WEBSERVICE «BARCODE»
ANLEITUNG FÜR ZUGRIFF MIT PHP

Version 2.4
Juli 2018

DIE POST

Inhaltsverzeichnis

Referenzen

[1] Webservice «Barcode», Handbuch, http://www.post.ch/post-barcode-handbuch.pdf

[2] Dokumentationen zu Webservice «Barcode», http://www.post.ch/post-barcode-cug.htm

1 Einleitung

Diese Dokumentation erklärt die Verwendung des Webservice «Barcode» in PHP. An einigen typischen Beispielen wird die Verwendung des Webservice erklärt.

Diese Dokumentation ist keine vollständige Spezifikation der Webserviceschnittstelle, sondern lediglich eine Anleitung, wie der Webservice in PHP verwendet werden kann.

Für eine vollständige Dokumentation des Webservice «Barcode» verweisen wir auf das Webservice-Benutzerhandbuch [1]. Dort werden auch sämtliche Begriffe, Attribute, Dienstleistungscodes, usw. erklärt. Weitere Ressourcen und Hilfsmittel zum Webservice «Barcode» finden Sie auf der Webservice-Barcode-Webseite [2].

Sämtliche Beispiele stehen auch als PHP-Sources zur Verfügung. Die hier abgedruckten Beispiele verwenden ein paar einfache Hilfsmethoden, die in der Datei **wsbc-utils.php** zur Verfügung stehen.

Alle inhaltlich geänderten Abschnitte haben wir am Rand mit einem Strich markiert.

2 Konfiguration

2.1 WSDL-Dateien

Der PHP SOAPClient hat ein Problem mit dem Zugriff auf WSDL-Dateien über Proxies. Aus diesem Grund wird dringend empfohlen, das WSDL-File und das zugehörige XSD-File lokal für den Zugriff aus PHP abzulegen.

Laden Sie dazu die folgenden beiden XML-Dateien herunter und speichern Sie diese unter angegebenem Dateinamen in einem Verzeichnis, auf das unter PHP zugegriffen werden kann. Für die folgenden Beispiele wird angenommen, dass diese Dateien im gleichen Verzeichnis liegen wie die PHP-Sources.

Für die folgenden Beispiele wird angenommen, dass diese Dateien im gleichen Verzeichnis liegen wie die PHP Source-Files.

URL für Download	Speichern unter Dateiname
http://www.post.ch/post-barcode-description-service.wsdl	barcode_v2_4.wsdl
http://www.post.ch/barcode_v2_4.xsd	barcode_v2_4.xsd

ACHTUNG: Beide Dateien müssen im gleichen Verzeichnis liegen und die Datei barcode_v2_4.xsd muss exakt so benannt werden, da diese Datei in der WSDL-Datei eingebunden wird.

In den PHP-Beispiel-Sources sind diese Dateien ebenfalls enthalten.

2.2 Konfiguration SOAPClient

Der Zugriff auf den Webservice erfolgt über eine SOAPClient-Instanz. Das folgende Beispiel demonstriert die Initialisierung des SOAPClient-Objekts für den Zugriff auf den Webservice «Barcode».

Die auskommentierten Parameter sind optional und können bei Bedarf verwendet werden.

```
// SOAP Configuration

$username = 'your username';
$password = 'your password';
$endpoint_url= 'http://anylocationURL/v2_4';
$SOAP_wsdl_file_path= 'barcode_v2_4.wsdl';

$SOAP_config = array(
    // Webservice Endpoint URL
    'location' => 'https://wsbc.post.ch/wsbc/barcode/v2_4,

    // Webservice Barcode Login
    'login' => $username,
    'password' => $password,

    // Encoding for Strings
    // 'encoding' => 'ISO-8859-1',

    // Optional Proxy Config
    // (if you are behind a proxy):
    // 'proxy_host' => 'proxy.mydomain.com',
    // 'proxy_port' => 8080,

    // Optional Proxy Authentication
    // (if your proxy needs a username and password):
    // 'proxy_login' => 'proxy-username',
    // 'proxy_password' => 'proxy-password',

    // Addtional debug trace information:
    // 'trace' => true,

    // Connection timeout (in seconds):
    // 'connection_timeout' => 90

);

// SOAP Client Initialization
try {
    $SOAP_Client = new SoapClient($SOAP_wsdl_file_path, $SOAP_config);
}
catch (SoapFault $fault) {
    echo('Error in SOAP Initialization: '. $fault -> __toString() .'<br/>');
    exit;
}
```

Eine Beispielkonfiguration steht auch in der Datei **wsbc-init.php** zur Verfügung, die in allen anderen Beispielen eingebunden wird.

