexión.
- **domain**: Dominio del portal en el que se va a realizar la conexión.
- **exportType**: Tipo de exportación del documento.
    - 1: Información extendida
    - 2: Documento
    - 3: Intercambio
    - 4: Informe
- **parameters**: Son un conjunto de datos con los que se hará referencia al documento que se quiere exportar. Consisten en una secuencia de datos con formato campo/valor, a través del cual se harán el filtro de búsqueda de los documentos.

2.2.3 RESPUESTA

El resultado de la función será un fichero comprimido en formato ZIP con el resultado de la exportación del documento especificado por la llamada del webservice.

- **Respuesta satisfactoria:** En caso de que se haya realizado la firma digital, el servicio devolverá un documento comprimido en formato ZIP con el resultado de la exportación del documento.

- **Respuesta con errores:** En caso de que se produzca algún error durante el proceso de obtención del documento, el webservice devolverá una una excepción con información acerca del error que se haya producido. El error tendrá un código genérico de error y un texto explicativo del error producido. Ejemplos de este tipo de excepciones sería por ejemplo que el usuario no se haya autentificado correctamente en el sistema, o que no se haya obtenido ningún documento con los parámetros enviados.

Para ver una lista completa de los posibles códigos de error y respuestas asociadas consultar el punto Lista de Códigos de Error.

2.2.4 EJEMPLOS

Código del cliente:

```
public byte[] getDocument(String user, String password, String
domain, int exportType, Param[] parameters) {
     byte[] data = null;
     try {
          data = portalTicketsService.getDocument(user,
password, domain, exportType, parameters);
     } catch ( PortalTicketsException PTe) {
          PTe.printStackTrace();
          System.out.println("PortalTicketsExcepcion: " +
PTe.getText());
     } catch (RemoteException Re) {
          Re.printStackTrace();
          System.out.println("RemoteExcepcion: " +
Re.toString());
     } catch (Exception e) {
          e.printStackTrace();
          System.out.println("Exception: " + e.toString());
     }
}
```

Petición SOAP:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tic="http://tickets.service.ediwinws.edicom.com">
   <soapenv:Header/>
```

```
   <soapenv:Body>
      <tic:getDocument>
         <tic:user>xxxxx</tic:user>
         <tic:password>xxxxx</tic:password>
         <tic:domain>xxxxx</tic:domain>
         <tic:exportType>2</tic:exportType>
         <!--1 or more repetitions:-->
         <tic:parameters>
            <tic:name>id</tic:name>
            <tic:value>23456</tic:value>
         </tic:parameters>
      </tic:getDocument>
   </soapenv:Body>
</soapenv:Envelope>
```

Respuesta SOAP:

El siguiente ejemplo es una respuesta de un **envío satisfactorio**.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <publishDocumentResponse
xmlns="http://tickets.service.ediwinws.edicom.com">
         <publishDocumentReturn xsi:nil="true"/>
      </publishDocumentResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

El siguiente ejemplo es una respuesta de un **envío satisfactorio** con datos de respuesta.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <getDocumentResponse
xmlns="http://tickets.service.ediwinws.edicom.com">
         <getDocumentReturn>UEsDBBQACAAIABaGZkAAAAAAAAAAAAAAAAAAM
AAAAY29tcG9zZXIucGRm3LvZtrrakjd4zxjnHZaCIiAqrYKKIp0tNtj3iCAISqOe
zPNdfK9YF/UGVS9SrP9eZ+99qnJnZdb4rmpdALOJmL+IGREzpiNWbiKrOFGi/wbk
/o//83/73/8G/A0gvipf4eX+N+Dr66vR+Pr1Tv/KUmKb7zD
[...]
dMd8lN8wvT+iPpF/D4wDodI1Y/eIVqBe7j/8eK9yfz2i905Xb5rTAGSBW8Yjj/11
DvkT1BLBwgT+YmIPjsAADVUAABQSwECFAAUAAgACAAWhmZAE/mJiD47AAA1VAAAD
AAAAAAAAAAAAAAAAAAAAAAAY29tcG9zZXIucGRmUEsFBgAAAAABAAEAOgAAAHg7A
AAAAA==</getDocumentReturn>
```

```
      </getDocumentResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Nota: *Los datos que aparecen en este ejemplo no son reales. El código del fichero en base64 ha sido resumido [...]*

El siguiente ejemplo es una respuesta **con errores.**

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server.userException</faultcode>
         <faultstring>El tipo de exportación 6 no es
válido.</faultstring>
         <detail>
            <ns1:fault
xmlns:ns1="http://tickets.service.ediwinws.edicom.com">
               <ns1:cod>120</ns1:cod>
               <ns1:errors xsi:nil="true"/>
               <ns1:text>El tipo de exportación 6 no es
válido.</ns1:text>
            </ns1:fault>
            <ns2:hostname
xmlns:ns2="http://xml.apache.org/axis/">ed267</ns2:hostname>
         </detail>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

2.3 OBTENCIÓN DE DATOS PERSONALIZADOS.

