Interface description

mobilepay

Version: **2.10**

Date: **29.06.2017**

4Pay Networks GmbH
Fischerstraße 49
D - 40477 Düsseldorf

Geschäftsführer: Martin Kolisch
HRB79650, Amtsgericht Düsseldorf
Steuernummer 103/5751/2620

Commerzbank Düsseldorf
IBAN DE91 3004 0000 0151 5154 00
BIC COBADEDDXXX

Change History

Version	Date	Author	Description
1.0	15.09.2004	Claus Leonhardt	Released version 1.0
1.1	09.10.2004	Claus Leonhardt	Introduced request *stopsubscription*
1.2	12.11.2004	Claus Leonhardt	Added field amount in requests bill- and refund Added field *callbackurl* in request *smsauthorize*
1.3	19.11.2004	Claus Leonhardt	Introduced request *getmno* Documented restrictions with billing in subscription transactions
1.4	01.03.2005	Claus Leonhardt	WAP billing: added parameter *nabyuser* to indicate missing authorization of payment by the user to call to the errorURL
1.5	04.05.2005	Claus Leonhardt	Added optional parameters *txt1*, *txt2*, *txt3* to authorization request
1.6	01.09.2005	Claus Leonhardt	Added error message *LIMIT_EXCEED*
	11.10.2005	Claus Leonhardt	Introduced Easy Billing
1.7	02.03.2006	Claus Leonhardt	Introduced Bulk Billing
	06.03.2006	Claus Leonhardt	Introduced option for returning more verbose result of billing operation
1.8	14.02.2007	Jürgen Brardt	Introduced MSISDN lookup
1.9	31.08.2007	Andreas Pritzlaff	Review
1.9	11.12.2007	Nils Runge	Review and extensions
1.9	05.02.2008	Nils Runge	Documented specifics for different countries, formatting
2.0	22.05.2008	Lars Korbel	Formatting
2.0	30.06.2008	Nils Runge	Reworked chapter 5 in preparation of introduction of new billing types
2.1	17.09.2009	Nils Runge	Documented technical changes in authorization response Review and corrections
2.2	30.12.2009	Nils Runge	Added parameter *mccmnc* for authorization request
2.2	21.10.2010	Henrieta Kaniewski	Review and formatting
2.3	30.12.2010	Nils Runge	Introduced *Downscaling* and *Stop Subscription Callback*
2.4	18.04.2011	Daniel Sedlack	Review and formatting
2.5	26.05.2011	Daniel Sedlack	Introduced additional status codes
2.6	11.06.2012	Michael Suthe	Introducing the parameters *description*, *gtc*, *imprint*, *contact* and *faq* for Mobile Internet Payment, introducing restrictions for 3rd-party panel templates.
2.7	01.01.2014	Henrieta Kaniewski	Review and formatting

Version	Date	Author	Description
2.8	06.08.2014	Henrieta Kaniewski	Review and formatting
2.9	14.06.2017	Christoph Berger	Reworked chapter 4.1.3. Added chapter 4.1.3.2 “Web Billing with redirect” 4.2 Added parameter *msisdn, mcc, mnc* to Table 1 Response: Bill Removed former chapter 7.1 “Technical Prerequisites“ WAP Billing renamed as *Mobile Billing* Review and formatting
2.10	29.06.2017	Tarek El-Sibay	Fixed parameter names in msisdn service
	29.06.2017	Christoph Berger	Removed Easy Billing Review and formatting

1 Introduction

Many mobile network operators (MNOs) allow their users to make purchases on third party platforms which will be charged to their mobile phone bill. Third parties can get access to this service through 4Pay Networks which is connected to the relevant MNO's billing interfaces, required for using this payment method.

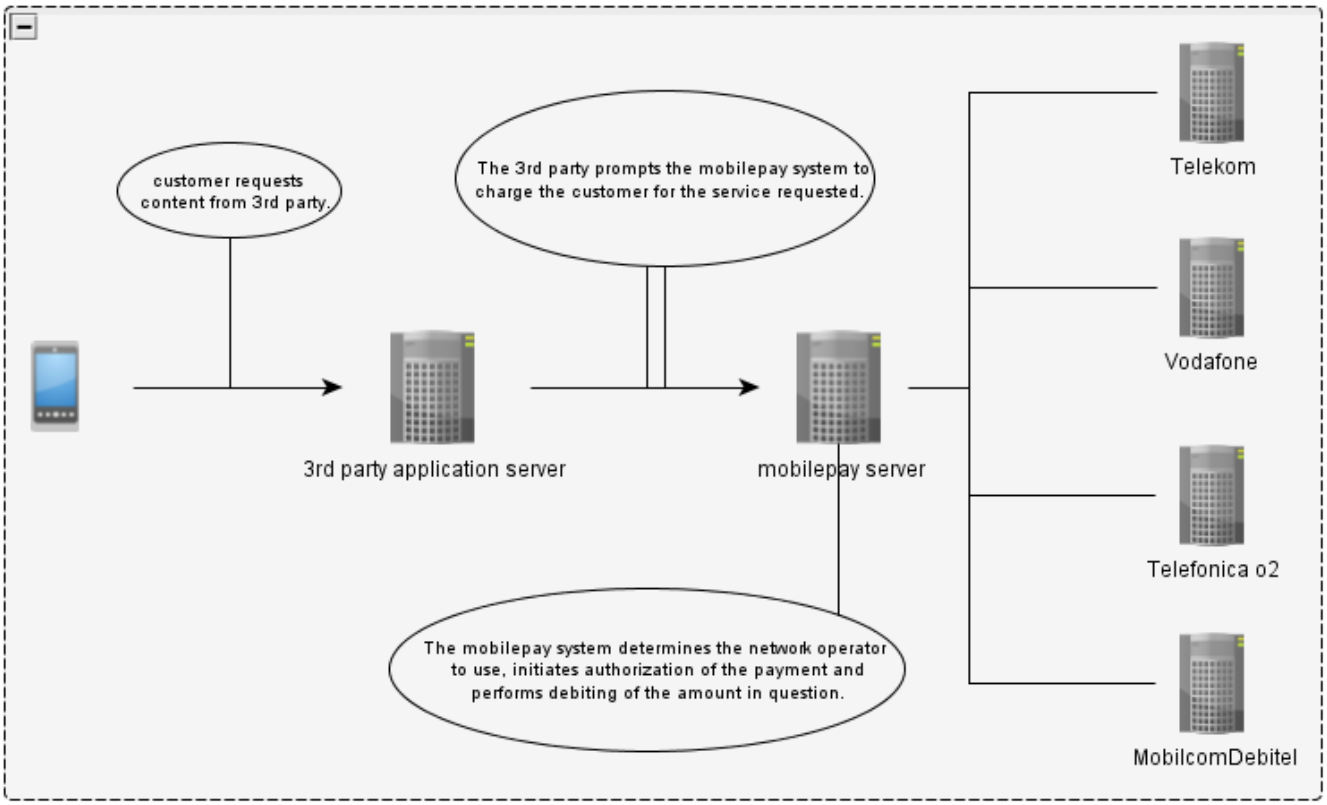


Figure 1 Structure mobilepay (Germany)

From a technical point of view these interfaces vary widely between operators. In addition to this, a contractual agreement between the third party and each operator is required to be able to use these interfaces. Furthermore, prior to each payment transaction, the customer's network operator needs to be determined to select the correct connection to be used. This requires access to a number lookup service, like the *Zentrale Master Routing Datenbank* (zMRDB) in Germany, that in turn requires additional effort in implementing the payment solution and entails additional costs for the third party.

4Pay Networks GmbH ("4Pay") has developed a unified mobile payment solution system called *mobilepay*. With mobilepay you can perform micropayment transactions for all German mobile network providers as well as some foreign ones through one single interface. All existing technical distinctions between the network operators will be handled by 4Pay. As an additional benefit, there is no need for you to sign agreements contracts with each operator directly as the contract with 4Pay already provides access to mobile subscribers in the relevant countries.

The workings of *mobilepay* are displayed schematically in Figure 1. Each payment transaction is being performed by the contracting party via *mobilepay*. That system will automatically determine the subscriber's mobile network operator and employ the connection suitable to initialise the actual billing. In this process, the contracting party will solely be required to communicate with the *mobilepay* system while receiving feedback concerning the payment transaction's progress.

All information needed for using the *mobilepay* interface will be discussed in this document.

For Premium SMS services, please use the PSMS interface description in Annex 1.

2 Functions

The *mobilepay* system offers three different billing variants:

- *SMS Billing*
- *Mobile Billing*
- *Web Billing*

All three of these have in common the ability to either authorize a **single** payment or a series of multiple payments in a **subscription**. When using a single-transaction, the customer authorizes the payment for a service or product requested by her. Hence, she can only be billed once per transaction. In contrast to this, subscriptions offer the capability to perform repeated billings in one transaction after authorising only once. Further information on subscriptions can be found in chapter 2.2.2.

2.1 Billing Variants

2.1.1 SMS Billing

2.1.1.1 Standard Billing

When using SMS Billing, the subscriber orders a service or product by sending an SMS to a free short code or entering her MSISDN on a website. In return, an SMS is sent informing her of the resulting costs and asking her to confirm the transaction by replying to the message just received.

2.1.2 Web Billing

Web Billing is a billing variant based on the customer requesting a product or service by entering her MSISDN on the contracting party's website. In return, a PIN will be sent by SMS which the subscriber is required to enter on the website in order to validate the payment transaction.

2.1.3 Mobile Billing

Mobile Billing (former WAP Billing) is a third billing variant. It requires the subscriber to be in possession of a mobile internet providing device she uses to request a product or service. A confirmation page is sent to her and she is asked to validate the payment transaction directly on that page. The actual billing may then be performed right after the successful completion of this process.

2.2 Billing Types

Three different types of billings exist: single billing, subscription and cumulative billing. They are being explained in the following paragraphs.

2.2.1 Single Billing

When using a single billing transaction, the customer may only be billed once. This is a two-step process consisting of authorization and the subsequent debiting of the customer. After the debiting was performed, the transaction is closed. If additional debits are to be performed, new transactions need to be created for each one.

2.2.2 Subscription

In contrast to single billing transactions, subscriptions (*multiple billings*) allow the contracting party to perform several debits in a single transaction.

Analogue to the flow of single billing transactions, subscriptions work as a two-step process. In the first phase, authorization has to be performed analogous to the single billing. In difference to the authorization performed when using single billing, the authorization's type needs to be set to *multiple* to indicate that a subscription is to be authorised (see chapter 4.1). The subscriber will be explicitly notified about this fact in the validation request sent by SMS (see chapter 2.2.2.1). After successful validation, the contracting party will be able to debit the customer according to the boundaries determined in the contractual details set for the respective service without performing authorization anew.

However, take note of the fact that *mobilepay* only serves as a mediator between the contracting party and the mobile network operators. As a direct result of this, the system does not provide functionality for managing subscriptions. Therefore, this needs to be implemented on the side of the contracting party. Not only does this concern the subscription’s maturity and eventual termination (if applicable), but also keeping track of the points in time when billings are to be performed and the amounts to be debited. This means that the contracting party's system is required to handle event and time-based subscriptions in accordance to the billing service's description. For each billing performed the contracting party may bill an amount equal or less than the amount originally specified in the authorization. Also, the contracting party is obligated to terminate subscriptions after reaching maturity or on request by the subscriber. Furthermore, it is part of the contracting party's obligations to ensure conformity to any existing regulatory rules concerning the service being run.

In addition to this, network operators may exist that enforce the subscription's service description by technical means. This, however, does not release the contracting party from the obligations stated above.

2.2.2.1 Authorization Handshake

Subscriptions require the subscriber to confirm having taken note of the service's description and its pricing. It is mandatory to inform the customer about the name of the company providing the service, the product's name, the service's price and the interval of billing.

2.2.2.2 Welcome SMS

After performing the authorization handshake for the service, the contracting party is liable to inform the subscriber by SMS about the name of the company providing the

service, a hotline number for contacting said company and the terms of the subscription's termination.

Sending welcome SMS is done provider-dependent by either the network operator or the *mobilepay* system. In the latter case, it is possible for the contracting party to individualize the texts sent upon consultation with 4Pay. Further details can be found in chapter 4.8.