Damit die Beispiele funktionieren, müssen in der Konfiguration ein gültiger Username und ein gültiges Passwort für den Zugriff auf den Webservice «Barcode» eingetragen werden.

Alle folgenden Beispiele setzen voraus, dass ein entsprechender SOAPClient vorgängig korrekt initialisiert wurde.

2.3 Proxykonfiguration

Ein allfälliger Proxy für den Zugriff aufs Internet muss im **SOAP_config**-Array entsprechend konfiguriert werden. Siehe auskommentierte Zeilen im vorherigen Beispiel im Abschnitt 2.2.

2.4 Zeichensatz-Encoding

Der SOAPClient kann so konfiguriert werden, dass er mit einem anderen Zeichensatz-Encoding als UTF-8 arbeitet. Siehe dazu die Einstellung «encoding» im vorherigen Beispiel in Abschnitt 2.2. Diese Einstellung bewirkt, dass alle übergebenen Strings im eingestellten Encoding erwartet und entsprechend nach UTF-8 gewandelt werden. Alle Rückgabewerte werden ebenfalls von UTF-8 in den entsprechend eingestellten Zeichensatz zurückgewandelt.

In dieser Einstellung ist somit möglichst das Encoding einzustellen, das in den PHP-Sources verwendet wird, damit nicht ständig die Strings encodiert werden müssen.

Die mitgelieferten PHP-Beispiel-Sources sind übrigens in UTF-8 codiert, womit diese Einstellung in den Beispielen entfällt.

3 Lese-Operationen

Mit den Lese-Operationen können alle aktuell unterstützten Dienstleistungscodes und Labellayouts des Webservice abgefragt werden.

