GLS ONLINE API

GLS online SOAP webservice for printing services

URLs:

HU - https://online.gls-hungary.com/webservices/soap_server.php?wsdl&ver=18.09.12.01

SK - https://online.gls-slovakia.sk/webservices/soap_server.php?wsdl&ver=18.09.12.01

CZ - https://online.gls-czech.com/webservices/soap_server.php?wsdl&ver=18.09.12.01

RO - https://online.gls-romania.ro/webservices/soap_server.php?wsdl&ver=18.09.12.01

SI - https://connect.gls-slovenia.com/webservices/soap_server.php?wsdl&ver=18.09.12.01

HR - https://online.gls-croatia.com/webservices/soap_server.php?wsdl&ver=18.09.12.01

Style: rpc

Operation type: *Request-response.* The endpoint receives a message, and sends a correlated message.

printlabel method description

Purpose: prepare and print one GLS shipment data and labels

SOAP action: printlabel

Input: printlabelRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*	
***password** type *string*	*- password – request it from GLS*	
***senderid** type *string*	*- GLS client number – request it from GLS*	
***sender_name** type *string*	*- sender name*	
***sender_address** type *string*	*- sender address*	
***sender_city** type *string*	*- sender city*	
***sender_zipcode** type *string*	*- sender zip code*	
***sender_country** type *string*	*- sender country code or country name*	
sender_contact type *string*	*- sender contact person*	
sender_phone type *string*	*- sender phone*	
sender_email type *string*	*- sender email address*	
***consig_name** type *string*	*- consignee name*	
***consig_address** type *string*	*- consignee address*	
***consig_city** type *string*	*- consignee city*	
***consig_zipcode** type *string*	*- consignee zip code*	
***consig_country** type *string*	*- consignee country code or country name*	
consig_contact type *string*	*- consignee contact person*	
consig_phone type *string*	*- consignee phone or SMS number for services*	
consig_email type *string*	*- consignee email address – also used for services*	
***pcount** type *int*	*- count of the parcels / labels to print*	
***pickupdate** type *string*	*- pickup date in format yyyy-MM-dd*	
content type *string*	*- content of the parcel – info printed on label*	
clientref type *string*	*- client reference*	
codamount type *decimal*	*- COD amount*	
codref type *string*	*- COD reference – used if COD amount is set*	
services type *svcDataArray* - array of type *svcData*	*- array of services*	

- code type *string* - *3 letter service code, please see list of services in Appendix A*
- info type *string* - *parameter for service*

***printertemplate** type *string*	*- type of the printer – list in Appendix B*
***printit** type *Boolean*	*- true, if label has to be printed, false if stored in the list*

***timestamp** type *string*	*- timestamp of the request in format yyyyMMddHHmmss (some time difference is tolerated, 10 minutes)*
***hash** type *string*	*- hash code of the request – Appendix C*
customlabel type *Boolean*	*- if true, client will handle label printing – no label data returned*
is_autoprint_pdfs type *Boolean*	*- if false then js print() command will be omitted from the label pdfs*
gapid type *string*	*- ONLY IN HU! if it is exists and valid, the function is ready to use*

- *Mandatory fields marked with asterisk
- do not pass null for optional values, empty string is required
    - as **services** pass at least an empty array
- to avoid spamming of the service, same request can be sent max. 5 times within 5 minutes. In case it is required to print more labels with same data, use **pcount** variable to pass count

Output:

printlabelResponse (soap:body, use = encoded)

return type *printlabel_result*

- **pcls** type *ArrayOfString* - array of type *string – return list of the parcel numbers printed*
- **pcls_withcheckdigit** type *ArrayOfString* - array of type *string – return list of the parcel numbers with checkdigit*
- **pdfdata** type *string* *- pdf data to print or display pdf. Encoded with base64*
- **depo** type *string* *- additional data for custom printing on client side*
- **driver** type *string* *- additional data for custom printing on client side*
- **successfull** type *Boolean - if false, errcode and errdesc will be set*
- **errcode** type *int* *- numeric error code, please list in AppendixD*
- **errdesc** type *string* *- error description*

- **pcls** and **pdfdata** will be returned only if **printit** is *true* and **successfull** is *true*
- **depo**and **driver** will be returned only if **customlabel** is *true* – no pdfdata returned in this case

