# INTERNETMARKE (INTERNET STAMP)

## Technical Service Description - Quick Reference Guide

### 1C4A web service

### Web service for third-party providers

Document classification in accordance with the IT Security Manual (SHB IT):

<**open** | for internal use only | confidential | strictly confidential>

**Version: 2.0/Last revised: December 16, 2022**

We have taken great care in the production of this technical Service Description and are providing it to you free of charge to help you with the development of interfaces to our systems. However, we shall not be liable and give no guarantee for the provided information being up-to-date, correct, and complete.

# Table of contents

# 1 About this Document

## 1.1 Purpose

This document conveys the basic information required for integrating the OneClickForApplication web service (1C4A) in third-party service providers. The 1C4A web service can be tested based on the minimal example in Chapter *4 Actions* from page 9.

## 1.2 Target group

This document is intended for the developers at third-party service providers, who wish to integrate 1C4A in their own applications.

## 1.3 Service version

This document relates to service version 3 of the 1C4A web service.

**Note: The technical designators for the respective actions are used in this document.**

## 1.4 Other documents

| Document | Stored under |
|---|---|
| Webservice 1C4A_Information Package for Partners | ZIP file:<br>DP-Webservice 1C4A Partneranbindung INTERNETMARKE [DP Web Service 1C4A Partner Connection INTERNET STAMP] |
| Technical WEB Service Definition (WSDL) | https://internetmarke.deutschepost.de/OneClickForAppV3?wsdl |
| Product web service<br>(ZIP file with all content on the ProdWS) | ZIP file:<br>DP-Webservice 1C4A Partneranbindung INTERNETMARKE [DP Web Service 1C4A Partner Connection INTERNET STAMP] |

## 1.5 Further help

The detailed illustration of the 1C4A interface is provided in the document “INTERNETMARKE – Technical Service Description.” If you have any questions that are not clarified in these two documents, please e-mail the Support team (pcf-1click@deutschepost.de).

# 2 Functional operation

The 1C4A web service is used to sell Internetmarke stamps that are generated in the INTERNETMARKE application of Deutsche Post AG. The INTERNETMARKE is generated via an individual postage account assigned to the customer.

The 1C4A web service consists of three blocks of actions:

- User and postage account management
- Internetmarke generation
- Provision of information

## 2.1 User and postage account management

In this quick reference guide, only the “AuthenticateUser” action from the “User and postage account management” block is described. The “Technical Service Description” reference document contains the remaining actions with their relevant explanations.

| Action | Technical designator | Brief description |
|---|---|---|
| Authenticate user | AuthenticateUser | This action authenticates the customer or the customer’s postage account and authorizes subsequent queries to use the postage account linked to the e-mail address. |

## 2.2 Internetmarke generation

This Kurzanleitung deals only with the variant of Internetmarke generation in PDF format. In addition to PDF format, you can also use PNG as the file format for Internetmarke generation. You can find a brief description in the table below. The detailed description is provided in reference document “INTERNETMARKE – Technical Service Description.”

| Action | Technical designator | Brief description |
|---|---|---|
| Create PNG Internetmarke stamps | CheckoutShoppingCartPDF | The action validates the shopping cart and the payment.<br>The action debits the costs for the transaction to the postage account and returns a URL to a PDF file which contains the Internetmarke stamps. |
| Create PNG Internetmarke | CheckoutShoppingCartPNG | The action is comparable to the CheckoutShoppingCartPDF action, but instead of a link to a PDF, delivers a link to a ZIP file with illustrations of the purchased frankings in PNG format. |

## 2.3 Provision of information

The action in the following table is optional. You can find further optional actions in the reference documentation “INTERNETMARKE – Technical Service Description.”

| Action | Technical designator | Brief description |
| --- | --- | --- |
| Query the valid print formats | retrievePageFormats | The action is used to request valid print formats that have been created in the INTERNETMARKE application. |

# 3 Prerequisites for communication with the third-party system

## 3.1 Conditions for the SOAP request

The 1C4A interface expects a SOAP request on the basis of the HTTPS transport protocol. For all requests,

- SSL encryption must be enabled at transport level and
- a third-party provider-specific SOAP header and signature must be specified.