Das folgende PHP-Beispiel zeigt die Verwendung all dieser Lese-Operationen. Zudem zeigt es alle aktuell verfügbaren Dienstleistungscodes und Labellayouts in HTML.

```
include("wsbc-init.php");
include("wsbc-utils.php");

// covered requests:
// 1. ReadServiceGroups
// 2.1 ReadBasicServices
// 2.1.1 ReadAdditionalServices
// 2.1.2 ReadDeliveryInstructions
// 2.1.3 ReadLabelLayouts
// 3. ReadAllowedServicesByFrankingLicense

echo "<html><header></header><body>";

//
// 1. Read all service groups
$serviceGroupsRequest = array('Language' => 'de');
try {
    $serviceGroupsResponse = $SOAP_Client -> ReadServiceGroups($serviceGroupsRequest);
}
catch (SoapFault $fault) {
    echo('Error in ReadServiceGroups: '. $fault -> __toString() .'<br />');
    exit;
}

echo "<h1>Read Services Test Execution</h1>";
echo "<br/>";
echo "<br/>";

//
// 2. For each service group: read and display service codes and label layouts
foreach (getElements($serviceGroupsResponse->ServiceGroup) as $group) {
    echo "<b>Available service codes and layouts for Service-Group '".$group-
>Description."'</b>";
    echo "<br/>";
    echo "<br/>";

    //
    // 2.1. Basic services for service group
    $basicServicesRequest = array('Language' => 'de', 'ServiceGroupID' => $group-
>ServiceGroupID);
    try {
        $basicServicesResponse = $SOAP_Client -> ReadBasicServices($basicServicesRequest);
    }
    catch (SoapFault $fault) {
        echo('Error in ReadBasicServices: '. $fault -> __toString() .'<br />');
        exit;
    }
```

```
    echo "Basic Services:";
    echo "<ul>";
    foreach (getElements($basicServicesResponse->BasicService) as $basicService) {
        echo "<li>";
        $basicServiceCodes = getElements($basicService->PRZL);
        echo toCommaSeparatedString($basicServiceCodes);

        //
        // 2.1.1. Additional services for basic service
        $additionalServicesRequest = array('Language' => 'de', 'PRZL' =>
$basicServiceCodes);
        try {
            $additionalServicesResponse = $SOAP_Client -> ReadAdditionalServices
($additionalServicesRequest);
        }
        catch (SoapFault $fault) {
            echo('Error in ReadAdditonalServices: '. $fault -> __toString() .'<br />');
            exit;
        }
        echo "<ul>";
        echo "<li>Additional service codes: ";
        $delimiter = "";
        foreach (getElements($additionalServicesResponse->AdditionalService) as
$additionalService) {
            echo $delimiter.$additionalService->PRZL;
            $delimiter = ",";
        }
        echo "</li>";
        echo "</ul>";
        // 2.1.1.
        //

        //
        // 2.1.2. Delivery instructions for basic service
        $deliveryInstructionsRequest = array('Language' => 'de', 'PRZL' =>
$basicServiceCodes);
        try {
            $deliveryInstructionsResponse = $SOAP_Client -> ReadDeliveryInstructions
($deliveryInstructionsRequest);
        }
        catch (SoapFault $fault) {
            echo('Error in ReadDeliveryInstructions: '. $fault -> __toString() .'<br />');
            exit;
        }

        echo "<ul>";
        echo "<li>Delivery instruction codes: ";
        $delimiter = "";
        foreach (getElements($deliveryInstructionsResponse->DeliveryInstructions) as
$deliveryInstruction) {
            echo $delimiter.$deliveryInstruction->PRZL;
            $delimiter = ",";
        }
```

```
        echo "</li>";
        echo "</ul>";
        // -- 2.1.2.
        //

        //
        // 2.1.3. Label layouts for basic service
        $labelLayoutsRequest = array('Language' => 'de', 'PRZL' => $basicServiceCodes);
        try {
            $labelLayoutsResponse = $SOAP_Client -> ReadLabelLayouts
($labelLayoutsRequest);
        }
        catch (SoapFault $fault) {
            echo('Error in ReadLabelLayouts: '. $fault -> __toString() .'<br />');
            exit;
        }

        echo "<ul>";
        echo "<li>Label Layouts: ";
        // elements
        echo "<ul>";
        foreach (getElements($labelLayoutsResponse->LabelLayout) as $labelLayout) {
            echo "<li>";
            echo $labelLayout->LabelLayout.": ";
            echo "max. ".$labelLayout->MaxServices." services ";
            echo "and ".$labelLayout->MaxDeliveryInstructions." delivery instructions, ";
            if ($labelLayout->FreeTextAllowed) {
                echo "freetext allowed";
            }
            else {
                echo "freetext not allowed";
            }
            echo "</li>";
        }
        echo "</ul>";
        // -- elements
        echo "</li>";
        echo "</ul>";
        // -- 2.1.3.
        //

        echo "</li>";
    }
    echo "</ul>";

}

echo "<br/>";
echo "<br/>";
echo "<br/>";
echo "<br/>";
echo "<h1>Allowed Services By Franking License XXXXX</h1>";
//
```

```php
// 3. Read allowed services by franking license
$allowedServicesRequest = array('FrankingLicense' => 'XXXXX', 'Language' => 'de');
try {
    $allowedServicesResponse = $SOAP_Client -> ReadAllowedServicesByFrankingLicense
($allowedServicesRequest);
}
catch (SoapFault $fault) {
    echo('Error in ReadAllowedServicesByFrankingLicense: '. $fault -> __toString() .'<br
/>');
    exit;
}

echo "<ul>";
foreach (getElements($allowedServicesResponse->ServiceGroups) as $serviceGroup) {
    echo "<li>";
    echo "ServiceGroup: ".$serviceGroup->ServiceGroup->ServiceGroupID.", ".$serviceGroup-
>ServiceGroup->Description."";
    echo "<ul>";
    foreach (getElements($serviceGroup->BasicService) as $basicService) {
        echo "<li>";
        $przls = count($basicService->PRZL);
        if ($przls > 1) {
            echo "BasicService: ".$basicService->Description.": ".toCommaSeparatedString
($basicService->PRZL)."";
        } else {
            echo "BasicService: ".$basicService->Description.": ".$basicService->PRZL."";
        }
        echo "</li>";
    }
    echo "</ul>";
    echo "</li>";
    echo "<br/>";
}
echo "</ul>";

echo "</body></html>";
```

Dieses Beispiel steht in der Datei **wsbc-read.php** zur Verfügung.

4 Validierungsoperation

Der Webservice stellt eine Operation zur Verfügung, mit der eine Kombination von Dienstleistungscodes validiert werden kann. Es wird überprüft, ob die im Request enthaltenen Servicecodes (Basisleistung, Zusatzleistungen und Zustellanweisungen) miteinander kombiniert werden dürfen. Ebenfalls wird überprüft, ob die verwendeten Codes im gewählten Labelformat (A5, A6, A7 oder FE) erlaubt sind. Je nach Format ist nur eine gewisse Anzahl an Dienstleistungscodes und Zustellanweisungen erlaubt.