2.2.2.3 Stop Subscription Callback

If a subscription is terminated using the *stopsubscription* command (see chapter 4.6), mobilepay will inform a third party about this event using an HTTP request as a callback. This request will contain the transaction ID (TXID) and the service's name as POST parameters. To receive this callback, the URL, it is to be performed on, needs to be supplied with the authorization request as parameter *stopsubcallbackurl.*

2.2.2.4 Change of Terms

Changing the terms of a service (billing interval, pricing) requires the customers of existing subscriptions to renew their approval.

2.2.2.5 Types of Subscriptions

It can be differentiated between event- and time-based subscriptions and combinations of both. When using an event-based subscription, a billing can be triggered by the occurrence of certain events like the subscriber requesting content. When using time-based subscriptions, billings will always be performed in a regular interval.

2.2.3 Cumulative Billing

Cumulative billing is an extension of the single billing mechanism. After successful authorization, the amount to bill can be increased repeatedly in a short span of time before the actual billing is performed using the final amount (see chapter 4.5).

One scenario for using this feature is services billing the customer for a certain time interval (i.e. a voice-based service billed every minute). The subscriber's balance (in case of *pay as you go*) will be checked for each increase. After the last increase the complete amount may be billed by executing the *bill* command. In contrast to using the subscription mechanism, this leads to only one billing position appearing on the customer's invoice while using a subscription would lead to each increase appearing as a single billing. Furthermore, unlike subscriptions cumulative billings can only have one billing request executed per transaction and a transaction is considered finished after the billing has been executed.

2.2.4 Miscellaneous

2.2.4.1 Downscaling

The *Downscaling* mechanism provides the option of retrying a previously failed billing with a lower amount of money to bill. This opens the possibility of using up the entire balance on a pay as you go SIM card.

There are several different methods currently supported by *mobilepay* for calculating the reduced amount.

- Multiplication with a factor (for instance 0.5)
- Subtraction of a fixed amount (for instance 50 Cent)
- Combinations of 1 and 2
- Trying to bill a series of fixed amounts (i.e. 300 Cent, 200 Cent, 150 Cent)

Downscaling can be activated and configured for each service individually upon consultation with 4Pay.

3 Technical Implementation

For each billing variant, the contracting party has to abide by the given flow. All variants have in common the need for authorization by the customer prior to any other operation being executed. The authorization's technical specifics vary between billing variants. They are explained in detail in chapter 4.1. Authorization is a two-step process. First, the authorization request has to be sent to *mobilepay*. This request will trigger a check whether or not the designated amount is available to be debited. This check is especially relevant in the case of pay as you go subscribers whose balance may not be sufficient for the billing to be performed. If it is not possible to bill the designated amount, *mobilepay* will report this using its interface. If, however, the check is successful, the subscriber needs to validate the payment's execution. It is only after the successful validation and having rendered the service requested that the contracting party may execute the actual billing request.

Communication between the contracting party's system and the *mobilepay* server takes place using a standard HTTP or HTTPS connection, the latter being preferred due to security reasons. This enables the implementation of the *mobilepay* system's *Application Programming Interface* (API) using well-established tools and procedures. If this is, however, not desired, 4Pay provides its customers with a ready-to-use Java-based client that may be included into the contracting party's own application. This client is delivered containing JSP-based example applications for using Mobile (WAP), Web and SMS Billing.

3.1 HTTP Interface

When using the HTTP(S) interface directly, either HTTP-GET or HTTP-POST requests may be used for executing commands. The response's HTTP body consists of an XML structure containing the different fields and respective values belonging to the response. These structures are being discussed in detail in the following chapters.

The level of detail for each response may be influenced by using the optional request parameter *detail*. If this value is set to *true,* the response will contain the additional field *bookingmessage*. This in turn contains extended information in a non-specified format like status messages sourced from the billing providers directly. An example for a response without extended information is given in XML 1. Another example for a response including the *bookingmessage* field is shown in XML 2.

```
<result>
   <statustext>OK</statustext>
   <statuscode>100</statuscode>
</result>
```

XML 1 Example: Response with detail=false

```
<result>
   <statustext>OK</statustext>
   <statuscode>100</statuscode>
   <bookingsmessage>D1: OK</bookingsmessage>
</result>
```

XML 2 Example: Response with detail=true

3.2 Java Client

The Java client contains classes enabling the use of all the available billing variants. It is possible to use single billing transactions as well as subscriptions. In addition, the client also supports so called *Bulk Billing,* facilitating the billing of a larger set of subscribers simultaneously (see chapter 4.3).

Central to the client's usage are the class *AbstractMobilePaymentClient* and its concrete implementations of which there are four, representing the different billing variants (see chapter 2.1):

- *SmsPaymentClient*
- *WAPPaymentClient*
- *WebPaymentClient*

These clients may be instantiated by using their respective constructors and can then be used to execute the appropriate billing commands. For each command there is a corresponding method named after the command's designation as established in chapter 4, prefixed with *do* (i.e. *doAuthorize()*, *doBill()*). Furthermore, a method named *setDetail()* exists that may be used to activate detailed responses.

As a result, an object of a class derived from *AbstractCommandResult* is returned. Depending on the command executed, these result objects may contain different values which can be extracted using the respective getters. In general, *statusCode* and *statusText* will be available. All other fields may potentially contain the value *null*.

The methods for executing commands may throw a *TechnicalProblemException* if technical problems arise while executing the command. If this is the case, the message contained in the exception will explain these circumstances in greater detail.

Further details for using the client can be taken from the *Javadoc* documentation belonging to the client itself.

4 Operations

The *mobilepay* system provides access to a set of operations for conducting payment transactions. These may be used in the manners described in chapter 3.

A typical transaction consists of two phases: authorization and billing. The authorization phase is needed for ensuring that the billing is actually possible. Depending on the type of billing used, a validation of the transaction by the subscriber may be required. After successful authorization and having rendered the service requested, the contracting party may execute the billing request.

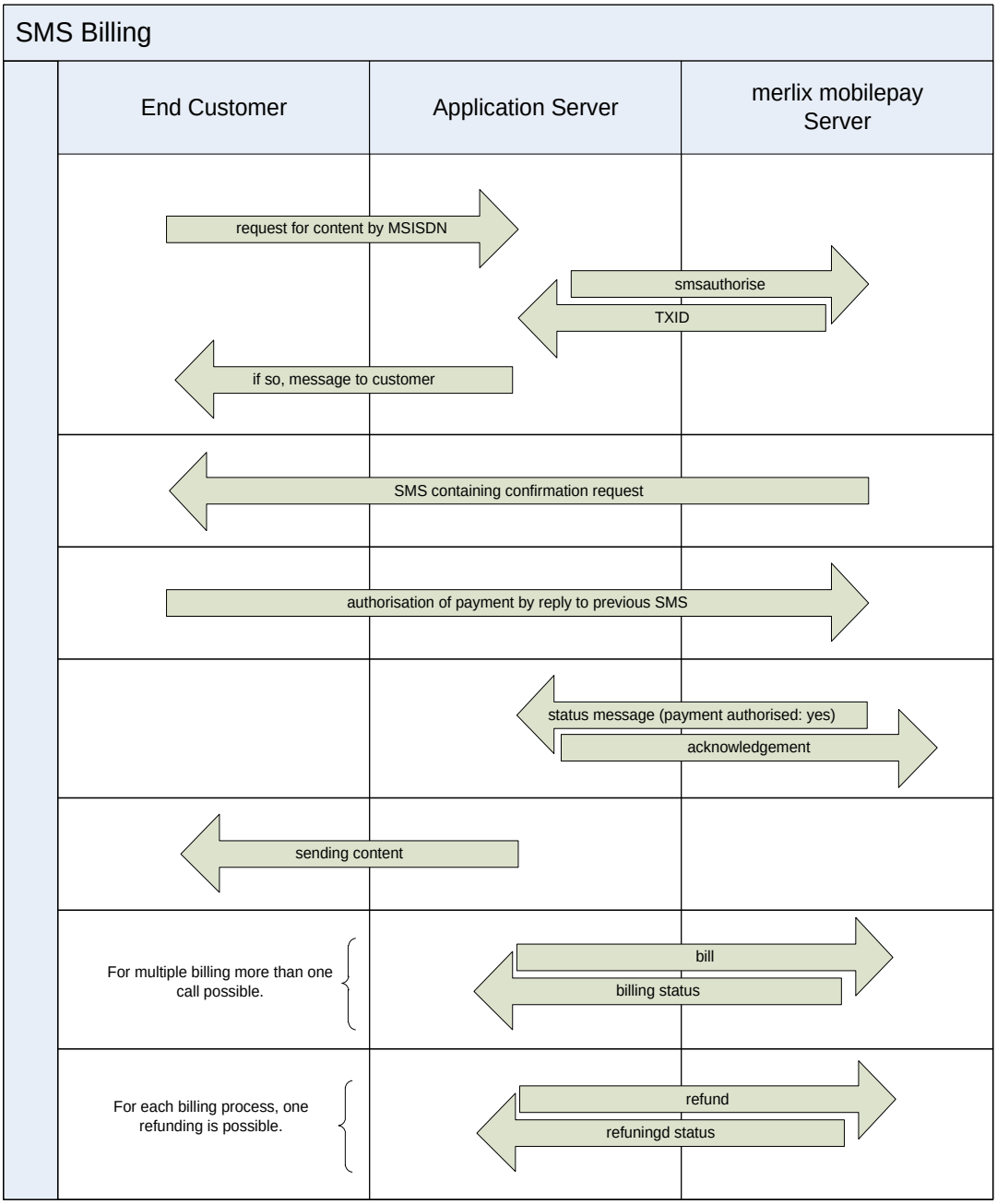
The operations *mobilepay* provides are described in the following sections.

4.1 Authorization

Authorization is the mandatory first step of each billing transaction. Its purpose is to ensure the requirements for performing the subsequent billing are met, reserve the amount of money to be debited and ask for the subscriber's validation if necessary. However, the authorization's exact flow differs between billing variants (see chapter 2.1).

4.1.1 Standard SMS Authorization

Standard SMS authorization is a procedure that consists of SMS-based validation of the payment transaction by the subscriber. To this end, the contracting party creates a new payment transaction by sending an HTTP request to the *mobilepay* system containing the *smsauthorize* command. Thereupon, an SMS will be sent to the subscriber informing her about the requested service's costs and asking her to validate the payment by replying to the SMS just received. That message is either being sent by 4Pay (for Telekom) or the network operators (for Vodafone and o2). The request flow is being displayed in *Figure 2*. After having received the confirmation message, the *mobilepay* system will inform the contracting party about the transaction's successful authorization using a callback HTTP request. The contracting party may then, after having rendered the service requested, perform the actual billing (*bill*).



4.1.1.1 Request Parameters

The parameters required for the HTTP request starting the authorization are listed in Table 2. The parameters contained in the HTTP response sent by *mobilepay* can be seen from Table 3.

