

Contenido

1	Control de cambios	2
2	Prefacio	2
2.1	Acerca de esta guía	2
2.2	Audiencia.....	2
2.3	Feedback para esta documentación	2
3	Transacción de anulación.....	3
3.1	Descripción de la anulación.....	3
3.2	Descripción del método del servicio web de anulación de transacciones.....	4
3.2.1	Operación nullify	4
3.2.2	Códigos de error.....	5
4	Anexo C: Ejemplos de integración con API SOAP Webpay	6
4.1	Ejemplo Java.....	7

1 Control de cambios

Fecha	Version	Descripción del cambio
12-12-12	1.0	Liberación inicial de la API de integración con WS anulaciones. Contiene ejemplos de integración en JAVA Futuros Release: <ul style="list-style-type: none">Ejemplos de integración PHP, .NET.

2 Prefacio

2.1 Acerca de esta guía

Esta guía describe los aspectos técnicos que deben ser considerados en la integración con Webpay utilizando API SOAP, describe la operación del servicio Web para Anulación de transacciones y cómo debe ser utilizado. Se incluye ejemplo Java.

2.2 Audiencia

Esta guía esta dirigida a implementadores que realizan la integración de Webpay en comercios utilizando la API SOAP para soportar en estos la anulación de transacciones Webpay.

Se recomienda que quién realice la integración posea conocimiento técnico de al menos en los siguientes temas:

- Servicios Web
- WS-Security
- Firma digital, generación y validación.

2.3 Feedback para esta documentación

Ayúdanos a mejorar esta información enviándonos comentarios a soporte@webpay.cl


3 Transacción de anulación

3.1 Descripción de la anulación

Este método permite a todo comercio habilitado anular una transacción que fue generada en plataforma Webpay 3G. El método contempla anular total o parcialmente una transacción. Para ello se deberá indicar los datos asociados a la transacción de venta en línea que se desea anular y los montos requeridos para anular. Se pueden realizar tantas anulaciones como saldo tenga disponible hasta el monto total autorizado. Se considera totalmente anulada una transacción cuando el monto anulado o el monto total de anulaciones cursadas alcancen el monto autorizado en la venta en línea.

Las ejecuciones con errores entregarán un SoapFault de acuerdo a la codificación de errores definida.

Resumen del método del servicio web de anulación de transacciones

Método	Descripción general
Nullify	<p>Permite solicitar a Webpay la anulación de una transacción realizada previamente y que se encuentra vigente.</p> <div><p>El método nullify debe ser invocado siempre indicando el código del comercio que realizó la transacción. En el caso de comercios MALL, el código debe ser el código de la tienda virtual.</p></div>

3.2 Descripción del método del servicio web de anulación de transacciones

3.2.1 Operación nullify

Método que permite anular una transacción de pago Webpay.

Parámetros de entrada

Nombre	Descripción
authorizationCode	<p><code>xs:string</code></p> <p>Código de autorización de la transacción que se requiere anular. Para el caso que se esté anulando una transacción de captura en línea, este código corresponde al código de autorización de la captura.</p> <p>Largo máximo: 6</p>
authorizedAmount	<p><code>xs:decimal</code></p> <p>Monto autorizado de la transacción que se requiere anular. Para el caso que se esté anulando una transacción de captura en línea, este monto corresponde al monto de la captura.</p> <p>Largo máximo: 10</p>
buyOrder	<p><code>xs:string</code></p> <p>Orden de compra de la transacción que se requiere anular</p> <p>Largo máximo: 26</p>
commercelid	<p><code>xs:long</code></p> <p>Código de comercio o tienda mall que realizó la transacción</p> <p>Largo: 12</p>
nullifyAmount	<p><code>xs:decimal</code></p> <p>Monto que se desea anular de la transacción</p> <p>Largo máximo:10</p>

Parámetros de salida

Campo	Descripción
Token	<code>xs:string</code> Token de la transacción.
authorizationCode	<code>xs:string</code> Código de autorización de la anulación
authorizationDate	<code>xs:dateTime</code> Fecha y hora de la autorización
Balance	<code>xs:decimal</code> Saldo actualizado disponible para anular
nullifiedAmount	<code>xs:decimal</code> Monto anulado

3.2.2 Códigos de error

Código	Descripción
304	Validación de campos de entrada nulos
245	Código de comercio no existe
22	El comercio no se encuentra activo
316	El comercio indicado no corresponde al certificado o no es hijo del comercio MALL en caso de transacciones MALL
308	Operación no permitida
274	Transacción no encontrada
16	La transacción no permite anulación
292	La transacción no está autorizada
284	Periodo de anulación excedido
310	Transacción anulada previamente
311	Monto a anular excede el saldo disponible para anular
315	Error del autorizador

4 Anexo C: Ejemplos de integración con API SOAP Webpay

Los siguientes ejemplos tienen por objetivo exponer una forma factible de integración con API SOAP Webpay para resolver los siguientes puntos asociados a la integración:

1. Generación de cliente o herramienta para consumir los servicios Web, lo cual permite abstraerse de la complejidad de mensajería SOAP asociada a los Webservice y hacer uso de las operaciones del servicio.
2. Firma del mensaje y validación de firma en la respuesta, existen frameworks y herramientas asociadas a cada lenguaje de programación que implementan el estándar WS Security, lo que se requiere es utilizar una de éstas, configurarla y que realice el proceso de firma digital del mensaje.