Das folgende Beispiel demonstriert die Überprüfung einer Kombination von Basisleistungscodes, Zusatzleistungscodes und Zustellanweisungscodes.

```
include("wsbc-init.php");
include("wsbc-utils.php");

echo "<html><header></header><body>";

// 1. Define Validation Request
// (see documentation of structure in "Barcode web service manual")
$validationRequest = array(
    'Language' => 'de',
    'Envelope' => array(
    'LabelDefinition' => array(
        'LabelLayout' => 'A5'
        ),
    'Data' => array(
        'Provider' => array(
            'Sending' => array(
                'Item' => array(
                    array( // 1.Item ...
                        'ItemID' => '1',
                        'Attributes' => array(
                        'PRZL' => array(
                        // At least one code is required (schema validation)
                        // Basic service code(s) (optional, default="ECO"):
                        'PRI',
                        // Additional service codes (optional)
                        'FRA',
                        'SI',
                        // Delivery instruction codes (optional)
                        'ZAW3213',
                        'ZAW3215',
                    )
                )
            ),
                // Add additional items here for multiple requests in one web service
call ...
                // array( // 2.Item ...
                //     ... // same structure as above
                // ),
))))));

// 2. Web service call
try {
    $response = $SOAP_Client -> ValidateCombination($validationRequest);
}
```

```
catch (SoapFault $fault) {
    echo('Error in ValidateCombination: '. $fault -> __toString() .'<br />');
}

// 3. Process response
// (see documentation of structure in "Barcode web service manual")
foreach (getElements($response->Envelope->Data->Provider->Sending->Item) as $item) {
    if ($item->Errors != null) {
        // Errors in validation ...
        // (error messages are returned in the requested language)
        $errorMessages = "";
        foreach (getElements($item->Errors->Error) as $error) {
            $errorMessages .= $error->Message.',';
        }
        echo '<p>Validation-ERROR for item with itemID='.$item->ItemID.":
".$errorMessages.'.<br/></p>';
    }
    else {
        // Successful validation
        echo '<p>Validation was successfull for service code combination in item with
itemID='.$item->ItemID.'.<br/></p>';

        // Also display warnings
        if ($item->Warnings != null) {
            $warningMessages = "";
            foreach (getElements($item->Warnings->Warning) as $warning) {
                $warningMessages .= $warning->Message.",";
            }
            echo 'with WARNINGS: '.$warningMessages.'.<br/>';
        }
    }
}

echo "</body></html>";
```

Dieses Beispiel validiert ohne Fehler.

Wird hingegen in dem Beispiel „ZAW3215" durch „ZAW3222" ersetzt, so liefert der Webservice Aufruf entsprechende Validierungs-Fehlermeldungen zurück.

Dieses Beispiel steht in der Datei **wsbc-validate.php** zur Verfügung.

