Valid from 1 January 2013

eBREV Webb
– Technical manual

Generally
The most recent version of this manual is available on stralfors.se.

Here's how you send eBREV Webb via Web Service
1. You deliver a zip archive to Stralfors
2. We check the file structure and warn about any errors which are not consistent with the printout date.
3. The file is distributed between the various printout centres where it is printed out, enveloped and submitted for distribution.
4. A waybill and invoice are sent to you.

Test routines
The implementation of Web Services must always be tested.

Description of test
Stralfors and your test officer draw up a timetable and test scenarios.
Tests are best conducted in stages.
- log-in and communication test environment for Web Service/PWS
- uploading of the zip archive and verification of included files.
- check of layout, typography and quality of letter.

The test address file shall contain several fictitious recipients.

Description of eBREV Webb Services

Communication overview
TFS/PWS publish Web Services in order to facilitate communication *vis-à-vis* the eBREV Webb service. PWS facilitates several methods for sending and retrieving files, adapted to different technical conditions and security requirements.

Concepts and terms

Generally regarding Web Services and SOAP
Web Services are a technique for systems integration with loose connections. Web Services utilises, among other things, SOAP (Simple Object Access Protocol) which comprises a programming model for distributed systems. The support in the development tool makes use of SOAP very easy.

SOAP involves questions/response-calls being sent as text messages on XML format. Each message consists of Envelope, Header and Body as well as any Attachments which are unstructured data which can be attached to a message. The transport protocol in most cases is HTTP(S), which is firewall-friendly and contains an established security model.

Envelope
Envelope is the top level in the SOAP message and contains a Header, a Body as well as one or more possible attachments.

Header
The Header can contain designated parameters which are not included as data in the actual message. Examples of header parameters can be information for authentication, transaction support or references.

Body
Body contains information as to which method shall be ordered and the arguments therefor. Complex data types can be used provided that there are mechanisms in place to serialise/de-serialise them as XML.

File structure of zip archive

In order to submit data to eBREV Webb on the Web Service, a compacted file archive is sent in zip format. The zip archive shall contain three or two files:

- letter file (PDF or Word doc)
- configuration file (cfg) with or without address list.
- address list (txt) if the addresses are not found in the configuration file.

The zip file must be compressed in such a way that it can be opened with WinZip. It may not be protected by passwords or similar.

Letter file

The letter file shall be in Microsoft Word or PDF format. For senders, especially of PDFs, there are a number of restrictions and rules. These may be found at posten.se under the Help section for eBREV Webb. Word documents may only contain images (gif, tiff, jpg, bmp), word graphics and text.

Address list

The list of recipients can be delivered in two different ways:
The address file must be a tab-delimited text file named with the suffix ".txt", e.g.
• As a part of the cfg file (preferable, see next section).
• As a separate tab-delimited text file.

Whichever method is used, the following rules apply:

- The address file must be a tab-delimited text file named with the suffix ".txt", e.g. "addresses.txt".
- The file must be saved in ANSI/ISO-8550-1 format. UTF 8 is not approved.
- The letter *Â* may not be used in address lists.
- Only one recipient is filled in per row
- Max 41 characters in each field
- Max 5,000 addressees.

There must be following tab-delineated fields:
Name Address2 Address3 address4 Zip city country
Name Address Line1 Address Line2 Address Line3 Postal Code Postal District Country Code

Field name	Max number characters	Explanation
Name (recipient)	Max 41 characters	Mandatory value
Address Line1	Max 41 characters	
Address Line2	Max 41 characters	
Address Line3	Max 41 characters	
Postal code	Max 12 characters	Mandatory value
Postal district	Max 41 characters	Mandatory value
Country code	Max 2 characters	Mandatory value, SE for Sweden

For paying-in slips, supplement with:

Pg/Bg: PG = post giro, BG = bank giro

		Mandatory value
Giro number	Depending on type, max. 20 characters	Mandatory value
Payee	max. 30 characters	Mandatory value
Amount in kronor, ören		Mandatory value
OCR number		
text1-text6	max. 30 characters	Free text field that appears in message field

Contact eBREV Webb Support for address file templates.