- sample code for NuSOAP client in Appendix E
- sample code for handling pdfdata from result in c# in Appendix F

getglshash method description

Purpose: for users who not able to generate the hash code themselves.
WARNING: It generates additional load on the server and additional data traffic, so please do not use this method if not really necessary, this document contains information to allow calculate the hash on client side (see Appendix A.).
SOAP action: getglshash
Input: getglshashRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*
***password** type *string*	*- password – request it from GLS*
***senderid** type *string*	*- GLS client number – request it from GLS*
***sender_name** type *string*	*- sender name*
***sender_address** type *string*	*- sender address*
***sender_city** type *string*	*- sender city*
***sender_zipcode** type *string*	*- sender zip code*
***sender_country** type *string*	*- sender country code or country name*
sender_contact type *string*	*- sender contact person*
sender_phone type *string*	*- sender phone*
sender_email type *string*	*- sender email address*
***consig_name** type *string*	*- consignee name*
***consig_address** type *string*	*- consignee address*
***consig_city** type *string*	*- consignee city*
***consig_zipcode** type *string*	*- consignee zip code*
***consig_country** type *string*	*- consignee country code or country name*
consig_contact type *string*	*- consignee contact person*
consig_phone type *string*	*- consignee phone or SMS number for services*
consig_email type *string*	*- consignee email address – also used for services*
***pcount** type *int*	*- count of the parcels / labels to print*
***pickupdate** type *string*	*- pickup date in format yyyy-MM-dd*
content type *string*	*- content of the parcel – info printed on label*
clientref type *string*	*- client reference*
codamount type *decimal*	*- COD amount*
codref type *string*	*- COD reference – used if COD amount is set*
gapid type *string*	*- ONLY IN HU! if it is exists and valid, the function is ready to use*

Output:

getglshashResponse (soap:body, use = encoded)

return type *string*		*the hash code for the communication stability check*

deletelabels method description

Purpose: delete one or more GLS shipment data

SOAP action: deletelabels

Input: deletelabelsRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*
***password** type *string*	*- password – request it from GLS*
***senderid** type *string*	*- GLS client number – request it from GLS*
***pclids** type *ArrayOfString*	*- array of id parcels for deletion*
gapid type *string*	*- if it is exists and valid, the function is ready to use*

Output:

deletelabelsResponse (soap:body, use = encoded)

return type *deletelabels_result*

- **pcls** type *ArrayOfString* — *- return list of the parcel numbers deleted or not*
- **successfull** type *Boolean* — *- if false, errcode and errdesc will be set*
- **errcode** type *int* — *- numeric error code, please list in AppendixD*
- **errdesc** type *string* — *- error description*

- when variable “successfull” is TRUE that variable “pcls” contains deleted parcels
- when variable “successfull” is FALSE that variable “pcls” contains problematic parcels
- if one of list parcels isn't possible to delete, none parcels will be deleted

preparelabels_gzipped_xml method description

Purpose: with this method client is able to send batch label data to preparing for later printing. The request data parameter must be gzip compressed GLS DTU XML string.

Is possible use own parcel number range when was assigned by GLS. For this option use tag 'PclIDs' in XML (see Appendix G). Without this tag is assigned parcel number automatically by API.

SOAP action: preparelabels_gzipped_xml
Input: preparelabels_gzipped_xmlRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*
***password** type *string*	*- password – request it from GLS*
***senderid** type *string*	*- GLS client number – request it from GLS*
***data** type *base64Binary*	*- gzipped XML contains the data for label data preparation, for XML structure information please see the GLS DTU specification documentation*
gapid type *string*	*- if it is exists and valid, the function is ready to use*

For sample input XML and sample codes please see Appendix G

Output:

deletelabels_xmlResponse (soap:body, use = encoded)

return type *string*	*- the return XML will contains the deleted parcels ID or errors for each problematic parcel*

For sample output XML please see Appendix H

preparelabels_xml method description

Purpose: similar to the preparelabels_gzipped_xml method. The main difference is that the communication is not compressed. Because of the greater traffic need of this method it is recommended to use the preparelabels_gzipped_xml method instead.

SOAP action: preparelabels_xml

Input: preparelabels_xmlRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*
***password** type *string*	*- password – request it from GLS*
***senderid** type *string*	*- GLS client number – request it from GLS*
***data** type *string*	*- XML contains the data for label data preparation, for XML structure information please see the GLS DTU specification documentation*
gapid type *string*	*- if it is exists and valid, the function is ready to use*

Output:

preparelabels_xmlResponse (soap:body, use = encoded)

return type *string*	*- the returned XML will contains the stored parcel ID numbers which can be used for batch label printing later*

deletelabels_gzipped_xml method description

Purpose: With this method client is able to delete label data.

SOAP action: deletelabels_gzipped_xml

Input: deletelabels_gzipped_xmlRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*
***password** type *string*	*- password – request it from GLS*
***senderid** type *string*	*- GLS client number – request it from GLS*
***data** type *base64Binary*	*- gzipped XML contains the data for deleting parcels, for XML structure information please see the GLS DTU specification documentation*
gapid type *string*	*- if it is exists and valid, the function is ready to use*

For sample output XML please see Appendix K

Output:

deletelabels_xmlResponse (soap:body, use = encoded)

return type *string*	*- the returned XML will contains the stored parcel ID numbers which can be used for batch label printing later*

For sample output XML please see Appendix L

deletelabels_xml method description

Purpose: similar to the deletelabels_gzipped_xml method. The main difference is that the communication is not compressed. Because of the greater traffic need of this method it is recommended to use the deletelabels_gzipped_xml method instead.