5 Labelgenerierung

Das folgende Beispiel demonstriert, wie ein einzelner Adressträger in PHP generiert und die Response ausgelesen werden kann. Der generierte Adressträger wird in einer Bilddatei abgespeichert.

```
include("wsbc-init.php");
include("wsbc-utils.php");

// Franking License Configuration
// TODO: you have to set this to a valid franking license for your barcode web service
user account!
$frankinglicense = '1234567890';

//$imgfile = 'default_logo.gif';
//$logo_binary_data = fread(fopen($imgfile, "r"), filesize($imgfile));

// 1. Define Label Request
// (see documentation of structure in "Barcode web service manual")
$generateLabelRequest = array(
    'Language' => 'de',
    'Envelope' => array(
        'LabelDefinition' => array(
            'LabelLayout' => 'A5',
            'PrintAddresses' => 'RecipientAndCustomer',
            'ImageFileType' => 'GIF',
            'ImageResolution' => 300,
            'PrintPreview' => false
        ),
        'FileInfos' => array(
            'FrankingLicense' => $frankinglicense,
            'PpFranking' => false,
            'Customer' => array(
                'Name1' => 'Meier AG',
                // 'Name2' => 'Generalagentur',
                'Street' => 'Viktoriaplatz 10',
                // 'POBox' => 'Postfach 600',
                'ZIP' => '8050',
                'City' => 'Zürich',
                // 'Country' => 'CH',
                'Logo' => $logo_binary_data,
                'LogoFormat' => 'PNG',
                'LogoRotation' => 0,
                'LogoAspectRatio' => 'KEEP',
                'LogoHorizontalAlign' => 'WITH_CONTENT',
                'LogoVerticalAlign' => 'MIDDLE',
                'DomicilePostOffice' => '3000 Bern'
        ),
        'CustomerSystem' => 'PHP Client System'
    ),
    'Data' => array(
        'Provider' => array(
            'Sending' => array(
                'Item' => array(
                    array( // 1.Item ...
                        'ItemID' => '1234',
```

```
                        // 'ItemNumber' => '12345678',
                        // 'IdentCode' => '1234',
                        'Recipient' => array(
                            //'PostIdent' => 'IdentCodeUser',
                            'Title' => 'Frau',
                            'Vorname' => 'Melanie',
                            'Name1' => 'Steiner',
                            //'Name2' => 'Müller AG',
                            'Street' => 'Viktoriastrasse',
                            'HouseNo' => '21',
                            //'FloorNo' => '1',
                            //'MailboxNo' => '1111',
                            'ZIP' => '3030',
                            'City' => 'Bern 1',
                            //'Country' => 'CH',
                            'Phone' => '0313381111', // für ZAW3213
                            //'Mobile' => '0793381111',
                            'EMail' => 'h.muster@post.ch'
                            ///'LabelAddress' => array(
                            //  'LabelLine' => array('LabelLine 1',
                            //            'LabelLine 2',
                            //            'LabelLine 3',
                            //            'LabelLine 4',
                            //            'LabelLine 5'
                            //            )
                            //)
                        ),
                        //'AdditionalINFOS' => array(
                        // Cash-On-Delivery amount in CHF for service 'BLN':
                        //    'AdditionalData' => array(
                        //    'Type' => 'NN_BETRAG',
                        //        'Value' => '12.5'
                        //    ),
                        // Cash-On-Cary ESR-Reference-No. for service 'BLN':
                        //    AdditionalData' => array(
                        //        'Type' => 'NN_ESR_REF_REFNR',
                        //        'Value' => 'xxxxxxxxxxxxxxxxxxxxxx' // a valid ESR
Ref. Nr. (with or without spaces)
                        //    ),
                        //),
                        'Attributes' => array(
                        'PRZL' => array(
                             // At least one code is required (schema validation)

                             // Basic service code(s) (optional, default="ECO"):

                             'PRI',
                             // Additional service codes (optional)
                             'N',
                             'FRA',
                             // Delivery instruction codes (optional)
                             'ZAW3211',
                             'ZAW3213'
                          ),
```

```
                        'FreeText' => 'Freitext',
                        // 'DeliveryDate' => '2010-06-19',
                        // 'ParcelNo' => 2,
                        // 'ParcelTotal' => 5,
                        // 'DeliveryPlace' => 'Vor der Haustüre',
                        'ProClima' => true,
                    )
                        // Setting notifications
                        //'Notification' => array(
                        //     'Service' => 64,
                        //     'Communication' => array(
                        //         'Email' => 'test@test.ch'
                        //     ),
                        //     'FreeText1' => 'Free-Text 1',
                        //     'FreeText2' => 'Free-Text 2',
                        //     'Language' => 'DE'
                        //)
                    )
                //,
                // Add addtional items here for multiple requests in one web service
call ...
                // array( // 2.Item ...
                //     ... // same structure as above
                // ),
                )
            )
        )
    )
  )
);

// Setting attribute of notification
// $notification = $generateLabelRequest -> Data -> Provider -> Sending -> Item[0]->
Notification;
// $notification['Type'] = ,EMAIL';

// 2. Web service call
$response = null;
try {
    $response = $SOAP_Client -> GenerateLabel($generateLabelRequest);
}
catch (SoapFault $fault) {
    echo('Error in GenerateLabel: '. $fault -> __toString() .'<br />');
}
// 3. Process requests: save label images and check for errors
// (see documentation of structure in "Barcode web service manual")
foreach (getElements($response->Envelope->Data->Provider->Sending->Item) as $item) {
    if ($item->Errors != null) {

        // Error in Label Request Item:
        // This barcode label was not generated due to errors.
        // The received error messages are returned in the specified language of the
request.
```

```
            // This means, that the label was not generated,
            // but other labels from other request items in same call
            // might have been generated successfully anyway.
            $errorMessages = "";
            $delimiter = "";
            foreach (getElements($item->Errors->Error) as $error) {
                $errorMessages .= $delimiter.$error->Message;
                $delimiter = ",";
            }
            echo '<p>ERROR for item with itemID='.$item->ItemID.":
    ".$errorMessages.'.<br/></p>';

        }
        else {
            // Get successfully generated label as binary data:
            $identCode = $item->IdentCode;
            $labelBinaryData = $item->Label;

            // Save the binary image data to image file:
            $filename =
    'outputfolder\testOutput_GenerateLabel_'.$identCode.'.gif';
            file_put_contents($filename, $labelBinaryData);

            // Printout some label information (and warnings, if any):
            echo '<p>Label generated successfully for identCode='.$identCode.': <br/>';
            if ($item->Warnings != null) {
                $warningMessages = "";
                foreach (getElements($item->Warnings->Warning) as $warning) {
                    $warningMessages .= $warning->Message.",";
                }
                echo 'with WARNINGS: '.$warningMessages.'.<br/>';
            }
            echo $filename.':<br/><img src="'.$filename.'"/><br/>';
            echo '</p>';
        }
    }

    echo "</body></html>";
```