```
getCustomData
```

Permite obtener datos personalizados de la explotación del portal.

Para poder implementar la llamada, el usuario debe autentificarse como usuario del servicio del portal de Tickets y especificar la función personalizada que se quiere ejecutar junto con los parámetros necesarios para su ejecución.

2.3.1 LLAMADA:

```
public byte[] getCustomData(String user, String password, String
domain, String process, Param[] parameters) throws
PortalTicketsException;
```

2.3.2 Parámetros:

- **user**: Usuario que va realizar la conexión.
- **password**: Contraseña del usuario que va realizar la conexión.
- **domain**: Dominio del portal en el que se va a realizar la conexión.
- **process**: Función personalizada por dominio que se quiere ejecutar. Cada portal tendrá sus funciones específicas.
- **parameters**: Son un conjunto de datos con los que se hará referencia a los parámetros de ejecución de la función. Consisten en una secuencia de datos con formato campo/valor.

2.3.3 Respuesta:

El resultado de la función será un fichero comprimido en formato ZIP con el resultado de la ejecución de la función de la llamada del webservice.

- **Respuesta satisfactoria:** En caso de que se haya realizado la firma digital, el servicio devolverá un documento comprimido en formato ZIP con el resultado de la ejecución de la función.
- **Respuesta con errores:** En caso de que se produzca algún error durante el proceso de ejecución de la función, el webservice devolverá una una excepción con información acerca del error que se haya producido. El error tendrá un código genérico de error y un texto explicativo del error producido. Ejemplos de este tipo de excepciones sería por ejemplo que el usuario no se haya autentificado correctamente en el sistema, o que no se haya obtenido ningún documento con los parámetros enviados.

Para ver una lista completa de los posibles códigos de error y respuestas asociadas consultar el punto Lista de Códigos de Error.

Código del cliente:

```
public byte[] getCustom(String user, String password, String
domain, String process, Param[] parameters) {
     byte[] data = null;
     try {
           data = portalTicketsService.getCustomData(user,
password, domain, process, parameters);
     } catch ( PortalTicketsException PTe) {
           PTe.printStackTrace();
           System.out.println("PortalTicketsExcepcion: " +
PTe.getText());
     } catch (RemoteException Re) {
           Re.printStackTrace();
```

```
            System.out.println("RemoteExcepcion: " +
Re.toString());
      } catch (Exception e) {
            e.printStackTrace();
            System.out.println("Exception: " + e.toString());
      }
}
```

Petición SOAP:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tic="http://tickets.service.ediwinws.edicom.com">
   <soapenv:Header/>
   <soapenv:Body>
      <tic:getCustomData>
         <tic:user>xxxxx</tic:user>
         <tic:password>xxxxx</tic:password>
         <tic:domain>xxxxx</tic:domain>
         <tic:process>process1</tic:process>
         <!--1 or more repetitions:-->
         <tic:parameters>
            <tic:name>id</tic:name>
            <tic:value>23456</tic:value>
         </tic:parameters>
      </tic:getCustomData>
   </soapenv:Body>
</soapenv:Envelope>
```

Respuesta SOAP:

El siguiente ejemplo es una respuesta de un **envío satisfactorio**.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <getCustomDataResponse
xmlns="http://tickets.service.ediwinws.edicom.com">
         <getCustomDataReturn>UEsDBBQACAAIABaGZkAAAAAAAAAAAAAAAAA
AMAAAAY29tcG9zZXIucGRm3LvZtrrakjd4zxjnHZaCIiAqrYKKIp0tNtj3iCAISq
OezPNdfK9YF/UGVS9SrP9eZ+99qnJnZdb4rmpdALOJmL+IGREzpiNWbiKrOFGi/w
bk/o//83/73/8G/A0gvipf4eX+N+Dr66vR+Pr1Tv/KUmKb7zD
[...]
```