SOAP action: deletelabels_xml

Input: deletelabels_xmlRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*
***password** type *string*	*- password – request it from GLS*
***senderid** type *string*	*- GLS client number – request it from GLS*
***data** type *string*	*- XML contains the data for deleting parcels, for XML structure information please see the GLS DTU specification documentation*
gapid type *string*	*- if it is exists and valid, the function is ready to use*

For sample output XML please see Appendix K

Output:

deletelabels_xmlResponse (soap:body, use = encoded)

return type *string*	*- the returned XML will contains the stored parcel ID numbers which can be used for batch label printing later*

For sample output XML please see Appendix L

getprintedlabels_gzipped_xml method description

Purpose: this method is used to retrieve the labels with the parcel IDs we got back from the preparation method before.

The return value is base64Binary, gzipped XML where the label information is within the DTU XML Label tag in Parcel in Pacels in Shipment in Shipments, in base64 encoded format.

SOAP action: getprintedlabels_gzipped_xml

Input: getprintedlabels_gzipped_xmlRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*	
***password** type *string*	*- password – request it from GLS*	
***senderid** type *string*	*- GLS client number – request it from GLS*	
***data** type *base64Binary*	*- gzipped XML contains the data for label printing, for XML structure information please see the GLS DTU specification documentation, and see the sample XML in Appendix I*	
***printertemplate** type *string*	*- type of the printer – list in Appendix B*	
is_autoprint_pdfs type *Boolean*	*- if false then js print() command will be omitted from the label pdfs*	
gapid type *string*	*- if it is exists and valid, the function is ready to use*	

Output:

getprintedlabels_gzipped_xmlResponse (soap:body, use = encoded)

return type *base64Binary - the returned gzipped XML will contains the pdf label in base64 encoded format*

getprintedlabels_xml method description

Purpose: similar to the getprintedlabels_gzipped_xml method. The main difference is that the communication is not compressed. Because of the greater traffic need of this method it is recommended to use the getprintedlabels_gzipped_xml method instead.

SOAP action: getprintedlabels _xml

Input: getprintedlabels_xmlRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*
***password** type *string*	*- password – request it from GLS*
***senderid** type *string*	*- GLS client number – request it from GLS*
***data** type *string*	*- XML contains the data for label printing, for XML structure information please see the GLS DTU specification documentation, and see the sample XML in Appendix I*
***printertemplate** type *string*	*- type of the printer – list in Appendix B*
is_autoprint_pdfs type *Boolean*	*- if false then js print() command will be omitted from the label pdfs*
gapid type *string*	*- if it is exists and valid, the function is ready to use*

Output:

getprintedlabels _xmlResponse (soap:body, use = encoded)

return type *string - the returned XML will contains the pdf label in base64 encoded format, for return XML sample see Appendix J*

getprintedlabels_zipped_pdfs method description

Purpose: this method is used to retrieve the labels with the parcel IDs we got back from the preparation method before.
In this case the return value is not XML but a zip archive what contains the pdf label files. The file are named <pclid>.pdfs. The input XML is compressed with gzip.
SOAP action: getprintedlabels_zipped_pdfs
Input: getprintedlabels_zipped_pdfsRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*
***password** type *string*	*- password – request it from GLS*
***senderid** type *string*	*- GLS client number – request it from GLS*
***data** type *base64Binary*	*- gzipped XML contains the data for label printing, for XML structure information please see the GLS DTU specification documentation, and see the sample XML in Appendix I*
***printertemplate** type *string*	*- type of the printer – list in Appendix B*
is_autoprint_pdfs type *Boolean*	*- if false then js print() command will be omitted from the label pdfs*
gapid type *string*	*- if it is exists and valid, the function is ready to use*

Output:

getprintedlabels_gzipped_pdfsResponse (soap:body, use = encoded)

return type *base64Binary - the returned zip archive will contains the pdf labels*

modifycod method description

Purpose: modify cod amount on existing parcel, which is already under delivery

SOAP action: modifycod

Input: modifycodRequest (soap:body, use = encoded)

***username** type *string*	*- user name – request it from GLS*
***password** type *string*	*- password – request it from GLS*
***senderid** type *string*	*- GLS client number – request it from GLS*
***pclid** type *string*	*- id of the parcel to modify cod*
***codamount** type *decimal*	*- new, changed amount of the cod*
***email** type *string*	*- email address where confirmation will be sent*
gapid type *string*	*- if it is exists and valid, the function is ready to use*

Output:

`modifycodResponse` (soap:body, use = encoded)

return type *standard_result*

- **successfull** type *boolean* — *- if false, errcode and errdesc will be set*
- **errcode** type *int* — *- numeric error code, please list in AppendixD*
- **errdesc** type *string* — *- error description*

- use 0 for codamount if you would like to storno COD service and not collect COD for selected parcel at all