Parameter	Description
command	***smsauthorize***
servicename	The name of the billing service to be used for the payment transaction. It will be chosen by 4Pay and communicated to the contracting party.
password	The password belonging to the service. It will be chosen by 4Pay and communicated to the contracting party.
amount	The gross amount the customer is billed. The amount has to be given in the smallest possible unit of currency. The currency used is set to a fixed value on a per-service basis. For instance, *300* (Cent) needs to be sent for invoicing a customer 3 Euro.
type	The type of billing to perform (see chapter 2.2) Values available: {***single, multiple, cumulative***} When using single billing, exactly one debiting may take place per authorised transaction. Multiple billings, on the other hand, allow for several debits in a single transaction. Cumulative billings enable increasing the amount repeatedly before the debiting is affected.
msisdn	The MSISDN to bill
mccmnc	*Mobile Country Code* (MCC) and *Mobile Network Code* (MNC) of the mobile network operator, if known Format: nnn-nn or nnnnn, e.g. 262-01 or 26201 for Telekom (262=MCC, 01=MNC).
callbackurl	The URL to call using an HTTP request after authorization finished successfully. That request contains the transaction's ID (*txid*) and the service's name (*servicename*).
stopsubcallbackurl	The URL to call using an HTTP request after a subscription has been terminated. That request contains the transaction's ID (txid) and the service's name (servicename) as POST parameters (see chapter 2.2.2). Only used for subscriptions.
txt1, txt2, txt3	Texts used to substitute the variables in the confirmation SMS to send (see chapter 4.8).
detail	Setting this optional parameter to ***true*** enables more verbose results (e.g. the parameter *bookingsmessage*) returned in the request's direct response. Values available: {***true***, ***false***}

Table 2 Request: SMS Authorize

Parameter	Description
statuscode	A numerical code indicating the status of the request's result (see chapter 5).
statustext	A text describing the status code in text form (see chapter 5).
txid	A unique ID identifying the transaction. It is generated automatically by *mobilepay* for each transaction.
bookingmessage	Additional information (e.g. error message directly from the network operators) describing the result in greater detail. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
provider	Name of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
mcc	*Mobile Country Code* of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
mnc	*Mobile Network Code* of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
mtid	ID of the confirmation SMS sent. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Table 3 Response: SMS Authorize

4.1.1.2 Example: SMS Billing via Short Code

The code snippets under Code 1 demonstrate how SMS Billing can be implemented when the customer's orders are being received using a short code.

```java
package sandbox;

import de.4Pay.client.mobilepay.SmsPaymentClient;
import de.4Pay.client.mobilepay.MobilePaymentException;
import de.4Pay.client.mobilepay.TechnicalProblemException;

/**
 * This is a simple demonstration how SMS Billing may be implemented. The
 * processes may be seen from the comments on each individual method.
 * In practical scenarios this process should incorporate use of
 * the callback request sent by mobilepay. That callback is being
 * executed once the subscriber has validated the payment transaction
 * by replying to the confirmation SMS sent to her.
 * Only after successful validation the bill request may be executed.
 *
 */
public class SmsBillingDemo {

   SmsPaymentClient smsPaymentClient = new SmsPaymentClient("PaymentServer-Url",
                                                            "servicename", "password");
   String msisdn = "491721234567";
   int amount = 199;
   boolean multiple = false;
   String myCallbackUrl = "http://www.mycallbackserver.de/mycallbackcontext";
   String txid;
   boolean authorized = false;

   /**
    * An exemplary process for SMS Billing
    *
    * @param args
    */
   public static void main(String[] args) {
      try {
         SmsBillingDemo smsBillingDemo = new SmsBillingDemo();
         System.out.println("Starting Authorization");
         smsBillingDemo.start();
         System.out.println("Authorization successful."
                            + "First attempt to bill in 30 seconds");
```

```
        try {
            Thread.sleep(30000);
        } catch (InterruptedException e1) {
        }
        smsBillingDemo.finish();
        System.out.println("Billing successfully completed.");
    } catch (TechnicalProblemException e) {
        System.err.println("Technical problem.");
        e.printStackTrace();
    } catch (MobilePaymentException e) {
        System.err.println("Billing not possible.\n status code: " + e.getStatusCode()
            + " status text:" + e.getStatusCode());
        e.printStackTrace();
    }
}

/**
 * Executes the SMS Authorize command.
 * A second methods of name <i>authorize</i> exists, having an
 * additional parameter, that may be used to pass a callback URL
 * that will be called once the subscriber has replied to the
 * confirmation SMS.
 *
 * <code>txid = smsPaymentClient.authorize(msisdn, amount, multiple,
 *                                                    myCallbackUrl);</code>
 *
 * @throws TechnicalProblemException
 * @throws MobilePaymentException
 */
public void start() throws TechnicalProblemException, MobilePaymentException {

    txid = smsPaymentClient.authorize(msisdn, amount, multiple);

}

/**
 * Calls {@link SmsBillingDemo#finish()} and checks whether or not validation
 * has already been performed. If it has not been performed, the method will
 * retry after 1 minute. Will give up after 10 tries.
 *
```

```
     * @throws TechnicalProblemException
     * @throws MobilePaymentException
     */
    public void finishWithRetry() throws TechnicalProblemException,
                                                                MobilePaymentException {
        int tries = 0;
        boolean retry = true;
        while (tries++ < 10 && retry) {
            try {
                finish();
                retry = false;
            } catch (MobilePaymentException e) {
                if(!e.getStatusCode().equals("410")) {
                    throw e;
                }
            }

            System.out.println("Billing not yet authorised. Waiting 1 minute....");
            try {
                Thread.sleep(60000);
            } catch (InterruptedException e1) {
            }
        }
    }

    /**
     * Executes the billing.
     * May only be successful after the subscriber has already validated the
     * transaction by replying to the confirmation SMS.
     *
     * @throws TechnicalProblemException
     * @throws MobilePaymentException
     */
    public void finish() throws TechnicalProblemException, MobilePaymentException {
        smsPaymentClient.bill(txid);
    }
}
```

Code 1 Example code SMS Billing

4.1.1.3 Example: SMS Billing via Web Application

This example demonstrates how SMS Billing may be used if the customer places an order using a website. The page shown in JSP 1 serves as an entry point where the customer starts the payment process.

```
<html>
   <head>
      <title>SMS Billing Test</title>
   </head>
   <body >
      <table border="1" cellspacing="0" cellpadding="5" align="center">
         <tr>
            <td>
               <img src="mv.jpg"/>
               <br/><br/>
               <b>Please enter your mobile subscriber number below to pay 6 Euro.</b>
               <br/><br/>
               <form method="GET" action="validate.jsp">
                  <input name="msisdn" type="text" /> 
                  <input type="submit" value="Bezahlen" />
               </form>
            </td>
         </tr>
      </table>
   </body>
</html>
```

JSP 1 index.jsp

The customer initiates the payment process for 6 Euro and is being redirected to a second page (JSP 2). On this page, the authorization by *mobilepay* is begun in the background. The platform will send the authorization SMS to the customer containing a description of the service ordered and a request to validate the payment by replying to the message just received.

```jsp
<%@ page import="de.4Pay.client.mobilepay.TechnicalProblemException" %>
<%@ page import="de.4Pay.client.mobilepay.MobilePaymentException" %>
<%@ page import="de.4Pay.client.mobilepay.SmsPaymentClient" %>
<%
  String url = application.getInitParameter("PaymentServer");
  String service = application.getInitParameter("Service");
  String pw = application.getInitParameter("Password");
  String msisdn = request.getParameter("msisdn");
  String text = "";
  String refresh = "";
  String txid = "";

   try {
      SmsPaymentClient smsp = new SmsPaymentClient(url,service,pw);
      txid = smsp.authorize(msisdn, 600, false);
      text = "Next you will receive an SMS."
              + "Please confirm this payment by replying via SMS";
      refresh = "<meta http-equiv=\"refresh\" content=\"2; URL=./content.jsp?txid="
                 + txid + "&amp;retry=10\" />";
   } catch (MobilePaymentException e) {
      text = "Unfortunately the payment could not be made: " + e.getStatusText();
   } catch (TechnicalProblemException e1) {
      text = "The payment could not be made due to technical issues";
   }
%>
<html>
   <head>
      <title>SMS Billing Test</title>
<%
      out.println(refresh);
%>
   </head>
   <body>
      <table border="1" cellspacing="0" cellpadding="5" align="center">
         <tr>
            <td>
<%
               out.println(text);
%>
            </td>
         </tr>
      </table>
   </body>
</html>
```

JSP 2 validate.jsp

After the validation phase the subscriber will be forwarded to a third page (JSP 3). On this page, the actual debiting will be performed. In case the subscriber has already validated the transaction and the debiting is successful, the content paid for will be displayed. If the transaction has not been validated yet, debiting is tried again every two seconds for ten times.

```
<%@ page import="de.4Pay.client.mobilepay.TechnicalProblemException" %>
<%@ page import="de.4Pay.client.mobilepay.MobilePaymentException" %>
<%@ page import="de.4Pay.client.mobilepay.SmsPaymentClient" %>
<%
   String url = application.getInitParameter("PaymentServer");
   String service = application.getInitParameter("Service");
   String pw = application.getInitParameter("Password");
   String txid = request.getParameter("txid");
   String text = "";
   String refresh = "";
   int I = new Integer(request.getParameter("retry")).intValue();

   try {
      SmsPaymentClient smsp = new SmsPaymentClient(url, service, pw, 10, 2000);
      smsp.bill(txid);
      text = "<img src=\"./bild.jpg\"/>";
   } catch (MobilePaymentException e) {
      if (e.getStatusCode().equals("410")) {
         if (i > 0) {
            i--;
            text = "The payment has not been authorised yet. Trying again in 2 seconds."
                    + i + " tries left.";
            refresh = "<meta http-equiv=\"refresh\" content=\"2; URL=./content.jsp?"
                       + txid=" + txid + "&amp;retry=" + I +"\" />";
         } else {
            text = "You have not confirmed the payment yet.";
         }
      } else {
         text = "Unfortunately the payment could not be made: " + e.getStatusText();
      }
   } catch (TechnicalProblemException e1) {
      text = ""The payment could not be made due to technical issues.";
   }
%>
<html>
   <head>
      <title>Web Billing Test</title>
<%
      out.println(refresh);
%>
   </head>
   <body>
      <table border="1" cellspacing="0" cellpadding="5" align="center">
         <tr>
            <td>
<%
               out.println(text);
%>
            </td>
         </tr>
      </table>
   </body>
</html>
```

JSP 3 content.jsp

4.1.1.4 Example HTTPS Request

https://www.mobilepay.de/mobilepay/mobilepay?command=smsauthorize&servicename=MyService&password=MyPassword&msisdn=4917112345678&callbackurl=http://www.myserver.com/callback.php&amount=300&type=single&txt1=&txt2=txt2&txt3=txt3&detail=true

4.1.2 Web Authorization

Web Billing is a method based on the customer entering a PIN code on a website to validate the payment. That PIN will be sent to her by SMS after the authorization process has been started.

There are two different flows to validate the PIN, depending on the service and mobile network operator:

- with a redirect to a web site of a MNO or
- without redirect, on a web site of the contracting party

If the *webauthorize* response contains an URL, the redirect flow must be used.

4.1.2.1 Web Authorization process without redirect

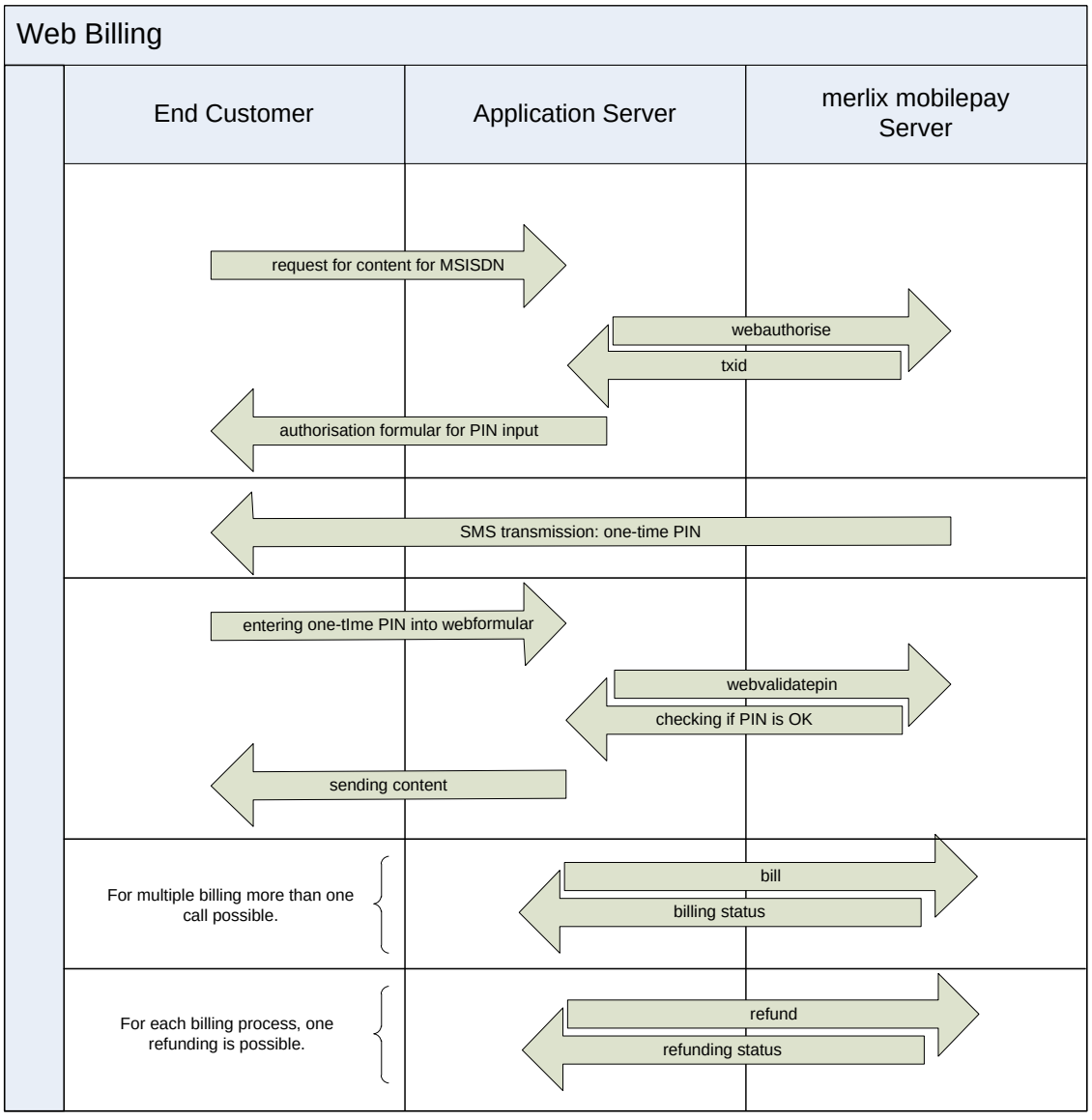


Figure 3 Process: Web Billing

This process is illustrated in Figure 3. First, the customer requests a service. Hereon, the contracting party executes a call to *mobilepay* (*webauthorize*) in order to initiate the authorization process. Afterwards, an SMS containing the PIN code is sent to the subscriber. She then is required to validate the payment by entering the PIN in a form on a webpage belonging to the contracting party. Afterwards, the contracting party needs to send this value to the *mobilepay* system using the (*webpinvalidate*) command to finish the validation step. The *mobilepay* system will confirm the successful validation and the contracting party may affect the debiting after having rendered the service requested.