```
dMd8lN8wvT+iPpF/D4wDodI1Y/eIVqBe7j/8eK9yfz2i905Xb5rTAGSBW8Yjj/11
DvkT1BLBwgT+YmIPjsAADVUAABQSwECFAAUAAgACAAWhmZAE/mJiD47AAA1VAAAD
AAAAAAAAAAAAAAAAAAAAAAAY29tcG9zZXIucGRmUEsFBgAAAAABAAEAOgAAAHg7A
AAAAA==</getCustomDataReturn>
      </getCustomDataResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Nota: *Los datos que aparecen en este ejemplo no son reales. El código del fichero en base64 ha sido resumido [...]*

El siguiente ejemplo es una respuesta **con errores.**

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server.userException</faultcode>
         <faultstring>El parámetro process no puede esar
vacío.</faultstring>
         <detail>
            <ns1:fault
xmlns:ns1="http://tickets.service.ediwinws.edicom.com">
               <ns1:cod>120</ns1:cod>
               <ns1:errors xsi:nil="true"/>
               <ns1:text>El parámetro process no puede estar
vacío.</ns1:text>
            </ns1:fault>
            <ns2:hostname
xmlns:ns2="http://xml.apache.org/axis/">ed267</ns2:hostname>
         </detail>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

2.3.4 REQUERIMIENTOS

Para poder ejecutar el mapa personalizado, el mapa espera un fichero con una máscara determinada como fichero origen para ejecutar el proceso.

Para unificar criterios y no pasar esta máscara dentro de los datos del webservice, el propio webservice define este fichero origen como un fichero en blanco con el nombre “fichero.txt”.

Por este motivo, todos aquellos mapas personalizados deben tener como origen un interfaz que lea un fichero en blanco con el nombre “fichero.txt”.

3 LISTA DE CÓDIGOS DE ERROR

3.1 INTRODUCCIÓN

El servicio del Portal de Tickets tiene códigos de error genéricos y específicos de cada portal.

3.1.1 CÓDIGOS GENÉRICOS DEFINIDOS POR EDICOM

Estos códigos han sido definidos como códigos de error genéricos por Edicom. Se indican también los posibles escenarios en que se presentan estos errores.

Código	Error
100	**E**rror genérico del servicio del Portal de Tickets.
110	Error de autentificación.
120	Error de datos.
130	Error al publicar documentos**.**
131	Error al publicar documento origen**.**
140	Error al exportar documentos**.**
150	Error al obtener datos personalizados.

3.1.2 CÓDIGOS ESPECÍFICOS PARA EL PORTAL**.**

Cada portal puede definir ciertos tipos de códigos de errores y aplicarlos en las distintas ejecuciones de los procesos. De esta forma, cada porta puede configurarse sus propios códigos de error específicos.

3.1.3 EJEMPLO DE UNA RESPUESTA DE ERROR

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server.userException</faultcode>
         <faultstring>El parámetro process no puede esar
vacío.</faultstring>
         <detail>
            <ns1:fault
xmlns:ns1="http://tickets.service.ediwinws.edicom.com">
               <ns1:cod>120</ns1:cod>
               <ns1:errors xsi:nil="true"/>
               <ns1:text>El parámetro process no puede estar
vacío.</ns1:text>
```

```
            </ns1:fault>
            <ns2:hostname
xmlns:ns2="http://xml.apache.org/axis/">ed267</ns2:hostname>
         </detail>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

3.2 WSDL WEBSERVICE PORTALTICKET

El contenido del wsdl del servicio es el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
targetNamespace="http://tickets.service.ediwinws.edicom.com"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://tickets.service.ediwinws.edicom.com"
xmlns:intf="http://tickets.service.ediwinws.edicom.com"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <wsdl:types>
  <schema elementFormDefault="qualified"
targetNamespace="http://tickets.service.ediwinws.edicom.com"
xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="getDocument">
    <complexType>
     <sequence>
      <element name="user" type="xsd:string"/>
      <element name="password" type="xsd:string"/>
      <element name="domain" type="xsd:string"/>
      <element name="exportType" type="xsd:int"/>
      <element maxOccurs="unbounded" name="parameters"
type="impl:Param"/>
     </sequence>
    </complexType>
   </element>
   <complexType name="Param">
    <sequence>
     <element name="name" nillable="true" type="xsd:string"/>
     <element name="value" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <element name="getDocumentResponse">
    <complexType>
     <sequence>
      <element name="getDocumentReturn"
type="xsd:base64Binary"/>
     </sequence>
    </complexType>
```