Foreign shipments

A correct country code must be quoted when sending letters to foreign recipients.
The country code must follow the ISO 3166 standard.
Zip and City are mandatory values, whichever country the shipment is bound for, address fields 1–3 are voluntary and can contain max. 41 characters.
Billing in accordance with current postage table.

Configuration file eBREV-WebbD.cfg

The configuration file is unique for eBREV Webb services. The XML file contains data such as sender, recipients, assignment type, etc. The file has an XML structure and the character code shall be UTF-8.

Definitions for XML tags

<SenderData>

Tag	Description
<UserID>	the value in the UserID field must be the one selected in conjunction with registration
<BPN>	the value allocated to you in accordance with the welcome letter. It is a numerical value comprising ten positions including two initial zeros (e.g. 002222333344).
<Email>	Contact address used in connection with any problems.

<PriceInfo> alt <ProductOptions>

Tag	Description
<ColorType>	“1” for colour, “0” for black white printout
<DeliveryType>	“A” for 1st class letter “B” for second-class letter.
<DocumentId>	1 = With continuation sheet (i.e. recipient and sender written on a separate sheet) 2 = Giro (only paying-in slip and only via Posten.se) 3 = Giro (information contained in attached address file) 4 = Without endpaper

<SenderAddress>

The sender’s address in letters to which undeliverable post is returned. Max. 41 characters per address row.

Tag	Description
<addressLine1>	Sender’s name
<addressLine2>	Address line 2 (street)
<addressLine3>	Address line 3
<addressLine4>	Address line 4
<Zipcode>	Max. 8 characters
<Country>	Only SE as sender

<Attachments>

There are three attachment types: PDF, DOC and Address List.

PDF and DOC indicate the letter file and Address List indicates a tab-separated address file (when used). If you want to specify recipients in the cfg file, this section is not needed.

Tag	Description
<Type>	File type. DOC, PDF or Address List
<Name>	The file's name
<Email>	Contact address used in connection with any problems.

Ex.

```
<Attachments>
<Type>PDF</Type>
<Name>pdf.pdf</Name>
</Attachments>
<Attachments>
<Type>Adresslista</Type>
<Name>listan.txt</Name>
</Attachments>
```

Recipients

This array contains recipient addresses. If this array is contained in the cfg file, no separate address file is needed. The content and limitations of the fields are described in the section entitled *Address List.*

```
<Recipients>
<Letter>
<File>DOCUMENT.PDF</File>
<ZipCode>806 36</ZipCode>
<City>GÄVLE</City>
<Country>SE</Country>
<RecipientName>Gunilla Åberg</RecipientName>
<AddressLine1/>
<AddressLine2>Ängsullsvägen 6</AddressLine2>
<AddressLine3/>
</Letter>
</Recipients>
```

Med giro info skall även nedanstående ingå I Recipient taggen:

```
<GiroInfo>
<GiroType>PG</GiroType>
<GiroNumber>12345</GiroNumber>
<Amount>100,20</Amount>
<SenderName>robert</SenderName>
<OCR>848484848488</OCR>
<RefText1>Detta är en fritext</RefText1>
<RefText2/>
<RefText3/>
<RefText4/>
<RefText5/>
<RefText6/>
</GiroInfo>
```

API

Authentication

For HTTP Basic, user and password are stated as a base64-coded HTTP-header in accordance with RFC-1945.

Exceptions and error codes

The sendWithAddressing and send method throws exceptions when the shipment is not approved or when there are operational problems; these errors must be dealt with by the client application.

There is a description of the error codes below under the heading “Exceptions and error codes”.

Return values

The return value (label) returned from the “send” methods must always be saved by the customer. It should be linked to the data you sent to Posten in a database or similar.
This label is used by the Posten service desk to refer to your shipment.
This label must also be used in the event of a complaint.

Methods

sendWithAddressing

```
String sendWithAddressing( byte[] data,
String tfsMsgType,
String tfsSender,
String tfsReceiver )
```

Return value:

String containing status code and transaction reference (e.g. 200 PWS.xxxx.xxxx.xxxx).

Description: Send data as an argument in the form of a memory buffer as well as addressing parameters. tfsMsgType and tfsReceiverare normally “ebrevwebb”, tfsSender is a customer-specfic ID id.