Wichtige Anmerkungen zur Operation GenerateLabel:

- Damit das obige Beispiel funktioniert, muss in jedem Fall eine gültige Frankierlizenz für den verwendeten Webservice-Login gesetzt sein.
- Enthält ein Item in einer Response Fehler, so bedeutet das, dass dieser Adressträger nicht generiert werden konnte. Es kann allerdings sein, dass andere Adressträger im gleichen Request korrekt generiert wurden.
- Auch wenn ein Adressträger erfolgreich generiert wurde, kann es sein, dass das Item Warnungen enthält. Dies ist vor allem dann der Fall, wenn zum Beispiel Parameter übergeben wurden, die für die gewählten Dienstleistungen gar nicht benötigt werden.

In einem GenerateLabelRequest können noch diverse weitere Felder gesetzt werden. Wir verweisen dazu auf die vollständige Dokumentation des Webservice «Barcode» [1], insbesondere auf Abschnitt 5.3.1.

Ausserdem finden Sie in der Beispieldatei **wsbc-generate.php** das obige Beispiel mit zusätzlichen Kommentaren und allen optionalen Feldern, die noch zusätzlich gesetzt werden können.

6 Single Barcodes Generierung

Das folgende Beispiel demonstriert, wie mehrere Barcode-Elemente der Funktion GenerateSingleBarcodes in PHP generiert und die Response ausgelesen werden kann. Die generierten Barcode-Elemente werden als Bild-Dateien abgespeichert.

```
include("wsbc-init.php");
include("wsbc-utils.php");

// Franking License Configuration
// TODO: you have to set this to a valid franking license for your barcode web service
user account!
$frankinglicense = '1234567890';

// 1. Define Label Request
// (see documentation of structure in "Barcode web service manual")
$generateSingleBarcodesRequest = array(
    'Language' => 'de',
    'Envelope' => array(
        'BarcodeDefinition' => array(
            'ImageFileType' => 'PNG',
            'ImageResolution' => 300,
        ),
        'FileInfos' => array(
            'FrankingLicense' => $frankinglicense,
            'PpFranking' => false,
            'Customer' => array(
                'Name1' => 'Meier AG',
                // 'Name2' => 'Generalagentur',
                'Street' => 'Viktoriaplatz 10',
                // 'POBox' => 'Postfach 600',
                'ZIP' => '8050',
                'City' => 'Zürich',
                // 'Country' => 'CH',
                'DomicilePostOffice' => '3000 Bern'
        ),
    ),
    'Data' => array(
        'Provider' => array(
            'Sending' => array(
                'Item' => array(
                    array( // 1.Item ...
                        'ItemID' => '1234',
                        // 'ItemNumber' => '12345678',
                        // 'IdentCode' => '1234',
                        'Recipient' => array(
                            //'PostIdent' => 'IdentCodeUser',
                            'Title' => 'Frau',
                            'Vorname' => 'Melanie',
                            'Name1' => 'Steiner',
                            //'Name2' => 'Müller AG',
                            'Street' => 'Viktoriastrasse',
                            'HouseNo' => '21',
                            //'FloorNo' => '1',
                            //'MailboxNo' => '1111',
```

```
                        'ZIP' => '3030',
                        'City' => 'Bern 1',
                        //'Country' => 'CH',
                        //'Phone' => '0313381111', // für ZAW3213
                        //'Mobile' => '0793381111',
                        'EMail' => 'h.muster@post.ch'
                        ///'LabelAddress' => array(
                        //  'LabelLine' => array('LabelLine 1',
                        //            'LabelLine 2',
                        //            'LabelLine 3',
                        //            'LabelLine 4',
                        //            'LabelLine 5'
                        //            )
                        //)
                    ),
                    //'AdditionalINFOS' => array(
                    // Cash-On-Cary amount in CHF for service 'BLN':
                    //    'AdditionalData' => array(
                    //    'Type' => 'NN_BETRAG',
                    //        'Value' => '12.5'
                    //    ),
                    // Cash-On-Cary ESR-Reference-No. for service 'BLN':