Appendix A
List of services

Service code	Service name	Parameter
T12	Express Service	
PSS	Pick&Ship Service	Pickup data in format yyyy-MM-dd
PRS	Pick&Return Service	Pickup data in format yyyy-MM-dd
XS	Exchange Service	• it is necessary to print second label for return parcel
SZL	DocumentReturn Service	Document number – string, max. 15 char
INS	DeclaredValueInsurance Service	Value of the parcel
SBS	Standby Service	
DDS	DayDefinite Service	Date of delivery in format yyyy-MM-dd
SDS	ScheduledDelivery Service	Time range of delivery in format HH:mm-HH:mm
SAT	Saturday Service	
AOS	AddresseeOnly Service	Name of the recipient / contact person can be used
24H	Guaranteed24 Service	
EXW	ExWorks Service	
SM1	SMS Service	SMS Phone number and SMS text in format ”phone nr in international format|sms text”. Variable #ParcelNr# can be used for parcel number.
SM2	PreAdvice Service	SMS Phone number in international format
CS1	Contact Service	Recipient phone number / contact phone number can be used
TGS	ThinkGreen Service	
FDS	FlexDelivery Service	Email address
FSS	FlexDeivery SMS Service	SMS phone number in international format • not available without FDS
PSD	ShopDelivery Service	DropOffPoint ID
DPV	DeclaredParcelValue	Used in case of HR, 20xxx zip codes, to declare value of the parcel
ADR	HazardousGoods Service	Used in case of HU, if the ADR is required. The parameters: The mark and the ADR Code in the info line is, for example LQ2330 and the content line is the quantity, for example 1X5 kg. The separator is ’,’

- not all services are available in each country and each area
- in case of service PSS/PRS label is not printed, just service ordered

Appendix B
Types of the printer template

Type	Description
A6	A6 format, blank label
A6_PP	A6 format, preprinted label
A6_ONA4	A6 format, printed on A4
A4_2x2	A4 format, 4 labels on layout 2x2
A4_4x1	A4 format, 4 labels on layout 4x1
T_85x85	85x85 mm format for thermal labels

Appendix C
Hash code

- hash code is used to verify intergrity of the sent request
- **calculation**: join following fields content into string and use sha1() algorithm:
  username, password, senderid, sender_name, sender_address, sender_city, sender_zipcode, sender_country, sender_contact, sender_phone, sender_email, consig_name, consig_address, consig_city, consig_zipcode, consig_country, consig_contact, consig_phone, consig_email, pcount, pickupdate, content, clientref, codamout, codref
  (each input field without *services, printertemplate, printit, timestamp, hashcode*)

Sample PHP function:

```
functiongetHash($data) {
    $hashBase = '';
foreach($data as $key => $value) {
if ($key != 'services'
&& $key != 'hash'
&& $key != 'timestamp'
&& $key != 'printit'
&& $key != 'printertemplate'
          && $key != 'customlabel') {
            $hashBase .= $value;
        }
    }

return sha1($hashBase);
}
```

Sample c# function:

```
usingSystem.Security.Cryptography;
privatestring getHash()
{
       string expression =
              string.Format("{0}{1}{2}{3}{4}{5}{6}{7}{8}{9}{10}{11}{12}{13}{14}{15}{16}
              {17}{18}{19}{20}{21}{22}{23}{24}", user, pwd, senderid, senderName,
              senderAddress, senderCity, senderZipcode, senderCountry, senderContact,
              senderPhone, senderEmail, consigName, consigAddress, consigCity,
              consigZipcode, consigCountry, consigContact, consigPhone, consigEmail,
              pcount, pickupdate, content, clientref, codamount, codref));

       SHA1 sha1 =SHA1.Create();
       string hash = sha1.ComputeHash(System.Text.Encoding.UTF8.GetBytes(expression));
       return BitConverter.ToString(hash).Replace("-", "").ToLower();
}
```

Appendix D
Error codes

Type	Description
0	OK
1	Authentication failed
2	Invalid hash
3	Unable to store data please try again later
4	Invalid printer template please check documentation
5	Missing parameters:
6	Invalid timestamp
7	Invalid sender country
8	Invalid consignee country
9	Invalid sender zipcode
10	Invalid consignee zipcode
11	Invalid pickup date
12	Parcel count must be between 1 and 99
13	Missing contact person for export parcel
14	COD is not allowed to this export country
15	Max value for COD is:
16	Invalid COD rounding the smallest fraction is
17	Invalid service code(s):
18	Invalid service combination(s):
19	Service(s) not available in pickup country:
20	Service(s) not available between sender and consignee country:
21	Service(s) not available on consignee country/zipcode:
22	Invalid / missing parameters for service(s):
23	FSS service is valid only with ordered FDS service
24	For parcels to HR with zipcode 20xxx please use DPV parameter info to send declared parcel value for parcel
25	Same request sent 5 times within last 5 minutes!
26	Requested parcel not belongs to the user!
27	Wrong XML format!
28	One or more parcels is impossible to delete.
29	No parcels to deleting.
30	Parcel was deleted before this request.
31	Parcel number was not found.
32	You can send only one parcel with ADR service in order.
33	You can send only one parcel with ADR service in order.
34	You can't use the same parcel number more times.