Ping

```
String ping()
```

Description

Checks the status of the web service. For testing only.
Be careful not to call this method between each call to sendxxx, as this extends the process time for your submissions.

Return value

One string. “pong”.

Exceptions and error codes

When PWS does not accept an item of mail it generates an “exception”. Example of printout of an error message from an exception: Errorcode: (300) Invalid zip. Too few files, found: 2, expected: 3
The error message contains an error code as described below.

Code	Description
200	Ok
300	Non-approved zip file
301	Non-approved cfg file
302	Configuration file missing
303	Incorrect document
304	Document (doc/pdf) missing
305	Address list missing
306	Invalid address list
307	Conversion error, contact support
308	Communications error, contact support
309	Incorrect character set. The address list must be ANSI (ISO-8859-1). Config shall be UTF-8
310	XML Parsing unsuccessful, incorrect config file
312	Document designated in CFG does not correspond with the file's name
313	Zip file is too large (max. 3MB)
314	The document is too large (max. 3MB)
315	SOAP header missing (only if you use “send”). Switch to “sendToTfs” method instead.
316	Conversion error, contact support
317	Too many recipients in address file (max. is 10,000)
318	Invalid PDF. Most commonly occurring when the document is created by Ghostscript in combination with TTF fonts. Check which document created the PDF and that the fonts which are used are valid. Fonts which have a name of type xxxx+TTE are not valid.
319	Incorrect number of pages in the document. Min. 1, max. 6 (12 for second class letters/)

Customer Support

See contact details at posten.se

FAQs

When can you send in your letters?

The service is open around the clock, except for service.

Who do you contact if you have communication problems?

See contact details at posten.se

Which SOAP standards are used?

PWS supports the unofficial W3C standards:
SOAP 1.1, http://www.w3.org/TR/soap
WSDL 1.1, http://www.w3.org/TR/wsdl

Which implementations of SOAP are compatible with PWS?

PWS is based on Apache Axis 1.1. compatibility with other SOAP implementations as per various interoperability tests, e.g. http://www.whitemesa.com/interop.htm.
In order to attempt to maintain compatibility, as far as possible complex data types have been avoided in parameters and return values.

Example

Configuration file with separate address file

NB. Character coding in UTF-8

Filename=eBREV-WebbD.cfg

Example of independent address list:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\xml\config.xsd">
<SenderData>
<UserId>xxxx</UserId>
<CustomerId>yyyy</CustomerId>
<BPN>1234567</BPN>
<DocumentId>4</DocumentId>
</SenderData>
<PriceInfo>
<ColorType>0</ColorType>
<DeliveryType>B</DeliveryType>
</PriceInfo>
<senderAddress>
<addressLine1>Företaget AB</addressLine1>
<addressLine2>Box 123</addressLine2>
<addressLine3></addressLine3>
<addressLine4></addressLine4>
<zipCode>12345</zipCode>
<city>Stockholm</city>
<country>SE</country>
</senderAddress>
<Attachments>
<Type>DOC</Type>
<Name>doc.doc</Name>
</Attachments>
<Attachments>
<Type>Adresslista</Type>
<Name>adresses.txt</Name>
</Attachments>
</Config>
```

Configuration file with address list and giro information

This kind of configuration is preferable to one where the recipients' addresses are delivered in a separate text file.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Config>
<SenderData>
<BPN>0020000000</BPN>
<UserId>myuser</UserId>
<Email>kale@mycompany.com</Email>
</SenderData>
<ProductOptions>
<ColorType>0</ColorType>
<DeliveryType>B</DeliveryType>
<DocumentId>3</DocumentId>
</ProductOptions>
<SenderAddress>
<AddressLine1></AddressLine1>
<AddressLine2></AddressLine2>
<AddressLine3></AddressLine3>
<AddressLine4></AddressLine4>
<ZipCode>12345</ZipCode>
<City>Mycity</City>
<Country>SE</Country>
</SenderAddress>
<Recipients>
<Letter>
<File>DOCUMENT.PDF</File>
<ZipCode>75435</ZipCode>
<City>Uppsala</City>
<Country>SE</Country>
<RecipientName>Kalle Kanin</RecipientName>
<AddressLine1>Gatan 123</AddressLine1>
<AddressLine2 />



<AddressLine3 />
<GiroInfo>
<GiroType>PG</GiroType>
<GiroNumber>123-123</GiroNumber>
<Amount>100,01</Amount>
<PaymentReceiver>My Company</PaymentReceiver>
<OCR>848484848488</OCR>
<RefText1>Fritext</RefText1>
<RefText2></RefText2>
<RefText3></RefText3>
<RefText4></RefText4>
<RefText5></RefText5>
<RefText6></RefText6>
</GiroInfo>
</Letter>
</Recipients>
</Config>
```