For some services, the customer is redirected to webpage belonging to mobile network operators

4.1.2.2 Web Authorization process with redirect

This process for Web Billing will be based on a redirect (HTTP 302) of the customer. The customer will be redirected to a validation page hosted by the operator to validate the payment. After successful or unsuccessful validation the customer will be redirected to the contracting party's *okurl,* respectively *errorurl*. In this process, no PIN-SMS is sent and therefore no webvalidatepin request has to be executed by the contracting party.

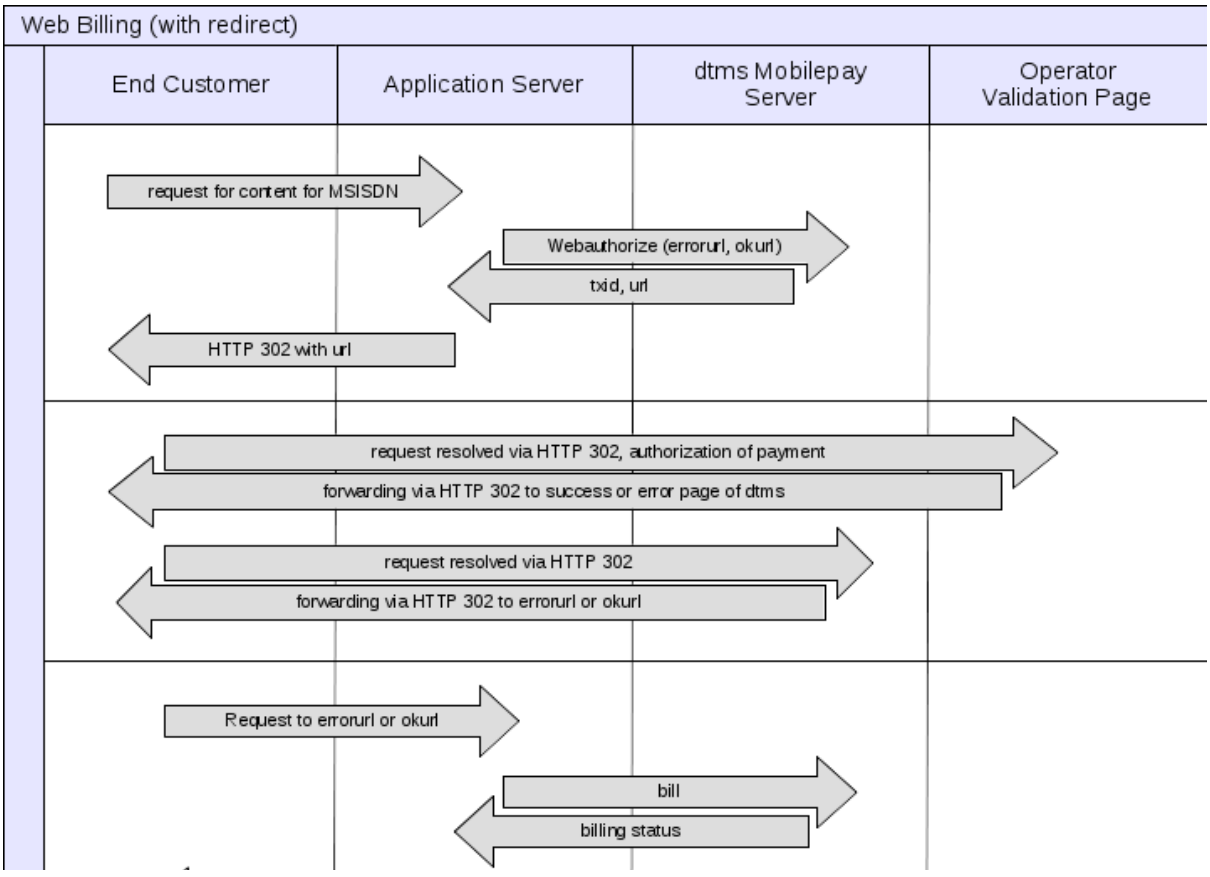


Figure 4 Process: Web Billing with redirect

4.1.2.3 Authorization Request

The parameters required for the HTTP request starting the authorization are listed in Table 4. The parameters contained in the HTTP response sent by *mobilepay* can be seen from Table 5.

Parameter	Description
command	***webauthorize***
servicename	The name of the billing service to use for the payment transaction It will be chosen by 4Pay and communicated to the contracting party.
password	The password belonging to the billing service. It will be chosen by 4Pay and communicated to the contracting party.
amount	The gross amount the customer is billed. The amount has to be given in the smallest possible unit of currency. The currency used is set to a fixed value on a per-service basis. For instance, *300* (Cent) needs to be sent for invoicing a customer 3 Euro.
type	The type of billing to perform (see chapter 2.2) Values available: {***single, multiple, cumulative***} When using single billing, exactly one debiting may take place per authorised transaction. Multiple billings, on the other hand, allow for several debits in a single transaction. Cumulative billings enable increasing the amount repeatedly before the debiting is affected.
msisdn	The MSISDN to bill
okurl	The URL to call using an HTTP 302 request after successful validation of the customer on the operator validation page. Must be set.
errorurl	The URL to call using an HTTP 302 request after unsuccessful validation of the customer on the operator validation page. Must be set.
mccmnc	*Mobile Country Code* (MCC) and *Mobile Network Code* (MNC) of the mobile network operator, if known Format: nnn-nn or nnnnn, e.g. 262-01 or 26201 for Telekom (262=MCC, 01=MNC).
stopsubcallbackurl	The URL to call using an HTTP request after a subscription has been terminated. That request contains the transaction's ID (txid) and the service's name (servicename) as POST parameters (see chapter 2.2.2). Only used for subscriptions.
txt1, txt2, txt3	Texts used to substitute the variables in the confirmation SMS to send (see chapter 4.8).

Parameter	Description
detail	Setting this optional parameter to ***true*** enables more verbose results (e.g. the parameter *bookingsmessage*) returned in the request's direct response. Values available: {***true***, ***false***}

Table 4 Request: Web Authorize

Parameter	Description
statuscode	A numerical code indicating the status of the request's result (see chapter 5)
statustext	A text describing the status code in text form (see chapter 5)
txid	A unique ID identifying the transaction. It is being generated automatically by *mobilepay* for each transaction.
url	If the parameter is available in the response, then the new process with redirect must be used (see chapter 4.1.2.2). If the parameter is missing, the previous process without redirect must be used (see chapter 4.1.2.1)
bookingmessage	Additional information (e.g. error message directly from the network operators) describing the result in greater detail. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
provider	Name of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
mcc	*Mobile Country Code* of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
mnc	*Mobile Network Code* of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
mtid	ID of the confirmation SMS sent. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Table 5 Response: Web Authorize

4.1.2.4 PIN Validation

The parameters required for the HTTP request used for PIN validation are listed in Table 7. The parameters contained in the HTTP response sent by *mobilepay* can be seen from Table 8.

Parameter	Description
command	***webvalidatepin***
servicename	The name of the billing service to use for the payment transaction It will be chosen by 4Pay and communicated to the contracting party.
password	The password belonging to the billing service. It will be chosen by 4Pay and communicated to the contracting party.
txid	A unique ID identifying the transaction. It is being generated automatically by *mobilepay* for each transaction.
pin	A one-time PIN sent to the customer by SMS after the authorization process has been started.
txt1, txt2, txt3	Texts used to substitute the variables in the confirmation SMS to send (see chapter 4.8).
detail	Setting this optional parameter to ***true*** enables more verbose results (e.g. the parameter *bookingsmessage*) returned in the request's direct response. Values available: {***true***, ***false***}

Table 6 Request: Web Validate PIN

Parameter	Description
statuscode	A numerical code indicating the status of the request's result (see chapter 5)
statustext	A text describing the status code in text form (see chapter 5)
bookingmessage	Additional information (e.g. error message directly from the network operators) describing the result in greater detail. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Table 7 Response: Web Validate PIN

4.1.2.5 Example: Web Billing

This example demonstrates how Web Billing may be used if the customer places an order using a website. The page shown in JSP 4 serves as an entry point where the customer starts the payment process.

```
<html>
   <head>
      <title>Web Billing Test</title>
   </head>
   <body>
      <table border="1" cellspacing="0" cellpadding="5" align="center">
         <tr>
            <td>
               <img src="mv.jpg"/>
               <br/><br/>
               <b>
                  Please enter your mobile subscriber number below to pay 6 Euro.
               </b>
               <br/><br/>
               <form method="GET" action="pin.jsp">
                  <input name="msisdn" type="text" value="491724565046"/> 
                  <input type="submit" value="Pay" />
               </form>
            </td>
         </tr>
      </table>
   </body>
</html>
```

JSP 4 index.jsp

The customer starts the payment process for 6 Euro and is being forwarded to a second page (JSP 5) where the authorization process is being started. Working in the background *mobilepay* will send the message containing the PIN code to the customer. This code then needs to be entered by the customer into the form on the page to finish validation of the payment transaction.

```
<%@ page import="de.4Pay.client.mobilepay.TechnicalProblemException" %>
<%@ page import="de.4Pay.client.mobilepay.MobilePaymentException" %>
<%@ page import="de.4Pay.client.mobilepay.WebPaymentClient" %>
<%
   String url = application.getInitParameter("PaymentServer");
   String service = application.getInitParameter("Service");
   String pw = application.getInitParameter("Password");
   String msisdn = request.getParameter("msisdn");
   String text = "";
   String txid = "";

   try {
      WebPaymentClient webp = new WebPaymentClient(url, service, pw);
      txid = webp.authorize(msisdn, 600, false);
      text = "Next you will receive an SMS containing a PIN code.<br/>"
              + "Please enter that PIN code below:<br/>"
              + "<form method=\"GET\" action=\"status.jsp\"> "
              + "<input name=\"pin\" type=\"text\"/>"
              + "<input name=\"txid\" type=\"hidden\" value=\"" + txid + "\"/>"
              + "<input type=\"submit\" value=\"Send\"/></form>";
   } catch (MobilePaymentException e) {
      text = "Unfortunately the payment could not be made: " + e.getStatusText();
   }  catch (TechnicalProblemException e1) {
      text = "The payment could not be made due to technical issues.";
   }
%>
<html>
   <head>
      <title>Web Billing Test</title>
   </head>
   <body>
     <table border="1" cellspacing="0" cellpadding="5" align="center">
         <tr>
            <td>
<%
               out.println(text);
%>
            </td>
         </tr>
      </table>
   </body>
</html>
```