35	Count of parcels and inserted numbers are different.
36	Parcel number has a bad format.
37	The parcel number isn't assigned to your sender ID.
38	The parcel number was already used.
39	You don't have necessary rights to use the following sender id:
40	Missing contact phone for consignee (%s service)
41	Invalid parcel number
42	Missing COD amount
43	COD amount is less than zero
44	COD amount is greater than maximum value
45	Invalid e-mail address
46	Process failed:

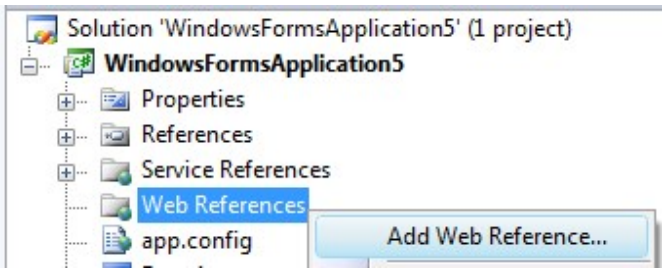
Appendix E
NuSOAP client sample code in PHP

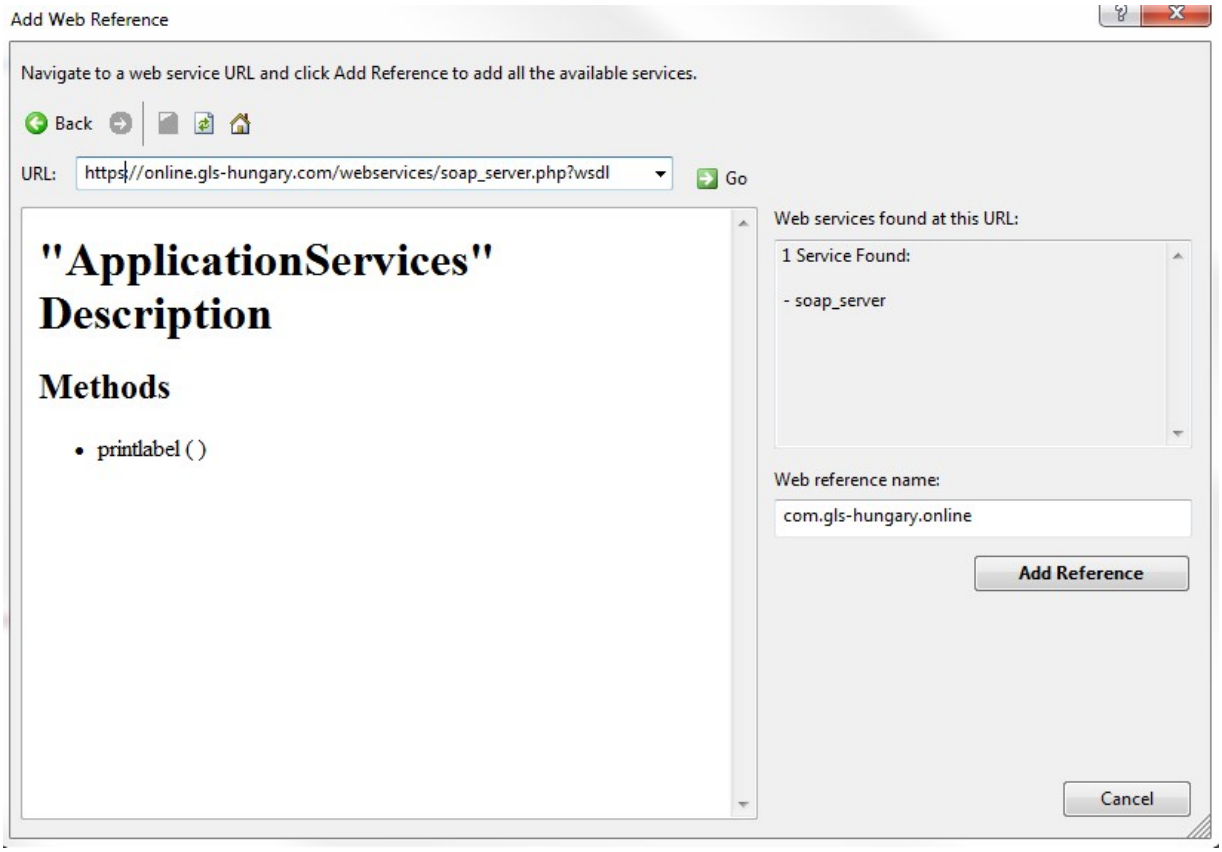
```
<?php
require_once('lib/nusoap.php');
$_HTTP = !empty($_SERVER['HTTPS']) ? 'https://' : 'http://';
$wsdl_path =
$_HTTP.$_SERVER['HTTP_HOST'].'/webservices/soap_server.php?wsdl&ver=14.05.20.01';
$client = new nusoap_client($wsdl_path,'wsdl');
$client->soap_defencoding = 'UTF-8';
$client->decode_utf8 = false;

$in = array(
            'username' => 'username',
            'password' => 'password',
            'senderid' => '000000000',
            'sender_name' => 'sendername',
            'sender_address' => 'senderaddress',
            'sender_city' => 'sendercity',
            'sender_zipcode' => '2351',
            'sender_country' => 'HU',
            'sender_contact' => 'sendercontact',
            'sender_phone' => 'senderphone',
            'sender_email' => 'valaki@valaki.sk',
            'consig_name' => 'consigname',
            'consig_address' => 'consigaddress',
            'consig_city' => 'consigcity',
            'consig_zipcode' => '2351',
            'consig_country' => 'HU',
            'consig_contact' => 'consigcontact',
            'consig_phone' => 'consigphone',
            'consig_email' => 'valaki@valaki.sk',
            'pcount' => 1,
            'pickupdate' => '2014-01-29',
            'content' => 'tartalom',
            'clientref' => 'clientref',
            'codamount' => '1526',
            'codref' => 'codref',
            'services' => array(),
            'printertemplate' => 'A4_2x2',
            'printit' => true,
            'timestamp' => '20140129150000',
            'hash' => 'xsd:string',
       'customlabel' =>false );

$in['hash'] = getHash($in);

$return = $client->call('printlabel', $in);
if ($return) {
if ($return['successfull']) {
header('Content-type: application/pdf');
die(base64_decode($return['pdfdata']));
    } else {
var_dump($return);
    }
}
```