## 3.2 Form of target address

In live operation, the web service is addressed via the following URLs: *https://internetmarke.deutschepost.de/OneClickForAppV3*

## 3.3 Structure of the SOAP request

For each action, optional and mandatory fields are specified for the request and must be provided in the SOAP body. Whether the request is correct in terms of structure and contains all mandatory elements is checked for every request through SchemaValidation.

All functional error types and their functional error codes are described in the WSDL. All errors are returned embedded in a SOAP fault element, with functional errors being defined in advance as types in the WSDL.

Technical errors are returned to the service recipient in the form of Java runtime errors.

The service recipient can rely on the fact that it is always the error codes defined in the WSDL that are returned if the functional error type contains an ID element. With specific error types that reflect only a single functional error situation, the ID element is dispensed with.

**Note: The service recipient should avoid checking text contents of Message elements in functional error types, since these can be changed without notice.**

Before a SOAP request is forwarded to the business logic of the 1C4A interface for processing, it is validated against the WSDL schema. The schema validator checks whether the structure of the request matches the schema and whether it contains all mandatory elements. If the request does not meet the criteria, a SchemaValidationException SOAP fault is returned.

## 3.4 Structure of the header

For each SOAP request, the 1C4A web service expects the following header.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v3="http://oneclickforapp.dpag.de/V3 ">
   <soapenv:Header>
      <v3:PARTNER_ID>DPPAR</v3:PARTNER_ID>
      <v3:REQUEST_TIMESTAMP>28012014-142729</v3:REQUEST_TIMESTAMP>
      <v3:KEY_PHASE>1</v3:KEY_PHASE>
      <v3:PARTNER_SIGNATURE>58578270</v3:PARTNER_SIGNATURE>
   </soapenv:Header>
   <soapenv:Body>
   <-- Nutzinformationen zugunsten der Verständlichkeit entfernt -->
```

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

The following header elements must be sent in every request. The header elements can be arranged in any sequence.

| Parameter | Description | Type | Length | Values from the example |
|---|---|---|---|---|
| PARTNER_ID | The mail order partner ID must be configured in the application, otherwise the request is rejected. | String | 5 | DPPAR |
| REQUEST_TIMESTAMP | This parameter contains the send time of the request with date and time in the format DDMMYYYY-HHMMSS. | String | 15 | 05032021-142729 |
| KEY_PHASE | The parameter contains the agreed version of the secret. | Integer | 3 | 1 |
| PARTNER_SIGNATURE | The parameter contains the signature of the request, which is calculated via the fields defined. | String | >=8 | 58578270 |

## 3.5 Checksum calculation

### 3.5.1 Manual checksum calculation

The signature is calculated as an SHA512 hash on the basis of the contents of the following fields.

- PARTNER_ID
- REQUEST_TIMESTAMP
- KEY_PHASE
- SIGNATURE KEY
  Length 32 characters; The content is static and is communicated to the Deutsche Post AG partner.

To create the checksum, the contents of the above fields are appended. All field contents have any blanks removed at the left and right and are concatenated with the “::” separator.

The outcome is used as input for SHA512. The bytes in their hexadecimal representation (with a–f in lower case letters) are converted to text. Hexadecimal representations of bytes with just one decimal (0–f) are expanded with a leading zero (00–0f).

The resulting text is stored in the PARTNER_SIGNATURE field.

At the recipient, the same steps are to be performed to verify the PARTNER_SIGNATURE.

#### 3.5.2 Automated checksum calculation

With the following Unix command, tested on RHEL 7 in a Bash-Shell, you can calculate the checksum. Replace the command parts marked in orange with your individual customer data.