JSP 5 pin.jsp

On a third page (JSP 6) the actual check of the PIN entered is being performed. If this is done successfully, the customer is forwarded again.

```
<%@ page import="de.4Pay.client.mobilepay.TechnicalProblemException" %>
<%@ page import="de.4Pay.client.mobilepay.MobilePaymentException" %>
<%@ page import="de.4Pay.client.mobilepay.WebPaymentClient" %>
<%
   String url = application.getInitParameter("PaymentServer");
   String service = application.getInitParameter("Service");
   String pw = application.getInitParameter("Password");
   String txid = request.getParameter("txid");
   String pin = request.getParameter("pin");
   String text = "";

   try {
    WebPaymentClient webp = new WebPaymentClient(url, service, pw);
    webp.validatePin(txid, pin);
    text="<a href=\"content.jsp?txid=" + txid + "\">This way to the content.</a>";
  } catch (MobilePaymentException e) {
    text = "Unfortunately the payment could not be made: " + e.getStatusText();
  } catch (TechnicalProblemException e1) {
    text = "The payment could not be made due to technical issues.";
  }
%>
<html>
   <head>
      <title>Web Billing Test</title>
   </head>
   <body>
      <table border="1" cellspacing="0" cellpadding="5" align="center">
         <tr>
            <td>
               <img src="mv.jpg"/>
               <br/><br/>
<%
               out.println(text);
%>
            </td>
         </tr>
      </table>
   </body>
</html>
```

JSP 6 status.jsp

On a fourth page (JSP 7) the debiting will be performed and the customer is shown the status of her order or the ordered content is displayed.

```
<%@ page import="de.4Pay.client.mobilepay.WebPaymentClient" %>
<%@ page import="de.4Pay.client.mobilepay.TechnicalProblemException" %>
<%@ page import="de.4Pay.client.mobilepay.MobilePaymentException" %>
<%
   String url = application.getInitParameter("PaymentServer");
   String service = application.getInitParameter("Service");
   String pw = application.getInitParameter("Password");

   String txid = request.getParameter("txid");
   String text = "";

   try {
      WebPaymentClient webp = new WebPaymentClient(url, service, pw);
      webp.bill(txid);
      text = "<img src=\"./bild.jpg\"/>";
   } catch (MobilePaymentException e) {
      text = "Unfortunately the payment could not be made: " + e.getStatusText();
   } catch (TechnicalProblemException e1) {
      text = "The payment could not be made due to technical issues.";
   }
%>
<html>
   <head>
      <title>Web Billing Test</title>
   </head>
   <body>
      <table border="1" cellspacing="0" cellpadding="5" align="center">
         <tr>
            <td>
<%
               out.println(text);
%>
            </td>
         </tr>
      </table>
   </body>
</html>
```

JSP 7 content.jsp

4.1.2.6 Example HTTPS Requests

https://www.mobilepay.de/mobilepay/mobilepay?command=webauthorize&servicename=MyService&password=MyPassword&msisdn=4917112345678&callbackurl=http://www.myserver.com/callback.php&amount=300&type=single&txt1=&txt2=txt2&txt3=txt3&detail=true

https://www.mobilepay.de/mobilepay/mobilepay?command=webvalidatepin&servicename=MyService&password=MyPassword&txid=42c0cda8001dc6e6010021b8b1cf0091&pin=321554&detail=true

4.1.3 Mobile Billing Authorization

Mobile Billling authorization differs noticeably from the three previous authorization variants. The process is illustrated in Figure 5.

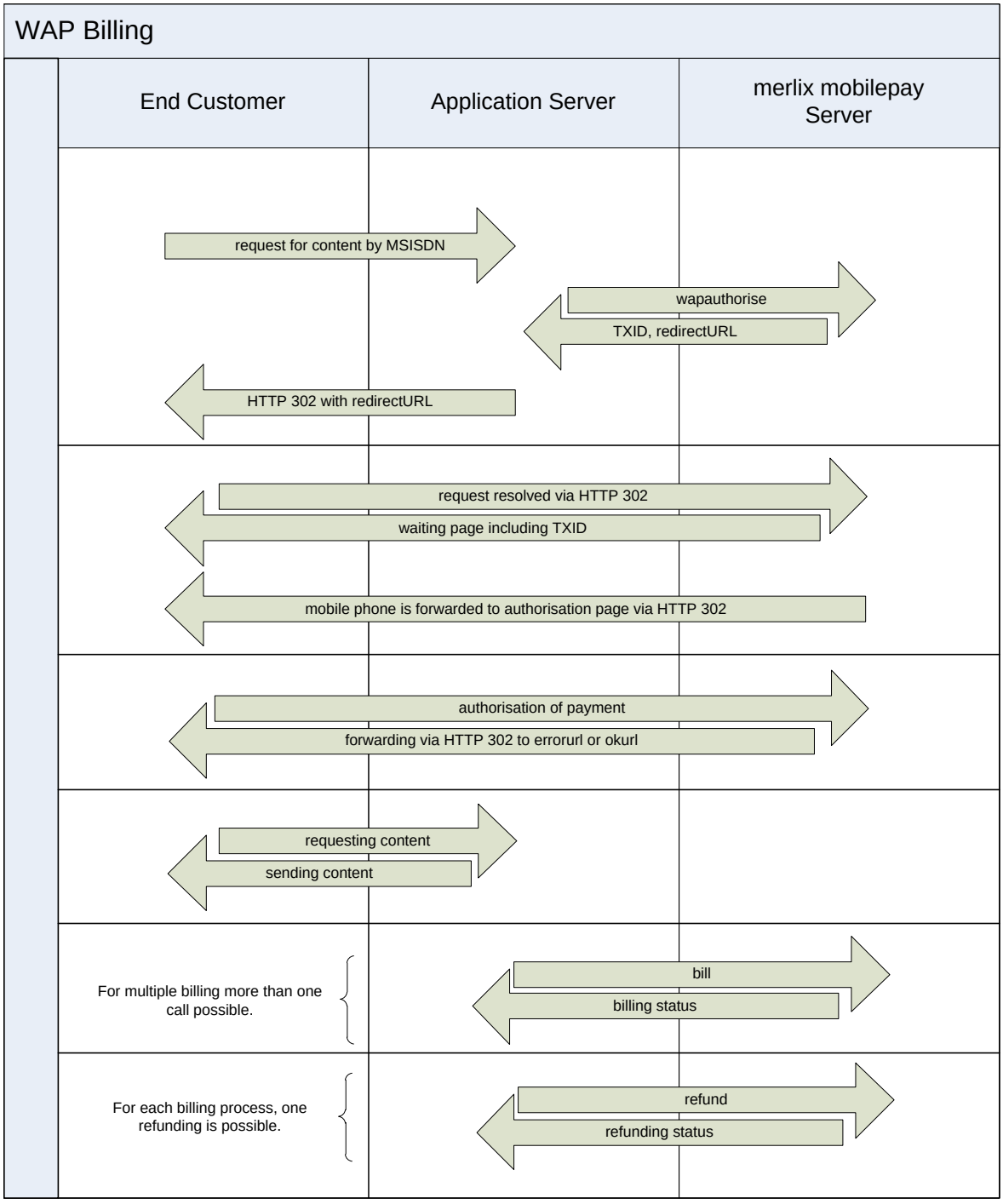


Figure 5 Process: Mobile Billing

The payment transaction is started by the contracting party using an HTTP request like it is done with the other billing variants (*wapauthorize*). Directly afterwards, using the HTTP Redirect mechanism the contracting party needs to forward the customer to a special page belonging to *mobilepay*. The payment will then be initiated by *mobilepay,* and the customer will be asked to confirm the transaction by clicking a hyperlink on that page. After this is done, *mobilepay* will again forward the customer back to the

contracting party's server, so the contracting party may render the service requested and perform the debiting using the *bill* command.

4.1.3.1 Authorization Request

The parameters required for the HTTP request starting the authorization are listed in Table 8. The parameters contained in the HTTP response sent by *mobilepay* can be seen from Table 9.

Parameter	Description
command	***wapauthorize***
servicename	The name of the billing service to use for the payment transaction It will be chosen by 4Pay and communicated to the contracting party.
password	The password belonging to the billing service. It will be chosen by 4Pay and communicated to the contracting party.
amount	The gross amount the customer is billed. The amount has to be given in the smallest possible unit of currency. The currency used is set to a fixed value on a per-service basis. For instance, *300* (Cent) needs to be sent for invoicing a customer 3 Euro.
type	The type of billing to perform (see chapter 2.2) Values available: {***single, multiple, cumulative***} When using single billing, exactly one debiting may take place per authorised transaction. Multiple billings, on the other hand, allow for several debits in a single transaction. Cumulative billings enable increasing the amount repeatedly before the debiting is affected.
msisdn	The MSISDN to bill (optional)
mccmnc	*Mobile Country Code* (MCC) and *Mobile Network Code* (MNC) of the mobile network operator, if known Format: nnn-nn or nnnnn, e.g. 262-01 or 26201 for Telekom (262=MCC, 01=MNC).
okurl	The URL to forward the customer to after successful validation of the payment transaction.
errorurl	The URL to forward the customer to if validating the payment transaction failed. In case the customer did not confirm the payment, the parameter *nabyuser=1* will be appended to the URL. If the validation failed due to technical reasons, this parameter will not be present.
stopsubcallbackurl	The URL to call using an HTTP request after a subscription has been terminated. That request contains the transaction's ID (txid) and the service's name (servicename) as POST parameters (see chapter 2.2.2). Only used for subscriptions.

Parameter	Description
txt1, txt2, txt3	Texts used to substitute the variables in the confirmation SMS to send (see chapter 4.8).
detail	Setting this optional parameter to ***true*** enables more verbose results (e.g. the parameter *bookingsmessage*) returned in the request's direct response. Values available: {***true***, ***false***}
description	Must contain a verbose description of the provided service. The value will be shown on the panel used for the user authorization.
gtc	Must contain a link to the general terms and conditions of the provided service. The link be shown on the panel used for the user authorization.
imprint	Must contain a link to the imprint of the provided service. The link be shown on the panel used for the user authorization.
contact	Must contain a link to the contacts of the provided service. The link be shown on the panel used for the user authorization.
faq	Must contain a link to the faq of the provided service. The link be shown on the panel used for the user authorization.

Table 8 Request: WAP Authorize

Parameter	Description
statuscode	A numerical code indicating the status of the request's result (see chapter 5)
statustext	A text describing the status code in text form (see chapter 5)
txid	A unique ID identifying the transaction. It is being generated automatically by *mobilepay* for each transaction.
url	The URL the contracting party needs to forward the customer to for the purpose of performing authorization.
bookingmessage	Additional information (e.g. error message directly from the network operators) describing the result in greater detail. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
provider	Name of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
mcc	*Mobile Country Code* of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Parameter	Description
mnc	*Mobile Network Code* of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
mtid	ID of the confirmation SMS sent. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Table 9 Response: WAP Authorize

4.1.3.2 Example: Mobile Billing

This example demonstrates how Mobile Billing may be implemented. It is assumed the subscriber's MSISDN is already known. If this is not the case, the process explained in chapter 7 may be used for determining it first. An example for a landing page for the customer is given under JSP 8. After the customer has loaded this page, the authorization process will be started in the background, and a new payment transaction will be created. In addition, a redirection URL is generated that needs to be embedded as a hyperlink in the page shown, so the customer may click it to begin the authorization.

```
<%@ page import="de.4Pay.client.mobilepay.TechnicalProblemException" %>
<%@ page import="de.4Pay.client.mobilepay.MobilePaymentException" %>
<%@ page import="de.4Pay.client.mobilepay.WapPaymentClient" %>
<%
   String url = application.getInitParameter("PaymentServer");
   String service = application.getInitParameter("Service");
   String pw = application.getInitParameter("Password");
   String txid = request.getParameter("txid");
   String text = "";

   try {
      WapPaymentClient wapp = new WapPaymentClient(url, service, pw);
      wapp.bill(txid);
      text = "This is your content.";
   } catch (MobilePaymentException e) {
      text = "Unfortunately the payment could not be made: " + e.getStatusText();
   } catch (TechnicalProblemException e1) {
      text = "The payment could not be made due to technical issues.";
   }
%>
   <wml>
      <card id="test">
     <p>
<%
         System.out.println(text);
%>
      </p>
   </card>
</wml>
```

JSP 8 index.jsp

Once the customer has followed the link all further steps in the process happen automatically and without any interaction by the contracting party. After the authorization has finished successfully, the customer will be forwarded to the URL passed as *okurl*. In this example, this is the page shown under JSP 9 where the debiting is performed and the ordered content is delivered.

```
<%@ page import="de.4Pay.client.mobilepay.TechnicalProblemException" %>
<%@ page import="de.4Pay.client.mobilepay.MobilePaymentException" %>
<%@ page import="de.4Pay.client.mobilepay.WapPaymentClient" %>
<%
   String url = application.getInitParameter("PaymentServer");
   String service = application.getInitParameter("Service");
   String pw = application.getInitParameter("Password");
   String txid = request.getParameter("txid");
   String text = "";

   try {
      WapPaymentClient wapp = new WapPaymentClient(url, service, pw);
      wapp.bill(txid);
      text = "This is your content.";
   } catch (MobilePaymentException e) {
      text = "Unfortunately the payment could not be made: " + e.getStatusText();
   } catch (TechnicalProblemException e1) {
      text = "The payment could not be made due to technical issues.";
   }
%>
   <wml>
      <card id="test">
     <p>
<%
         System.out.println(text);
%>
      </p>
   </card>
</wml>
```

JSP 9 ok.jsp

If authorization fails, the customer will be forwarded to the URL passed as value of the parameter *errorurl*. In this example that would be the page displayed as JSP 10. On this page she will be informed about the transaction's failure.

```
<wml>
   <card id="test">
      <p>You have not authorised the payment, yet.</p>
   </card>
</wml>
```

JSP 10 error.jsp

4.1.3.3 Example HTTPS Request

https://www.mobilepay.de/mobilepay/mobilepay?command=wapauthorize&servicename=MyService&password=MyPassword&msisdn=4917112345678&amount=300&okurl=http://www.myserver.com/ok.php&errorurl=http://www.myserver.com/error.php&amount=300&type=single&txt1=&txt2=txt2&txt3=txt3&detail=true

4.2 Billing Request

The billing request serves to affect the actual debiting of the amount to pay after the authorization's successful completion.