                    //    AdditionalData' => array(
                    //        'Type' => 'NN_ESR_REF_REFNR',
                    //        'Value' => 'xxxxxxxxxxxxxxxxxxxxxx' // a valid ESR
Ref. Nr. (with or without spaces)
                    //    ),
                    //),
                    'Attributes' => array(
                      'PRZL' => array(
                         // At least one code is required (schema validation)
                         'eAR',
                         'RINL',
                         'ZAW2511'
                      ),

                      'FreeText' => 'Freitext',
                      // 'DeliveryDate' => '2010-06-19',
                      // 'ParcelNo' => 2,
                      // 'ParcelTotal' => 5,
                      // 'DeliveryPlace' => 'Vor der Haustüre',
                      'ProClima' => true
                    )
                    //'Notification' => array(
                    // // Notification structure ...
                    //)
                )
            //,
            // Add additional items here for multiple requests in one web service
call ...
            // array( // 2.Item ...
            //     ... // same structure as above
            // ),
            )
```

```
            )
        )
    )
  )
);

// 2. Web service call
$response = null;
try {
    $response = $SOAP_Client -> GenerateSingleBarcodes($generateSingleBarcodesRequest);
}
catch (SoapFault $fault) {
    echo('Error in GenerateSingleBarcodes: '. $fault -> __toString() .'<br />');
}

// 3. Process requests: save label images and check for errors
// (see documentation of structure in "Barcode web service manual")
foreach (getElements($response->Envelope->Data->Provider->Sending->Item) as $item) {
    if ($item->Errors != null) {

        // Error in Single Request Item:
        // This barcode label was not generated due to errors.
        // The received error messages are returned in the specified language of the
request.
        // This means, that the label was not generated,
        // but other labels from other request items in same call
        // might have been generated successfully anyway.
        $errorMessages = "";
        $delimiter = "";
        foreach (getElements($item->Errors->Error) as $error) {
            $errorMessages .= $delimiter.$error->Message;
            $delimiter = ",";
        }
        echo '<p>ERROR for item with itemID='.$item->ItemID.":
".$errorMessages.'.<br/></p>';

}
    else {
        // Get successfully generated label as binary data:
        $itemID = $item->IdentID;
        $identCode = $item->IdentCode;

        $counter = 1;
        $basePath =
'outputfolder\testOutput_GenerateSingleBarcodes_'.$identCode.'_';
        foreach (getElements($item->Barcodes->Barcode) as $barcode) {
            // Save the binary image data to image file:
            $filename = ''.$basePath.''.$counter.'.gif';
            file_put_contents($filename, $barcode);
            $counter++;
        }
        $numberOfItems = $counter - 1;
         // Printout some label information (and warnings, if any):
        echo '<p>'.$numberOfItems.'Barcodes successfully generated for
identCode='.$identCode.': <br/>';
```

```
        if ($item->Warnings != null) {
            $warningMessages = "";
            foreach (getElements($item->Warnings->Warning) as $warning) {
                $warningMessages .= $warning->Message.",";
            }
            echo 'with WARNINGS: '.$warningMessages.'.<br/>';
        }
        echo 'All files start with: <br/><img src="'.$basePath.'"/><br/>';
        echo '</p>';
    }
}

echo "</body></html>";
```

Wichtige Anmerkungen zur Operation GenerateSingle-Barcodes:

– Damit das obige Beispiel funktioniert, muss in jedem Fall eine gültige Frankierlizenz für den verwendeten Webservice Login gesetzt sein.
– Enthält ein Item in einer-GenerateSingleBarcodes Response Fehler, so bedeutet das, dass die Barcode-Elemente nicht generiert werden konnten.
– Auch wenn die Barcode-Elemente erfolgreich generiert wurden, kann es sein, dass das Response-Item Warnungen enthält. Dies ist vor allem dann der Fall, wenn zum Beispiel Parameter übergeben wurden, die für die gewählten Dienstleistungen gar nicht benötigt werden.