```
echo -n "DPPAR::28012022-142729::1::Das_Ist_der_neue_Key" | sha512sum | awk '{print $1}'
```

These are:

| | |
| --- | --- |
| **DPPAR** | the ID for third-party providers |
| **28012022-142729** | the current request date and time (e.g., from the command date --date='TZ="Europe/Berlin" now' +%d%m%Y-%H%M%S) |
| **1** | the version number of the secret |
| **Das_Ist_der_neue_Key** | the secret |

The request date and the time are variable here.

**Note: In the Annex, there is a programming example for checksum calculation in Java.**

# 4 Actions

This chapter contains a functional minimal example with the “AuthenticateUser” and “CheckoutShoppingCartPDF” actions. Replace the code parts marked in orange with your individual customer data. The code parts marked in blue are transferred from the response to the next request.

## 4.1 AuthenticateUser

### 4.1.1 Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v3="http://oneclickforapp.dpag.de/V3">
      <soapenv:Header>
            <v3:PARTNER_ID>DPPAR</v3:PARTNER_ID>
            <v3:REQUEST_TIMESTAMP>28012014-142729</v3:REQUEST_TIMESTAMP>
            <v3:KEY_PHASE>1</v3:KEY_PHASE>
            <v3:PARTNER_SIGNATURE>58578270</v3:PARTNER_SIGNATURE>
       </soapenv:Header>
       <soapenv:Body>
            <AuthenticateUserRequest xmlns="http://oneclickforapp.dpag.de/V2">
                  <username>max.mustermann@deutschepost.de</username>
                  <password><![CDATA[vH$OkF*z40]]></password>
            </AuthenticateUserRequest>
       </soapenv:Body>
</soapenv:Envelope>
```

| Parameter | Description |
| --- | --- |
| password | Password assigned when the INTERNETMARKE user was registered |
| username | E-mail address used for user registration |

The remaining parameters as well as permitted values and data types can be found in the reference documentation “INTERNETMARKE – Technical Service Description.”

### 4.1.2 Response

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Header/>
   <S:Body>
      <AuthenticateUserResponse xmlns="http://oneclickforapp.dpag.de/V3">
         <userToken>5NOi3VHvC9PukGUkSzTYsHL/JrMgG8+15BAY/8wgG5CSj6tzy1jJmg==</userToken>
         <walletBalance>67424</walletBalance>
         <showTermsAndConditions>false</showTermsAndConditions>
      </AuthenticateUserResponse>
   </S:Body>
</S:Envelope>
```

| Parameter | Description |
|---|---|
| showTermsAndConditions | License agreement |
| userToken | Token to confirm authentication and to authorize the use of a postage account for the purchase |
| walletBalance | Credit balance on the postage account in eurocents after the purchase or at the time of authentication |

The remaining parameters as well as permitted values and data types can be found in the reference documentation “INTERNETMARKE – Technical Service Description.”

## 4.2 CheckoutShoppingCartPDF

### 4.2.1 Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v3="http://oneclickforapp.dpag.de/V3">
      <soapenv:Header>
            <v3:PARTNER_ID>DPPAR</v3:PARTNER_ID>
            <v3:REQUEST_TIMESTAMP>28012014-142729</v3:REQUEST_TIMESTAMP>
            <v3:KEY_PHASE>1</v3:KEY_PHASE>
            <v3:PARTNER_SIGNATURE>58578270</v3:PARTNER_SIGNATURE>
       </soapenv:Header>
   <soapenv:Body>
      <v3:CheckoutShoppingCartPDFRequest>

<v3:userToken>5NOi3VHvC9PukGUkSzTYsHL/JrMgG8+15BAY/8wgG5CSj6tzy1jJmg==</v3:userToken>
         <v3:pageFormatId>1</v3:pageFormatId>
         <v3:positions>
            <v3:productCode>1</v3:productCode>
            <v3:voucherLayout>AddressZone</v3:voucherLayout>
            <v3:position>
               <v3:labelX>1</v3:labelX>
               <v3:labelY>1</v3:labelY>
               <v3:page>1</v3:page>
            </v3:position>
         </v3:positions>
         <v3:total>70</v3:total>
      </v3:CheckoutShoppingCartPDFRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

| Parameter | Description |
|---|---|
| labelX | Specifies the X-position (column) on the page |
| labelY | Specifies the Y-position (row) on the page |
| page | Number of the page on which the stamp is located |

| Parameter | Description |
|---|---|
| pageFormatId | ID of the print format used for printing |
| position | Positioning of the franking mark on the PDF document (page, row, and column) |
| positions | Shopping cart positions |
| productCode | ID that identifies the product within the product and price list |
| total | Total value of the shopping cart in eurocents |
| userToken | Token to confirm authentication and to authorize the use of a postage account for the purchase |
| voucherLayout | Intended positioning of the franking on the letter (address window or franking zone), determines the look of the franking mark |

The remaining parameters as well as permitted values and data types can be found in the reference documentation “INTERNETMARKE – Technical Service Description.”

### 4.2.2 Response

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Header/>
   <S:Body>
      <CheckoutShoppingCartPDFResponse xmlns="http://oneclickforapp.dpag.de/V3">

<link>https://internetmarke.deutschepost.de/PcfExtensionWeb/document?keyphase=1&amp;data=OR
wfSjJTHkSUF4dj6mEfahg3St8HEBbf</link>
         <walletBallance>67354</walletBallance>
         <shoppingCart>
            <shopOrderId>718474626</shopOrderId>
            <voucherList>
               <voucher>
                  <voucherId>A0011B1B9D0000008C71</voucherId>
               </voucher>
            </voucherList>
         </shoppingCart>
      </CheckoutShoppingCartPDFResponse>
   </S:Body>
</S:Envelope>
```