The parameters required for the HTTP request are listed in Table 10. Table 11 lists the parameters contained in the response by *mobilepay*.

Parameter	Description
command	***bill***
servicename	The name of the billing service to use for the payment transaction It will be chosen by 4Pay and communicated to the contracting party.
password	The password belonging to the billing service. It will be chosen by 4Pay and communicated to the contracting party.
txid	A unique ID identifying the transaction. It is generated automatically by *mobilepay* for each transaction.
amount	The gross amount the customer is billed. The amount has to be given in the smallest possible unit of currency. The currency used is set to a fixed value on a per-service basis. For instance, *300* (Cent) needs to be sent for invoicing a customer 3 Euro. This parameter is optional. If it is not given, the amount specified at authorization is used. If a value is given, it must be equal or less the amount used for authorization.
detail	Setting this optional parameter to ***true*** enables more verbose results (e.g. the parameter *bookingsmessage*) returned in the request's direct response. Values available: {***true***, ***false***}

Table 10 Request: Bill

Parameter	Description
statuscode	A numerical code indicating the status of the request's result (see chapter 5)
statustext	A text describing the status code in text form (see chapter 5)
bookingmessage	Additional information (e.g. error message directly from the network operators) describing the result in greater detail. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
Additional Parameter for Mobile Billing	
msisdn	The MSISDN of the customer.

	This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
mcc	*Mobile Country Code* of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.
mnc	*Mobile Network Code* of the provider used for authorization. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Table 11 Response: Bill

4.2.1 Example HTTPS Request

https://www.mobilepay.de/mobilepay/mobilepay?command=bill&servicename=MyService&password=MyPassword&txid=42c0cda8001dc6e6010021bfb1cd2091&detail=true

4.3 Bulk Billing

Apart from the direct and synchronous execution of the debiting using the *bill* command in a payment transaction, another option exists. It consists of performing up to 500 debits simultaneously in an asynchronous fashion. As a prerequisite for using this mechanism all of the transactions to be billed need to be authorized already since debiting would fail otherwise.

Hence, the typical scenario for using this mechanism is debiting a larger number of subscription transactions where billings are performed on a regular basis. From a technical point of view the contracting party will first request a bulk billing by calling the appropriate command and supplying a list of TXIDs indicating the transactions to be billed. The *mobilepay* system will process this request and perform a callback request after completion containing the execution's results.

4.3.1 Billing Request

The request used for employing bulk billing follows the syntax established by the commands already described. The parameters required for the HTTP request are listed in Table 12. The parameters contained in the HTTP response sent by *mobilepay* can be seen from Table 13.

Parameter	Description
command	***billbulk***
servicename	The name of the billing service to use for the payment transaction It will be chosen by 4Pay and communicated to the contracting party.
password	The password belonging to the billing service. It will be chosen by 4Pay and communicated to the contracting party.

Parameter	Description
txidlist	The list of Ids denoting the transactions to be billed. The individual TXIDs need to be separated by the character ';'. A single bulk billing request may contain up to 500 transactions.
amount	The gross amount the customer is billed. The amount has to be given in the smallest possible unit of currency. The currency used is set to a fixed value on a per-service basis. For instance, *300* (Cent) needs to be sent for invoicing a customer 3 Euro. This parameter is optional. If it is not given, the amount specified at authorization is used. If a value is given, it must be equal or less the amount used for authorization.
bulkuid	An ID serving as a unique key to the bulk. It is used to avoid executing a single bulk multiple times. If a second bulk is transmitted using the same ID, it will be rejected. The ID may be chosen as an arbitrary string with a length of up to 32 characters.
callbackurl	The URL to call using an HTTP request after authorization finished successfully. That request contains the transaction's ID (*txid*) and the service's name (*servicename*).
detail	Setting this optional parameter to ***true*** enables more verbose results (e.g. the parameter *bookingsmessage*) returned in the request's direct response. Values available: {***true***, ***false***}

Table 12 Request Bill Bulk

Parameter	Description
statuscode	A numerical code indicating the status of the request's result (see chapter 5)
statustext	A text describing the status code in text form (see chapter 5)
bookingmessage	Additional information (e.g. error message directly from the network operators) describing the result in greater detail. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Table 13 Response Bill Bulk

4.3.1.1 Example HTTPS Request

https://www.mobilepay.de/mobilepay/mobilepay?command=billbulk&servicename=MyService&password=MyPassword&txidlist=42c0cda8001dc6e6010021bfb1cd2091;42c0cda8001dc6e6010021bfb1cd3653;42c0cda8001dc6e6010021bfb1cd477a&amount=199&bulkuid=myUniqueBulkId4711&callbackurl=http://www.myserver.com/billbulk.php&detail=true

4.3.2 Callback Request

After the bulk has been fully processed, *mobilepay* executes a callback request for the URL passed in the request initiating the bulk. This request serves to return the bulk execution's results. The parameters contained in this HTTP request are listed in Table 14. The contracting party's system is required to acknowledge having received the results by replying with a request of its own. That one is described in Table 15.

Parameter	Description
txidlist	The list of TXIDs processed in the bulk. ID are separated from one another by the character ';'.
resultlist	The results of the bulk's processing. These are ordered in the same way as the TXIDs and separated by the character ';'. Each result may either have the value *true* or *false* indicating whether the billing succeeded or failed respectively.
resultmessagelist	A list of messages detailing the bulk processing's results. These are ordered the same as the TXIDs and separated by the character ';'.

Table 14 Request: Bill Bulk Callback

Parameter	Description
	After having received the callback request, an acknowledgement by sending another request to *mobilepay* is required. Its body needs to contain the string *OK* or else *mobilepay* will retry transmitting the results until receiving the acknowledgement.

Table 15 Response: Bill Bulk Callback

4.3.3 Java Client

The Java-based client provided by 4Pay implements the usage of bulk billing. To this end, the concrete classes derived from *AbstractMobilePaymentClient* each contain the two methods:

doBillBulk(List<String>, String, String, Long)

and

doBillBulk(List<String>, String, String)

Both expect a list of transactions to be billed as their first parameter. Keep in mind that prior authorization for each of them is required or else executing the billing will fail. The second parameter passed is the callback URL used to report the results of the bulk's execution to. The bulk ID is passed as the third parameter. Furthermore, using the four parameter method it is possible to specify the amount of money to be debited if the user wishes to perform a billing for an amount different from the one initially authorised

(only amounts equal or less to the one authorised are allowed). The method's second variant which only takes three parameters omits this feature and instead will debit the amount of money given in the authorization.

For handling the callback request, the client contains a special servlet that may be used. It is, however, necessary to enable usage of this servlet first by modifying the configuration of the server utilised as shown in XML 3.

```
<context-param>
   <param-name>CallBack</param-name>
   <param-value>http://localhost:8080/example/bulkresult</param-value>
</context-param>
<context-param>
   <param-name>BulkResponseProcessor</param-name>
   <param-value>
        de.4Pay.client.mobilepay.ExampleBulkResponseProcessor
   </param-value>
</context-param>
<servlet>
   <servlet-name>bulkresponseprocessor</servlet-name>
   <servlet-class>
      de.4Pay.client.mobilepay.BulkResponseReceiver
   </servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>bulkresponseprocessor</servlet-name>
   <url-pattern>/bulkresult</url-pattern>
</servlet-mapping>
```

XML 3 web.xml

The parameter *BulkResponseProcessor* specifies the class used to handle processing of the callback request's data. This class is required to implement the interface *BulkResponseProcessor*. In case the data cannot be processed the method *process()* is required to throw a *ResponseNotStoredException*. In this case, the *mobilepay* server will resend the callback. An example for a processor implementation printing the results to console is shown under Code 2.

```
package de.4Pay.client.mobilepay;

import java.util.List;

public class ExampleBulkResponseProcessor implements BulkResponseProcessor {

   public void process(List results) throws ResponseNotStoredException {

      for (int i = 0; I < results.size(); i++) {
         BulkResult result = (BulkResult) results.get(i);
         System.out.println(result.getTxid() + ":"
            + result.getResult() + ":" + result.getResultmessage());
      }

   }
}
```

Code 2 Example: processing the callback request

4.4 Refunding

The refunding mechanism serves to undo a prior billing request and credit the amount of money to the customer.

Take note that several restrictions apply to this operation. Basically, it may only be used after at least one debiting has been performed. Each debiting may only be refunded exactly once. In addition, only the last debiting performed in a transaction may be refunded. This restriction mainly concerns subscriptions since it will always only be possible to refund the last one in a series of debits.

It is, however, possible to perform another debiting after having refunded a previous one. This new debiting may itself be refunded again although mobile network operators are known that do not support repeated debiting and refunding.

The parameters required for the HTTP request are listed in Table 16. Table 17 lists the parameters contained in the response by *mobilepay*.

Parameter	Description
command	***refund***
servicename	The name of the billing service to use for the payment transaction It will be chosen by 4Pay and communicated to the contracting party.
password	The password belonging to the billing service. It will be chosen by 4Pay and communicated to the contracting party.
txid	A unique ID identifying the transaction. It is generated automatically by *mobilepay* for each transaction.
amount	The gross amount credited to the customer. The amount has to be given in the smallest possible unit of currency. The currency used is set to a fixed value on a per-service basis. For instance *300* (Cent) needs to be transferred for invoicing a customer 3 Euro. This parameter is optional. If it is not given, the amount specified at authorization or billing (if given) is used. If a value is given, it must be equals or less the amount used for authorization.
detail	Setting this optional parameter to ***true*** enables more verbose results (e.g. the parameter *bookingsmessage*) returned in the request's direct response. Values available: {***true***, ***false***}

Table 16 Request: Refund

Parameter	Description
statuscode	A numerical code indicating the status of the request's result (see chapter 5)

Parameter	Description
statustext	A text describing the status code in text form (see chapter 5)
bookingmessage	Additional information (e.g. error message directly from the network operators) describing the result in greater detail. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Table 17 Response: Refund

4.4.1 Example HTTPS Request

https://www.mobilepay.de/mobilepay/mobilepay?command=refund&servicename=MyService&password=MyPassword&txid=42c0cda8001dc6e6010021bfb1cd2091&detail=true

4.5 Increasing the Amount

If a cumulative payment transaction has been authorised, it is possible to increase the amount to pay repeatedly before performing the actual debiting. For this purpose, the *Increase Amount* command exists. It may only be used if the transaction's type is set to *cumulative* (see chapter 2.2.3). As an additional restriction, cumulative billing is currently only supported for Telekom customers. For a transaction supporting the mechanism the command may be called repeatedly each time increasing the amount to bill by the amount given. Increasing may only be done as long as the *bill* command has not been executed yet. As soon as the debiting is executed, no further increases are possible and the transaction will be considered finished.