In einem GenerateSingleBarcodes Request können noch diverse weitere Felder gesetzt werden. Wir verweisen dazu auf die vollständige Dokumentation des Webservice Barcodes [1], insbesondere auf Abschnitt 5.5.1.

Ausserdem finden Sie in der Beispiel-Datei **wsbc-generate-singlebarcodes.php** das obige Beispiel mit zusätzlichen Kommentaren mit allen optionalen Feldern, welche noch zusätzlich gesetzt werden können.

7 Individual Barcodes Generierung

Das folgende Beispiel demonstriert, wie einzelne Barcodes mit der Funktion GenerateBarcodes in PHP generiert und die Response ausgelesen werden kann. Die Barcodes werden als Bild-Dateien abgespeichert.

```
include("wsbc-init.php");
include("wsbc-utils.php");

echo "<html><header></header><body>";

// 1. Define Label Request
// (see documentation of structure in "Barcode web service manual")
$generateBarcodeRequest = array(
    'Language' => 'de',
    'BarcodeDefinition' => array(
        'BarcodeType' => 'LSO_2',
        'ImageFileType' => 'PNG',
        'ImageResolution' => 200
    )
);

// 2. Web service call
$response = null;
try {
    $response = $SOAP_Client -> GenerateBarcode($generateBarcodeRequest);
}
catch (SoapFault $fault) {
    echo('Error in GenerateBarcode: '. $fault -> __toString() .'<br />');
}

// 3. Process requests: save label images and check for errors
// (see documentation of structure in "Barcode web service manual")
foreach (getElements($response->Data) as $data) {
    if ($data->Errors != null) {

        // Error in Barcode Request Item:
        // This barcode label was not generated due to errors.
        // The received error messages are returned in the specified language of the
request.
        // This means, that the label was not generated,
        // but other labels from other request items in same call
        // might have been generated successfully anyway.
        $errorMessages = "";
        $delimiter = "";
        foreach (getElements($data->Errors->Error) as $error) {
            $errorMessages .= $delimiter.$error->Message;
            $delimiter = ",";
        }
        echo "<p>ERROR for request: ".$errorMessages.'.<br/></p>';

    }
```

```
    else {
        // Get successfully generated barcide as binary data:
        $barcodeBinaryData = $data->Barcode;
        $deliveryNoteRef = $data->DeliveryNoteRef;
        $barcodeDefinition = $data->BarcodeDefinition;
        $barcodeType = $barcodeDefinition->BarcodeType;
        $imageFileType = $barcodeDefinition->ImageFileType;
        $imageResolution = $barcodeDefinition->ImageResolution;

        // Save the binary image data to image file:
        $filename =
'outputfolder\testOutput_GenerateBarcode_'.$deliveryNoteRef.'.gif';
        file_put_contents($filename, $barcodeBinaryData);

        // Printout some label information (and warnings, if any):
        echo '<p>Label generated successfully for Delivery Note Reference =
'.$deliveryNoteRef.': <br/>';
        if ($data->Warnings != null) {
            $warningMessages = "";
            foreach (getElements($data->Warnings->Warning) as $warning) {
                $warningMessages .= $warning->Message.",";
            }
            echo 'with WARNINGS: '.$warningMessages.'.<br/>';
        }
        echo $filename.':<br/><img src="'.$filename.'"/><br/>';
        echo '</p>';
    }
}

echo "</body></html>";
```

Wichtige Anmerkungen zur Operation GenerateBarcode:

– Enthält ein Data-Objekt in einer GenerateBarcode-Response Fehler, so bedeutet das, dass der Barcode nicht generiert werden konnte.
– Auch wenn der Barcode erfolgreich generiert wurde, kann es sein, dass das Response-Data-Objekt Warnungen enthält. Dies ist vor allem dann der Fall, wenn zum Beispiel Parameter übergeben wurden, die für die gewählten Dienstleistungen gar nicht benötigt werden.

Für weitere Informationen verweisen wir auf die vollständige Dokumentation des Webservice Barcodes [1], insbesondere auf Abschnitt 5.4.1.

Ausserdem finden Sie in der Beispiel-Datei **wsbc-generate-barcode.php** das obige Beispiel.

4079 de 07.2018

Post CH AG
Support Webservices
Wankdorfallee 4
Postfach
3030 Bern

webservice@post.ch
www.post.ch/webservice

DIE POST