Code example

The code examples that call the PWS service in this section are written in Java (using the Axis product from Apache; http://ws.apache.org/axis/) and in C# (MS Visual Studio .NET). The examples must be supplemented with customer-specific parameters.

Please note that the test environment and the production environment use different URLs and login information.

Note that these examples are not complete and should only be used as guidelines.

Transmission with Java (>=1.5)

```
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import javax.xml.namespace.QName;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
public class Pws2Sample {
public final String NAMESPACE = “http://pws.posten.se”;
public final int BUFSIZE = 8152;
/**
* Endpoint make sure it is correct!
*/
public final String ENDPOINT = “http://localhost:9081/pws/services/PWS”;
/**
* Authentication. Change here
*/
public String username = “YOUR_USER”;
public String password = “YOUR_PASSWORD”;
public byte[] getByteBuffer( InputStream is ) throws IOException {
ByteArrayOutputStream os = new ByteArrayOutputStream();
byte[] buf = new byte[BUFSIZE];
int len;
while( true ) {
len = is.read( buf, 0, buf.length );
if( len < 0 )
break;
os.write( buf, 0, len );
}
return os.toByteArray();
}
public void go(String fileName){
try {
InputStream is = new FileInputStream( fileName );
// Replace sender with your own user
String tfsMsgType = “EBREVWEBB”;
String tfsSender = “Company X”;
String tfsReceiver = “EBREVWEBB”;
Service service = new Service();
```



```
Call call = (Call) service.createCall();
call.setTargetEndpointAddress( new URL( ENDPOINT ) );
call.setOperationName( new QName( NAMESPACE, “sendWithAddressing” ) );
call.setUsername( username );
call.setPassword( password );
String result = (String) call.invoke( new Object[] {getByteBuffer(is), tfsMsgType,
tfsSender, tfsReceiver} );
// The result contains 200 + mark (e.g. “200 SOAP.xxx.xxx”)
```

```
System.out.println(result);
}
catch( Exception e ) {
// Message contains error code and description
// Consult your manual for more information.
e.getMessage();
}
}
public static void main( String[] args ) {
Pws2Sample sample = new Pws2Sample();
sample.go(args[0]);
}
}
```

Transmission with C# (MS Visual Studio .NET 3.5 WCF)

Code fragment. The code uses a proxy class (`SendClient`) generated by Visual Studio

```
// Avoids the first (505) error
System.Net.ServicePointManager.Expect100Continue = false;
// Load the zip as a binary
string filename = “c:/mytestfile.zip”
System.IO.FileStream fs = File.OpenRead(filename);
byte[] data = new byte[fs.Length];
fs.Read(data, 0, data.Length);
EndpointAddress ea = new EndpointAddress(endpoint); // <-- Insert the correct URL
// Setup security
BasicHttpBinding basic = new BasicHttpBinding();
basic.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
basic.Security.Transport.ClientCredentialType = HttpClientCredentialType.Basic;
pws2.SendClient client = new pws2.SendClient(basic,ea);
client.ClientCredentials.UserName.UserName = “xxx”; // <-- Change here
client.ClientCredentials.UserName.Password = “yyy”; // <-- Change here
try
{
// The result contains 200 + mark (e.g. “200 PWS.xxx.xxx”)
// Save this in a database for later reference!
string result = client.sendWithAddressing(data, “EBREVWEBB”, “Your company”, “EBREVWEBB”);
// <-- Change TfsSnd (Your Company) here
lstStatus.Items.Add(“Result:” + result);
}
catch (FaultException fe)
{
// Add proper error handling!
MessageBox.Show(fe.ToString());
}
catch (Exception eee)
{ MessageBox.Show(eee.ToString());}
```

Strålfors Svenska AB
08-508 830 00
www.stralfors.se