The parameters required for the HTTP request are listed in Table 18. Table 19 lists the parameters contained in the response by *mobilepay*.

Parameter	Description
command	***increaseamount***
servicename	The name of the billing service to use for the payment transaction It will be chosen by 4Pay and communicated to the contracting party.
password	The password belonging to the billing service. It will be chosen by 4Pay and communicated to the contracting party.
txid	A unique ID identifying the transaction. It is generated automatically by *mobilepay* for each transaction.
amount	The gross amount to increase the amount the customer is invoiced by. The amount has to be given in the smallest possible unit of currency. The currency used is set to a fixed value on a per-service basis. For instance *300* (Cent) needs to be transferred for increasing the amount by 3 Euro.

Parameter	Description
detail	Setting this optional parameter to ***true*** enables more verbose results (e.g. the parameter *bookingsmessage*) returned in the request's direct response. Values available: {***true***, ***false***}

Table 18 Request: Increase Amount

Parameter	Description
statuscode	A numerical code indicating the status of the request's result (see chapter 5)
statustext	A text describing the status code in text form (see chapter 5)
bookingmessage	Additional information (e.g. error message directly from the network operators) describing the result in greater detail. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Table 19 Response: Increase Amount

4.5.1 Example HTTPS Request

https://www.mobilepay.de/mobilepay/mobilepay?command=increaseamount&servicename=MyService&password=MyPassword&txid=42c0cda8001dc6e6010021bfb1cd2091&amount=199&detail=true

4.6 Stopping a Subscription

Using the interface of *mobilepay* the contracting party may terminate existing subscription transactions (see chapter 2.2.2). Executing this command results in the transaction being closed and no more billings being possible. Furthermore, an HTTP callback will be executed to the URL provided at authorization.

The parameters required for the HTTP request are listed in Table 20. Table 21 lists the parameters contained in the response by *mobilepay*.

Parameter	Description
command	***stopsubscription***
servicename	The name of the billing service to use for the payment transaction It will be chosen by 4Pay and communicated to the contracting party.

Parameter	Description
password	The password belonging to the billing service. It will be chosen by 4Pay and communicated to the contracting party.
txid	A unique ID identifying the transaction. It is generated automatically by *mobilepay* for each transaction.
detail	Setting this optional parameter to ***true*** enables more verbose results (e.g. the parameter *bookingsmessage*) returned in the request's direct response. Values available: {***true***, ***false***}

Table 20 Request: Stop Subscription

Parameter	Description
statuscode	A numerical code indicating the status of the request's result (see chapter 5)
statustext	A text describing the status code in text form (see chapter 5)
bookingmessage	Additional information (e.g. error message directly from the network operators) describing the result in greater detail. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Table 21 Response: Stop Subscription

4.6.1 Example HTTPS Request

https://www.mobilepay.de/mobilepay/mobilepay?command=stopsubscription&servicename=MyService&password=MyPassword&txid=42c0cda8001dc6e6010021bfb1cd2091&detail=true

4.7 Looking Up the Mobile Network Operator

The *mobilepay* interface provides means for querying for the mobile network operator used in a previous authorization.

The parameters required for the HTTP request are listed in Table 22. Table 23 lists the parameters contained in the response by *mobilepay*.

Parameter	Description
command	***getmno***

Parameter	Description
servicename	The name of the billing service to use for the payment transaction It will be chosen by 4Pay and communicated to the contracting party.
password	The password belonging to the billing service. It will be chosen by 4Pay and communicated to the contracting party.
txid	A unique ID identifying the transaction. It is generated automatically by *mobilepay* for each transaction.
detail	Setting this optional parameter to ***true*** enables more verbose results (e.g. the parameter *bookingsmessage*) returned in the request's direct response. Values available: {***true***, ***false***}

Table 22 Request: Get MNO

Parameter	Description
statuscode	***600***
statustext	The *mobile network operator* (MNO) used for authorization
bookingmessage	Additional information (e.g. error message directly from the network operators) describing the result in greater detail. This parameter will be returned only if ***detail*** had been set to ***true*** in the initial request.

Table 23 Response: Get MNO

4.7.1 Example HTTPS Request

https://www.mobilepay.de/mobilepay/mobilepay?command=getmno&servicename=MyService&password=MyPassword&txid=42c0cda8001dc6e6010021bfb1cd2091&detail=true

4.8 Customizing the Message Texts

For all text messages sent by 4Pay on behalf of the contracting party the possibility for customization exists. The texts sent will be configured by 4Pay for each service individually on the *mobilepay* system. They may, however, contain wild cards that will be replaced dynamically by values passed as HTTP request parameters. The texts can be provided by the contracting party. Please note that these may be subject to regulatory rules that need to be respected.

The message texts themselves cannot only be configured on a per service basis but also per billing variant (see chapter 2.1) and type (see chapter 2.2). Additional configurable texts exist for a monthly message to the customer informing her about the

amount of money spent for that period and warning messages triggered by having spent certain amounts of money (*bill warnings*).

4.8.1 Wild Cards

Table 24 lists all wild cards available and their respective descriptions. The texts configured may contain these wild cards which will be replaced according to their descriptions.

Wild card	Description
%t1	Will be replaced by request parameter *txt1*.
%t2	Will be replaced by request parameter *txt1*.
%t3	Will be replaced by request parameter *txt1*.
%a	The amount in cent. Will be replaced by the request parameter *amount*.
%c	The amount in Euro. Will be replaced by the request parameter *amount*.
%p	The PIN used in Web Billing. Will be replaced by the request parameter *pin* (see chapter 4.1.2.4).

Table 24 SMS Text Wildcards

4.8.2 Order of Wild Cards

Replacing the wild cards is done in a specific order. This opens up the possibility of passing values for wild cards which in turn may again contain wildcards. First, the wild cards *%t1*, *%t2* and *%t3* will be replaced in that order. Second, the wild cards *%a* and *%c* will be replaced. Finally, when using the *webvalidatepin* command (see chapter 4.1.2.4), *%p* will be replaced.

Making use of this order it would for instance be possible to pass the wild cards *%a* or *%c* in one of the request parameters *%tx1-%tx3* and still have them replaced correctly in the messages sent.

5 Status Codes

The status codes returned by *mobilepay* are listed in Table 25.

Code	Descriptions
100	OK.
200	Technical error.
300	Billing for this MSISDN not available.
305	Increase amount not possible
310	Not authorized.
320	Service not available.
330	Txid not valid.
340	PIN not valid.
350	MSISDN not valid.
351	Request formally OK. MCCMNC invalid.
355	Transaction not found.
360	Refund not possible.
370	Amount too high
380	Txid does not match service
390	Service not available.
391	No credit
392	Cumulative billing not allowed for this service
393	Service is disabled
394	Billing variant unavailable
395	Billing type unavailable
400	Amount already billed.
405	Increase amount not allowed for Transaction
410	Transaction was not validated.
411	Transaction already validated
420	A transaction is still pending. Cannot start another transaction.
430	Amount not yet billed. Refund not possible.
440	Subscription was already stopped. Cannot stop again.
441	Transaction was already aborted
450	Monthly limit exceeded. No more billings possible in this month.

Code	Descriptions
455	MSISDN blacklisted.
460	BulkUid already used. Choose another BulkUid.
470	BulkUid missing.
480	TXID list missing.
481	TXID list in wrong format. Expected as string separated by ';'.
482	TXID list too long. Send a maximum of 500 TXIDs.
490	Callback-URL missing.
491	Request parameter missing or invalid value
492	Command not found
600	MNO, über den die Zahlung autorisiert wurde.

Table 25 Status Codes

6 Country Specifics

Unfortunately, it is not possible to completely unify the behaviour of the *mobilepay* system for all connections. The reason for this is differences in the connections employed in each country that cannot be levelled. Chapter 2 describes the processes for Germany.

7 MSISDN Lookup for Mobile Billing

Since knowing the MSISDN of a subscriber is a prerequisite for performing Mobile Billing, the mobile network operators provide means for determining it if it is not yet known. Using this functionality is, however, different for each operator. Thus, *mobilepay* provides a unified mechanism usable by the contracting party for identifying the customer's MSISDN.

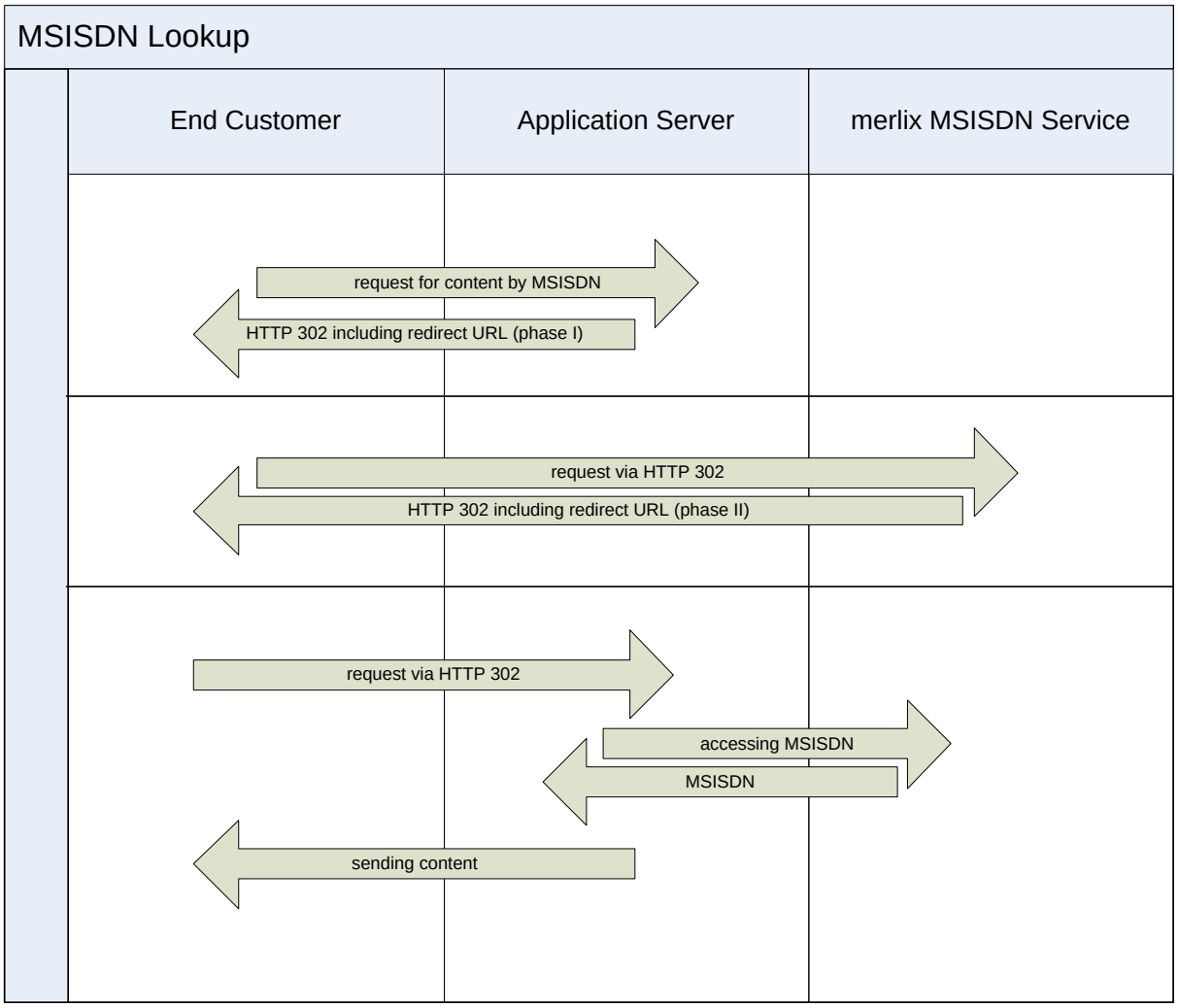


Figure 6 Process: MSISDN Lookup

From a technical point of view determination of the MSISDN is based on using multiple HTTP redirects. That process is shown in Figure 6. First, the customer requests a service at the contracting party and then receives an HTTP redirect forwarding her to a page belonging to *mobilepay*. The *mobilepay* system is then able to determine the mobile subscriber's MSISDN. This will be done in the background while the customer is transparently redirected back to the contracting party's system. After the MSISDN lookup finished, the MSISDN may be obtained from *mobilepay* by the contracting party by querying the MSISDN interface using a simple HTTP request.