| Parameter | Description |
| --- | --- |
| link | URL that refers to a PDF file containing the purchased franking marks |
| shopOrderId | Order number |
| shoppingCart | Shopping cart |
| voucherList | List of franking IDs (voucher ID) |
| walletBalance | Credit balance on the postage account in eurocents after the purchase or at the time of authentication |

The remaining parameters as well as permitted values and data types can be found in the reference documentation “INTERNETMARKE – Technical Service Description.”

# 5 Notes

## 5.1 Programming example for checksum calculation in Java

```
/*
* Copyright(c) 2000-2022 Deutsche Post DHL Group. All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
*  1. Redistributions of source code must retain the above copyright notice,
*     this list of conditions and the following disclaimer.
*  2. Redistributions in binary form must reproduce the above copyright
*     notice, this list of conditions and the following disclaimer in the
*     documentation and/or other materials provided with the distribution.
*  3. Neither the name of the copyright holder nor the names of its
*     contributors may be used to endorse or promote products derived from
*     this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*/
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Deutsche Post DHL Group - Internetmarke.
 *
 * Example of an SHA512 signing / hashing algorithm, to generate the hashes to
 * be enclosed in the Internetmarke interface SOAP headers.
 *
 * You may freely take the code for your Internetmarke handling stubs.
 */
public class SHA512Sample {
    /** Logging output for the hashing algorithm. */
    private static final Logger logger = Logger.getAnonymousLogger();

    /** The identification of the SHA512 hashing algorithm. */
```

```
public static final String SHA512 = "SHA-512";
/** Encoding style for the DPDHL Internetmarke. */
public static final String ENCODING_STYLE = "UTF-8";
/** Separator character between the fields to encode / hash. */
public static final String FIELDS_SEPARATOR = "::";
/** Maximum string size for the fields to be hashed. */
public static final int MAX_FIELD_STRING_SIZE = 256;

/**
 * Main function to start the example algorithm from the command line.
 * @param args Command line arguments.
 */
public static void main(String args[]) {

   DateFormat formatter = new SimpleDateFormat( "ddMMyyyy-HHmmss" );
   String theDate = formatter.format( new Date() );

   // String theDate = "28012014-142729"; // set a fixed date as an example

   // Example: To be replaced by the partner id provided by Deutsche Post
   String thePartnerID = "DPPAR";
   // Example: To be replaced by the secret provided by Deutsche Post
   String thePartnerSecret = "Das_Ist_der_neue_Key";
   // Example: To be replaced by the key phase provided by Deutsche Post
   String theKeyPhase = "1";

   SHA512Sample sha512Sample = new SHA512Sample();

   // The main part, hashing of the given felds.
   byte[] hash = sha512Sample.sign(new String[] {
         thePartnerID, theDate,theKeyPhase },
         thePartnerSecret);

   if (hash != null) {
      String theSignature = sha512Sample.hashToString(hash);

      // Debugging output...
      logger.info("Hash signature (as string): " + theSignature);
   }
}

/**
 * Signs a given set of parameter fields.