```
   </element>
   <complexType name="Error">
    <sequence>
     <element name="cod" type="xsd:int"/>
     <element name="text" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <complexType name="ArrayOfError">
    <sequence>
     <element maxOccurs="unbounded" minOccurs="0" name="item"
type="impl:Error"/>
    </sequence>
   </complexType>
   <complexType name="PortalTicketsException">
    <sequence>
     <element name="cod" type="xsd:int"/>
     <element name="errors" nillable="true"
type="impl:ArrayOfError"/>
     <element name="text" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <element name="fault" type="impl:PortalTicketsException"/>
   <element name="publishDocument">
    <complexType>
     <sequence>
      <element name="user" type="xsd:string"/>
      <element name="password" type="xsd:string"/>
      <element name="domain" type="xsd:string"/>
      <element name="publishType" type="xsd:int"/>
      <element name="message" type="xsd:base64Binary"/>
      <element name="process" type="xsd:string"/>
      <element name="sendDocument" type="xsd:boolean"/>
      <element name="returnData" type="xsd:boolean"/>
      <element name="returnDataProcess" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="publishDocumentResponse">
    <complexType>
     <sequence>
      <element name="publishDocumentReturn"
type="xsd:base64Binary"/>
     </sequence>
    </complexType>
   </element>
   <element name="getCustomData">
    <complexType>
     <sequence>
```

```
      <element name="user" type="xsd:string"/>
      <element name="password" type="xsd:string"/>
      <element name="domain" type="xsd:string"/>
      <element name="process" type="xsd:string"/>
      <element maxOccurs="unbounded" name="parameters"
type="impl:Param"/>
     </sequence>
    </complexType>
   </element>
   <element name="getCustomDataResponse">
    <complexType>
     <sequence>
      <element name="getCustomDataReturn"
type="xsd:base64Binary"/>
     </sequence>
    </complexType>
   </element>
  </schema>
 </wsdl:types>
   <wsdl:message name="PortalTicketsException">
      <wsdl:part element="impl:fault" name="fault">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="getCustomDataRequest">
      <wsdl:part element="impl:getCustomData" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="publishDocumentRequest">
      <wsdl:part element="impl:publishDocument"
name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="publishDocumentResponse">
      <wsdl:part element="impl:publishDocumentResponse"
name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="getDocumentRequest">
      <wsdl:part element="impl:getDocument" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="getCustomDataResponse">
      <wsdl:part element="impl:getCustomDataResponse"
name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="getDocumentResponse">
      <wsdl:part element="impl:getDocumentResponse"
name="parameters">
```

```
      </wsdl:part>
   </wsdl:message>
   <wsdl:portType name="PortalTickets">
      <wsdl:operation name="getDocument">
         <wsdl:input message="impl:getDocumentRequest"
name="getDocumentRequest">
       </wsdl:input>
         <wsdl:output message="impl:getDocumentResponse"
name="getDocumentResponse">
       </wsdl:output>
         <wsdl:fault message="impl:PortalTicketsException"
name="PortalTicketsException">
       </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="publishDocument">
         <wsdl:input message="impl:publishDocumentRequest"
name="publishDocumentRequest">
       </wsdl:input>
         <wsdl:output message="impl:publishDocumentResponse"
name="publishDocumentResponse">
       </wsdl:output>
         <wsdl:fault message="impl:PortalTicketsException"
name="PortalTicketsException">
       </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="getCustomData">
         <wsdl:input message="impl:getCustomDataRequest"
name="getCustomDataRequest">
       </wsdl:input>
         <wsdl:output message="impl:getCustomDataResponse"
name="getCustomDataResponse">
       </wsdl:output>
         <wsdl:fault message="impl:PortalTicketsException"
name="PortalTicketsException">
       </wsdl:fault>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="PortalTicketsSoapBinding"
type="impl:PortalTickets">
      <wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getDocument">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getDocumentRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="getDocumentResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
         <wsdl:fault name="PortalTicketsException">
```

```
            <wsdlsoap:fault name="PortalTicketsException"
use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="publishDocument">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="publishDocumentRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="publishDocumentResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
         <wsdl:fault name="PortalTicketsException">
            <wsdlsoap:fault name="PortalTicketsException"
use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="getCustomData">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getCustomDataRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="getCustomDataResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
         <wsdl:fault name="PortalTicketsException">
            <wsdlsoap:fault name="PortalTicketsException"
use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="PortalTicketsService">
      <wsdl:port binding="impl:PortalTicketsSoapBinding"
name="PortalTickets">
         <wsdlsoap:address
location="https://web.sedeb2b.com/EdiwinWS/services/PortalTicket
s"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Para acceder a la última versión de la WSDL a través de internet:

https://web.sedeb2b.com/EdiwinWS/services/PortalTickets?wsdl