4.1 Ejemplo Java

Este ejemplo hará uso de los siguientes frameworks para consumir los servicios Web de Webpay utilizando WS Security:

- **Apache CXF**, es un framework open source que ayuda a construir y consumir servicios Web en Java. En este ejemplo se utilizará para:
 - Generar el cliente del Webservice o STUBS.
 - Consumir los servicios Web
- **Apache WSS4J**, proporciona la implementación del estándar WS Security, nos permitirá:
 - Firmar los mensajes SOAP antes de enviarlo a Webpay.
 - Validar la firma de la respuesta del servicio Web de Webpay.
- **Spring framework 3.0**, permite que CXF y WSS4J trabajen en conjunto, también se utiliza para configurar WS Security en la firma del mensaje SOAP.

Pasos a seguir:

1. Generación de cliente del Webservice.

Para generar el código Java que implementará el cliente SOAP se utilizará `wsdl2java` de CXF, el cual toma el WSDL del servicio y genera todas las clases necesarias para invocar el servicio Web. Más información en <http://cxf.apache.org/docs/wsdl-to-java.html>

```
wsdl2java -autoNameResolution <URL del wsdl>
```


2. Configuración de WS Security

Para configurar WS Security en CXF se deben habilitar y configurar los interceptores que realicen el trabajo de firmado del mensaje. La configuración de los interceptores se puede realizar a través de la API de servicios Web o a través del XML de configuración de Spring, en este caso se realizará a través de Spring en el archivo applicationContext.xml de la aplicación.

Se deben habilitar y configurar 2 interceptores, uno para realizar la firma de los mensajes enviados al invocar una operación del servicio Web de Webpay y otro para validar la firma de la respuesta del servicio Web.

Interceptor de salida

```
<bean class="org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor"
      id="signOutRequestInterceptor">
  <constructor-arg>
    <map>
      <entry key="signaturePropFile" value="signatureOut.properties"/>
      <entry key="user" value="${alias.client}"/>
      <entry key="action" value="Signature"/>
      <entry key="passwordCallbackClass"
            value="com.transbank.webpay.wsse.ClientCallBack"/>
      <entry key="signatureParts"
            value="{Element}{http://schemas.xmlsoap.org/soap/envelope/}Body"/>
    </map>
  </constructor-arg>
</bean>
```

Interceptor de entrada

```
<bean class="org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor"
      id="signInRequestInterceptor">
  <constructor-arg>
    <map>
      <entry key="action" value="Signature" />
      <entry key="signaturePropFile" value="signatureIn.properties"/>
      <entry key="passwordCallbackClass"
            value="com.transbank.webpay.wsse.ServerCallBack"/>
      <entry key="signatureParts"
            value="{Element}{http://schemas.xmlsoap.org/soap/envelope/}Body"/>
    </map>
  </constructor-arg>
</bean>
```

Las clases ClientCallBack y ServerCallBack implementan la interfaz javax.security.auth.callback.CallbackHandler, su implementación permite al framework de seguridad recuperar la contraseña para acceder al almacén de llaves de la aplicación (Java Key Store) que almacena los certificados digitales.

3. Llamada a operaciones del Webservice

Para realizar la llamada al método del servicio web, se debe importar las clases generadas por el framework que contienen objetos para los parámetros de entrada y salida, además de las interfaces del servicio a consumir. Se debe considerar que la fecha debe ser pasada en formato XMLGregorianCalendar o equivalente dependiendo del lenguaje de implementación.

Para el fragmento utilizado de ejemplo a continuación se utiliza java.util.GregorianCalendar que luego se parsea a XMLGregorianCalendar.

```
import com.transbank.webpay.wswebpay.service.NullificationInput;
import com.transbank.webpay.wswebpay.service.NullificationOutput;
import com.transbank.webpay.wswebpay.service.WSCommerceIntegrationService;
import com.transbank.webpay.wswebpay.service.WSCommerceIntegrationServiceImplService;
.....
import java.util.GregorianCalendar;
import javax.xml.datatype.XMLGregorianCalendar;
.....
WSCommerceIntegrationServiceImplService client = new
WSCommerceIntegrationServiceImplService();
WSCommerceIntegrationService service = client.getWSCommerceIntegrationServiceImplPort();

NullificationInput input = new NullificationInput();
XMLGregorianCalendar TrxDate = getFecha("2012-10-23 21:14:23");
input.setAuthorizationDate(TrxDate);
input.setAuthorizationCode("188051");
input.setAuthorizedAmount(new BigDecimal("54556"));
input.setBuyOrder("344343434");
input.setCommerceId(4445556667L);
input.setNullifyAmount(new BigDecimal("23200"));

NullificationOutput output = service.nullify(input);
.....
private static XMLGregorianCalendar getFecha(String s) throws ParseException,
DatatypeConfigurationException {
    GregorianCalendar cal = new GregorianCalendar();
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Date date = sdf.parse(s);
    cal.setTime(date);
    XMLGregorianCalendar fecha =
DatatypeFactory.newInstance().newXMLGregorianCalendar(cal);
    return fecha;
}
.....
```

URL:

<http://cxf.apache.org/docs/ws-security.html>

<http://ws.apache.org/wss4j/>

<http://cxf.apache.org/docs/wsdli-to-java.html>