 *
 * @param theFields Set of fields that are included in the signature and
 * are to be signed.
 * @param theSignatureKey Signature key, added to the list of fields.
 * @return byte[] Signature of the field set.
 */
public byte[] sign(String[] theFields, String theSignatureKey) {

   if (theFields != null && theFields.length > 0) {
```

```
      // Prepare a string buffer to assemble fields string
      StringBuffer fieldsString = new StringBuffer(MAX_FIELD_STRING_SIZE);

      // move all fields to be signed into string buffer with separators
      for (int i = 0; i < theFields.length; i++) {
         if (theFields[i] != null) {
            fieldsString.append(theFields[i].trim());
         }
         fieldsString.append(FIELDS_SEPARATOR);
      }
      fieldsString.append(theSignatureKey);

      // Debugging output....
      logger.info("String representation of fields: " + fieldsString.toString());

      try {

         // Get the hashing algorithm
         MessageDigest theDigest = MessageDigest.getInstance(SHA512);
         theDigest.reset();

         // hash the fields string
         theDigest.update(fieldsString.toString().getBytes(ENCODING_STYLE));

         return theDigest.digest();

      } catch (NoSuchAlgorithmException nsae) {
         logger.log(Level.SEVERE, nsae.toString(), nsae);
      } catch (UnsupportedEncodingException uee) {
         logger.log(Level.SEVERE, uee.toString(), uee);
      }
   }

   // Some sort of error occurred, just return null
   return null;
}

/**
 * Converts the byte hash into a string representation
 * (human-readable), Base16 encoding truncated to 8 characters.
 * @param hash The hash (byte string) to be converted.
 * @return String String representation of the hash.
 */
public String hashToString(byte[] hash) {

   StringBuffer hashStringRepresentation = new StringBuffer();

   for (int i = 0; i < hash.length; i++) {
      String hexByteString = Integer.toHexString(0xFF & hash[i]);
      if (hexByteString.length() == 1) {
         hashStringRepresentation.append("0");
```

```
          }
          hashStringRepresentation.append(hexByteString);
       }

       return hashStringRepresentation.toString();
    }
}
```

## 5.2 Glossary

| Term | Description |
| --- | --- |
| 1C4A | OneClickForApplication, Deutsche Post AG web service designed to integrate INTERNETMARKE franking |
| Third-party provider | In the narrower sense, a third-party provider refers to the marketplaces and partners that have integrated the INTERNETMARKE application via the web service described here. |
| INTERNETMARKE | INTERNETMARKE is a Deutsche Post AG application for the purchase of Internetmarke stamps. |
| Internetmarke (Internet stamp) | The term Internetmarke is used in the context of partner systems. Internetmarke stamps are franking marks that were generated by the INTERNETMARKE system. |
| Customer | A user who wishes to purchase franking marks. |
| Stamp | In this document, stamp is a synonym for Internetmarke stamp. |
| Postage account | PORTOKASSE [postage account] is a prepaid wallet that can be used to buy Internetmarke stamps or other products of Deutsche Post AG with previously topped up amounts of money. |
| Shipping list | Documentation of shipments with additional national and international services for the sender |