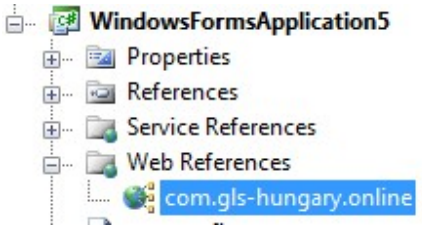
```
functiongetHash($data) {
    $hashBase = '';
foreach($data as $key => $value) {
if ($key != 'services'
&& $key != 'hash'
&& $key != 'timestamp'
&& $key != 'printit'
              && $key != 'printertemplate'
              && $key != 'customlabel') {
             $hashBase .= $value;
        }
    }

return sha1($hashBase);
}

?>
```

Appendix **F**
Handling pdfdata from result in c# (Visual Studio 2008)

Adding reference to project:







Use added namespace to consume methods from web reference:

```
com.gls_hungary.online.ApplicationServices svc = new com.gls_hungary.online.ApplicationServices();
try
{
    var result = svc.printlabel(user,
                    pwd,
                    senderid,
                    senderName,
                    senderAddress,
                    senderCity,
                    ...
```

Example of processing results:

```
byte[] data = Convert.FromBase64String(result.pdfdata);
System.IO.File.WriteAllBytes(@"c:\temp\label.pdf", data);
```

Appendix G
Sample XML for prepareLabels input data

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<DTU EmailAddress="testadmin@gls-hungary.com" Version="15.04.18.01" Created="2015-04-18T10:17:27.0088665+01:00" RequestType="GlsApiRequest" MethodName="prepareLabels">
  <Shipments>
    <Shipment SenderID="100000000" ExpSenderID="" PickupDate="2015-04-20T00:00:00+01:00" ClientRef="2015/0418/001" CODAmount="5" CODCurr="HUF" CODRef="20150418001" PCount="1" >
      <From Name="From Name 1" Address="From address 1" ZipCode="2351" City="Alsónémedi" CtrCode="HU" ContactName="Contact name" ContactPhone="Contact phone" EmailAddress="testfrom@gls-hungary.com" />
      <To Name="To Name 1" Address="To address 1" ZipCode="1111" City="Budapest" CtrCode="HU" ContactName="Contact name" ContactPhone="Contact phone" EmailAddress="testto@gls-hungary.com" />
      <PclIDs>
          <long>12345678901</long>
      </PclIDs>
      <Services>
        <Service Code="FDS">
          <Info>
            <ServiceInfo InfoType="INFO" InfoData="testfds@gls-hungary.com" />
          </Info>
        </Service>
      </Services>
    </Shipment>
    <Shipment SenderID="100000000" ExpSenderID="" PickupDate="2015-04-20T00:00:00+01:00" ClientRef="2015/0418/002" CODAmount="0" CODCurr="HUF" CODRef="20150418002" PCount="1" Info="Megjegyzés" >
      <From Name="From Name 2" Address="From address 2" ZipCode="2351" City="Alsónémedi" CtrCode="HU" ContactName="Contact name" ContactPhone="Contact phone" EmailAddress="testfrom@gls-hungary.com" />
      <To Name="To Name 2" Address="To address 2" ZipCode="1112" City="Budapest" CtrCode="HU" ContactName="Contact name" ContactPhone="Contact phone" EmailAddress="testto02@gls-hungary.com" />
    </Shipment>
  </Shipments>
</DTU>
```

sample_API_request_preparelabels.xml

Find the sample codes below to prepare labels, communication is compressed with gzip.

PHP sample codes:

```php
require_once('lib/nusoap.php');

$_HTTP = !emptyempty($_SERVER['HTTPS']) ? 'https://' : 'http://';
$wsdl_path = $_HTTP.$_SERVER['HTTP_HOST'].'/webservices/soap_server.php?wsdl&ver=15.04.18.01';
$client = new nusoap_client($wsdl_path,'wsdl');
$client->soap_defencoding = 'UTF-8';
$client->decode_utf8 = false;

preparelabels_gzipped_xml($client,true);

function preparelabels_gzipped_xml($client, $download_result = false)
{
    $data = '<?xml version="1.0" encoding="UTF-8" ?><DTU ... > ... </DTU>';
    // XML omitted to save space, see sample file, and replace it with xml generator code

    $in = array(
        'username' => 'user',
        'password' => 'pass',
        'senderid' => '100000000',
        'data' => base64_encode(gzencode($data,9))
    );

    $return = $client->call('preparelabels_gzipped_xml', $in);
    $result = gzdecode($return);

    if($download_result)
    {
        header('Content-type: text/xml');
        header('Content-Disposition: attachment; filename="result.xml"');
        die($result);
    }

    return $result;
}
```

C# sample codes:
Notice that in case of C# we used a prepared object _user, _pwd, and _senderid fields was set in the constructor on object creation, details are omitted for the sake of simpleness.

```
using System.IO;
using System.IO.Compression;


protected string gzUnZip(byte[] data)
{
    if (data == null || data.Length == 0)
        return null;

    using (MemoryStream ms = new MemoryStream(data))
    using (GZipStream gzip = new GZipStream(ms, CompressionMode.Decompress))
    using (StreamReader sr = new StreamReader(gzip))
        return sr.ReadToEnd();
}

protected byte[] gzZip(string data)
{
    if (data == null || data.Length == 0)
        return null;

    byte[] compressed;

    using (var outStream = new MemoryStream())
    {
        using (var tinyStream = new GZipStream(outStream, CompressionMode.Compress))
        using (var mStream = new MemoryStream(Encoding.UTF8.GetBytes(data)))
            tinyStream.Write(mStream.ToArray(), 0, mStream.ToArray().Length);

        compressed = outStream.ToArray();
    }

    return compressed;
}

public string preparepelabels_gzipped_xml(string xml_data)
{
    ApplicationServices api = new ApplicationServices();

    byte[] bytes = gzZip(xml_data);
    byte[] returnValue = api.preparelabels_gzipped_xml(_user, _pwd, _senderid, bytes);
    return gzUnZip(returnValue);
}
```

Appendix H

Sample XML for prepareLabels output data

```xml
<?xml version="1.0"?>
<DTU Version="15.04.18.01" EmailAddress="it@gls-hungary.com" Created="2015-04-19T19:03:57+02:00" RequestType="GlsApiResult" MethodName="prepareLabels">
    <Shipments>
        <Shipment ClientRef="2015/0418/001">
            <Status ErrorCode="0" ErrorDescription="">success</Status>
            <PclIDs><long>00000000001</long></PclIDs>
        </Shipment>
        <Shipment ClientRef="2015/0418/002">
            <Status ErrorCode="0" ErrorDescription="">success</Status>
            <PclIDs><long>00000000002</long></PclIDs>
        </Shipment>
    </Shipments>
</DTU>
```

sample_API_result_preparelabes.xml

Appendix I
Sample XML for getprintedlabels_gzipped_xml input data

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<DTU Version="15.04.18.01" EmailAddress="test@gls-hungary.com" Created="2015-04-18T11:15:00:00+01:00" RequestType="GlsApiRequest" MethodName="printLabels" >
    <Shipments>
        <Shipment>
            <PclIDs>
                <long>1</long>
            </PclIDs>
        </Shipment>
        <Shipment>
            <PclIDs>
                <long>2</long>
            </PclIDs>
        </Shipment>
    </Shipments>
</DTU>
```

sample_API_request_getprintedlabels.xml

Find the sample codes below to prepare labels, communication is compressed with gzip.

PHP sample codes:

Please take for consideration the next qutation:

"Since version 2.7.3 libxml limits the maximum size of a single text node to 10MB.
The limit can be removed with a new option, XML_PARSE_HUGE.
PHP has no way to specify this option to libxml."

This problem can be avoided if the user can create a custom xml parser, or . The standard GLS label size is about 1 MB. So with the base64 encoding and other parameters this means that with standard xml parser the requests should be limited to 8 labels.
Or find a way to supply the LIBXML_PARSEHUGE constant to the xml parser:
http://svn.php.net/viewvc/?view=revision&revision=291533

```php
require_once('lib/nusoap.php');

$_HTTP = !emptyempty($_SERVER['HTTPS']) ? 'https://' : 'http://';
$wsdl_path = $_HTTP.$_SERVER['HTTP_HOST'].'/webservices/soap_server.php?wsdl&ver=15.04.18.01';
$client = new nusoap_client($wsdl_path,'wsdl');
$client->soap_defencoding = 'UTF-8';
$client->decode_utf8 = false;

getprintedlabels_gzipped_xml($client,true);

function getprintedlabels_gzipped_xml($client, $download_result = false)
{
    $data = '<?xml version="1.0" encoding="UTF-8" ?><DTU ... > ... </DTU>';
        // XML omitted to save space, see sample file, and replace it with xml generator code

    $in = array(
        'username' => 'user',
        'password' => 'pass',
        'senderid' => '100000000',
        'data' => base64_encode(gzencode($data,9)),
        'printertemplate'=>'A6_PP',
        'is_autoprint_pdfs' => false
    );

    $return = $client->call('getprintedlabels_gzipped_xml', $in);

    $result = gzdecode($return);

    if($download_result)
    {
        header('Content-type: text/xml');
        header('Content-Disposition: attachment; filename="result.xml"');
        die($result);
    }

    return $result;
}
```

Difference when we requesting zipped pdfs, that the result is the .zip file content, so just have to save ot use it directly az a zip archive, no need for gunzip or xml parse (of course there are less error signing options too).

C# sample codes:

```
1.    public string getprintedlabels_gzipped_xml()
2.    {
3.        ApplicationServices api = new ApplicationServices();
4.
5.        byte[] bytes = gzZip(_xml_data);
6.        byte[] result = api.getprintedlabels_gzipped_xml(_user, _pwd, _senderid, bytes, _printertemplate.ToString(), _is_autoprinting_pdfs);
7.        return gzUnZip(result);
8.    }
9.
10.   public byte[] getprintedlabels_zipped_pdfs()
11.   {
12.       ApplicationServices api = new ApplicationServices();
13.
14.       byte[] bytes = gzZip(_xml_data);
15.       byte[] result = api.getprintedlabels_zipped_pdfs(_user, _pwd, _senderid, bytes, _printertemplate.ToString(), _is_autoprinting_pdfs);
16.       return result;
17.   }
```

Appendix J
Sample XML for getprintedlabels_gzipped_xml output data

```xml
<?xml version="1.0"?>
<DTU Version="15.04.18.01" EmailAddress="it@gls-hungary.com" Created="2015-04-19T21:38:30+02:00" RequestType="GlsApiResult" MethodName="printLabels">
    <Shipments>
        <Shipment>
            <Status ErrorCode="0" ErrorDescription="">success</Status>
            <Parcels>
                <Parcel PclId="1">
                    <Label>JVBERi0xLjcKO ... further content is too long so omitted,
                                it should be base64 decode and use the content as a PDF file ... </Label>
                </Parcel>
            </Parcels>
        </Shipment>
    </Shipments>
</DTU>
```

sample_API_result_getprintedlabels.xml

Appendix K
Sample XML for deletelabels_xml input data

```
<?xml version="1.0" encoding="UTF-8"?>
- <DTU MethodName="deleteLabels" RequestType="GlsApiRequest" Created="2017-01-00T13:00:00+01:00" Version="16.12.15.01" EmailAddress="email@address.com">
  - <Shipments>
    - <Shipment>
      - <PclIDs>
          <long>1234567</long>
        </PclIDs>
      </Shipment>
    - <Shipment>
      - <PclIDs>
          <long>12345678910</long>
        </PclIDs>
      </Shipment>
  </Shipments>
</DTU>
```

Appendix L
Sample XML for deletelabels_xml output data

```
<?xml version="1.0" encoding="UTF-8"?>
- <DTU MethodName="deleteparcelRequest" RequestType="GlsApiResult" Created="2017-01-01T20:00:00+01:00" Version="16.12.15.01" EmailAddress="email@address.com">
  - <Shipments>
    - <Shipment>
        <Status ErrorDescription="" ErrorCode="0">success</Status>
      - <Parcels>
          <Parcel PclId="1234567"/>
        </Parcels>
      </Shipment>
    - <Shipment>
        <Status ErrorDescription="Requested parcel not belongs to the user!" ErrorCode="26">failed</Status>
      - <Parcels>
          <Parcel PclId="12345678910"/>
        </Parcels>
      </Shipment>
  </Shipments>
</DTU>
```

Appendix M
DropOffPoint ID request

As receiver address the destination parcelshop address has to be printed to label (source: downloaded xml file). Authorized person for picking up the parcel must be included as contant

List of the actual parcelShops should be downladed / updated from:

https://datarequester.gls-hungary.com/glsconnect/getDropoffPoints.php?ctrcode=HU&updated=20130912120000

where ctrcode = country code, updated = timestamp of the last update from last downloaded xml file

- result is always encoded (gzencode)
- when current xml is up to date, (encoded) 1 is returned
- authentication: user: cliusr, password: cliusr1
- xml file has to be stored offline and updated max. once / day