.

7.1 Determining the MSISDN

As initially described, the mobile subscriber is forwarded repeatedly using HTTP redirects for identifying her MSISDN. Thus, the determination process consists of several phases that are described in the following sections.

For Deutsche Telekom (t-mobile) there is a different process: If the redirect in Phase II provides **OK** for *StatusInfo* and **t-mobile** for *Provider* (see also Table 26), the process described in 7.1.3 will not return any value for *Result*, hence this step can be skipped for Deutsche Telekom and no MSISDN will be provided during this step. Consequentially, the command *wapauthorize* is done without parameter MSISDN for Deutsche Telekom. In other words: you can always call *wapauthorize* directly, without msisdn, and 4Pay will resolve the msisdn during the whole transaction.

Nonetheless, the MSISDN will be returned after a successful bill request (Table 11 Response: Bill).

7.1.1 Phase I

The URL redirected to is required to contain the parameters listed in Table 26. In addition, an arbitrary number of custom parameters may be passed which will be returned by *mobilepay* in phase II.

Parameter	Description
ID	The name identifying the account to use for determining the subscriber's MSIDN
PASS	The password belonging to the account

Table 26 Parameters: Redirect Phase I

The URL to redirect the customer to is as follows:

http://wap.mobilepay.de/msisdnservice/getmsisdn

7.1.2 Phase II

The redirection used in phase II contains the parameters listed in Table 27 in addition to the ones from phase I.

Parameter	Description
StatusInfo	Indicates whether or not looking up the MSISDN was successful. ***FAILED***: MSISDN could not be determined. In this case the parameter *Status* will contain further details. ***OK***: MSISDN successfully determined
MSISDNID	An ID that may be used to retrieve the MSISDN after the lookup process has finished. Only usable once.
PROVIDER	The subscriber's mobile network operator (vodafone, t-mobile, O2)
STATUS	Extended information in case MSISDN lookup has failed

Table 27 Parameters: Redirect Phase II

7.1.3 Querying for the MSISDN

After successful completion of the lookup process, the subscriber's MSISDN may be obtained from *mobilepay*. The parameters required for this request are listed in Table 28. Querying is done by calling the following URL using an HTTP or HTTPS request:

http://wap.mobilepay.de/msisdnservice/msisdnproxy

Parameter	Description
ID	The ID identifying the customer on whose behalf the MSISDN is determined
PASS	The password for accessing the system from phase I
MSISDNID	The *MSISDN-ID* returned in phase II
Result	The MSISDN as text in the format: +<CC><NETWORK><SUBSCRIBER>

Table 28 Parameters: Request MSISDN

7.2 Code Example

An example for using the MSISDN lookup mechanism provided by *mobilepay* is given in *XML 4 Example: MSISDN Lookup*. First, this Java servlet redirects the mobile subscriber to a special page belonging to *mobilepay* used for lookup of the MSISDN. Afterwards, the subscriber's MSISDN will be obtained from the *mobilepay* server and can be used for performing Mobile Billing.

```
public class ExampleServlet extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response) throws
                                                                          IOException{
    …
    response.setHeader("Expires","Mon 26 Jul 1997 05:00:00 GMT");
    response.setHeader("Cache-Control","no-cache, must-revalidate");
    response.setHeader("Pragma","no-cache");
    String status = request.getParameter("STATUS");
    String msisdnid = request.getParameter("MSISDN");
    String msisdn = "FAILED";
    String provider = request.getParameter("PROVIDER");

    if (status == null) {                  // redirect to msisdn-interface for retrieval
      response.sendRedirect(
                "http://localhost/msisdnservice/getmsisdn?ID=service1&PASS=1234");
    } else if (status.equals("OK")){       // retrieval was successful get msisdn
       try {
          this.fetchMSISDNFromProxy(msisdnid);
       } catch (IOException ioe) {
          msisdn = "failed. "+ioe;
       }
    } else {                               // retrieval was not successful
        …
    }
     …
  }

  private String fetchMSISDNFromProxy(String msisdnid) throws IOException{
     HttpClient client = new HttpClient();

     PostMethod pm = new PostMethod();
     pm.setRequestHeader("Connection", "close");
     pm.addParameter("ID", "service1");
     pm.addParameter("PASS", "1234");
     pm.addParameter("MSISDNID", msisdnid);

     URI u = new URI("http://wap.mobilepay.de/msisdnservice/msisdnproxy", true);

     pm.setURI(u);

     int rc = (-1);

     rc = client.executeMethod(pm);

     if (rc == 200) {
        return pm.getResponseBodyAsString();
     } else {
        throw new IOException("fetching failed "+rc+" [" + pm.getStatusText() + "]");
     }
  }
}
```

XML 4 Example: MSISDN Lookup

Annex 1

Interface Description

Premium SMS and

SMS Termination

Version: 1.1

Date: 27.05.2017

4Pay Networks GmbH
Fischerstraße 49
D - 40477 Düsseldorf

Geschäftsführer: Martin Kolisch
HRB79650, Amtsgericht Düsseldorf
Steuernummer 103/5751/2620

Commerzbank Düsseldorf
IBAN DE91 3004 0000 0151 5154 00
BIC COBADEDDXXX

Inhalt

Change History

Version	Date	Author	Description
1.0	23.03.2017	Yusif Goabra	Released version 1.0
1.1	29.05.2017	Martin Kolisch	Added missing parameter for sms termination
1.1	28.06.2017	Christoph Berger	Review and formatted

1. Glossary

Abbreviation	Explanation
CSV	Comma-separated values
DLR	Delivery Report. Provides (usually) asynchronous information about the status of the message
E.164	ITU-T recommendation for the international public telecommunication numbering plan1
HLR	The home location register is a central database that contains details of each mobile phone subscriber that is authorized to use the GSM core network.
Inbound	See MO
Inbound Port	See MO Port
ISO-8601	International Standard for the representation of dates and times[2] The used format used in this interface is: yyyy-MM-dd'T'HH:mm:ss.SSSZ where: yyyy = four-digit year MM = two-digit month (01=January, etc.) dd = two-digit day of month (01 through 31) HH = two digits of hour (00 through 23) mm = two digits of minute (00 through 59) ss = two digits of second (00 through 59) SSS = digits represent-ing milliseconds Z = time zone designator Example: 2009-07-23T12:00:00.000+0200
MCC	Mobile Country Code
MG	Message Gateway
MNC	Mobile Network Code
MO	Mobile Originated: A SMS message which is sent from a mobile end device to, for example, the marketing. The equivalent of “Inbound”.
MO Port	Mobile phone number or short code to which the SMS messages can be sent and which will be processed by a server. The equivalent of Inbound Port.
MSISDN	Mobile Subscriber Integrated Services Digital Network Number, a number identifying a subscription in a mobile network.
MT	Mobile Terminated: A SMS message which, for example, is sent from the server to a mobile end device. The equivalent of “Outbound”.
NL	Number Lookup
Outbound	See MT
Provider	Service provider which is hooked up to the marketing server and which delivers SMS messages.
SMPP	Short Message Peer-to-Peer protocol for exchanging SMS messages
SMTP	Simple mail transfer protocol
UTC	Coordinated universal time (time standard based on International Atomic Time)[3]

2. SMS

2.1. SMS via HTTP

2.1.1. SMS transmission (MT)

2.1.1.1. Service URL

The MG receives SMS messages for the transmission under the following URL (POST):

http://<IPAddressMMG>/merlix/rest/sendsmsdlr

2.1.1.2. HTTP Request Parameter

The interface expects parameters according to the following chart. The parameters have to be (exclusively) transmitted per POST. The block *Use* states if a parameter is optional (O) or mandatory (M).

Name	Use	Meaning
login	M	Your login will be sent in a separated email
pin	M	Your pin will be sent in a separated email
service	M	Service name will be sent in a separated email
msg	M	(Alternative to the parameter data) The content of the text message. If this parameter is blank a space will be set.
udh	O	The UDH for text or binary messages, e.g. multipart text messages
data	M	(Alternative to the parameter msg) The content of the binary message with the format (UDH/ only for legacy purpose, use udh parameter instead) CONTENT.
snr	M	The source identifier of the message.
dnr	M	The receiver's number of the message with the international format beginning with 00, e.g. 00491701234567.
type	M	"text" or "binary"
callbackurl	O	URL to which the DLRs have to be reported.
externaldata	O	Identifier for the message which was given by the initiator and which is included in the DLR reports; the setting of this parameter is only sensible if the callbackurl is set, too.

Name	Use	Meaning
		sensible if the callbackurl is set, too.
Uuid	O	
ccuser	O	For further details, contact 4Pay

ccportal	O	For further details, contact 4Pay
cctimestamp	O	For further details, contact 4Pay
expiry	O	Allows you to specify the validity period of the message. It could be supplied in relative or absolute format: -relative: “+10” means a validity period of ten minutes -absolut: supplied as UTC in ISO-8601 Format (please see Glossary)

2.1.1.3. SMS MT Response Format

Response Code	Example http-Response	Description
200	<?xml version="1.0"> <message-events> <message-event> <type>MT_ACCEPTED</type> <time>2008-09-04T11:45:08.554+0200</time> <message-id>987654321ABCD</message-id> <external-data>1234567890</external-data> </message-event> </message-events>	Message has been accepted by the SMSC.
202	<?xml version="1.0"> <message-events> <message-event> <type>MT_BLACKLISTED</type> <time>2008-09-04T11:45:08.554+0200</time> </message-event> </message-events>	Message will not be sent since it is on the blacklist.
400	<?xml version="1.0"> <message-events> <message-event> <type>MT_INVALID</type> <time>2008-09-04T11:45:08.554+0200</time> </message-event> </message-events>	Occurs if, for example, the sender and the receiver-address cannot be parsed.
400	<?xml version="1.0"> <message-events> <message-event> <type>MT_ACCEPTION_EXCEPTION</type> <time>2008-09-04T11:45:08.554+0200</time> </message-event> </message-events>	Further errors.
403	<?xml version="1.0"> <message-events> <message-event> <type>MT_ACCEPTION_FORBIDDEN</type> <time>2008-09-04T11:45:08.554+0200</time>	Wrong login, pin or service.

```
    </message-event>
</message-events>
```

2.1.2. DLR - receiving

If a Callback-URL has been indicated during the transmission and if the SMS Gateway received a DLR this DLR will be reported to the Callback-URL.

Therefore, the URL is called and meanwhile a XML document is delivered as the parameter “data”. If a data for external data has been indicated during the transmission it will be reported as well.

1. *Example:*

```
<?xml version="1.0">
<message-events>
   <message-
    event>
    <type>MT_DELIVERED_TO_PHONE</type>
    <time>2008-09-04T11:45:08.554+0200</time>
    <message-id>AB10000</message-id>
    <external-data>A1231AF</external-data>
   </message-
     event>
</message-events>
```

<type>	Description
MT_DELIVERED_TO_PHONE	The message has been delivered to the end device.
MT_NOT_DELIVERED_TO_PHONE	The message could not be delivered to the end device.
MT_DELIVERED_EXTERNAL	The SMS Gateway has delivered the message to an external system in order to send it.

The receiving system (the DLR-report-receiver) has to give back a http-status code in the range 2xx in order to confirm the receiving of the DLR. Other status codes lead to new delivery-attempts until either a configured, maximum number of attempts is exceeded or alternatively a new delivery-attempt leads to a status code in the range 2xx.

In the case of “MT_NOT_DELIVERED_TO_PHONE” events it is possible to receive extended information in the detail element of the supplied DLR.

2. *Example:*

```
<?xml version="1.0" encoding="UTF-8"?>
<message-events>
  <message-event>
    <type>MT_NOT_DELIVERED_TO_PHONE</type>
    <time>2009-07-23T13:39:27.000+0200</time>
    <message-id>5410016-551011</message-id>
    <detail>80:EXPIRED</detail>
  </message-event>
</message-events>
```

Error codes for detailed element	Description
00:NOT_APPLICABLE	Reason code is not applicable
10:SUBSCRIBER_NOT_RECOGNIZED	Response from operator, the operator does not recognize the consumer

20:INSUFFICIENT_FUNDS	Response from operator, the consumer cannot fulfil the purchase
30:MESSAGEQUEUE_FULL	Response from operator, the operation is rejected
40:INVALID_CONTENT_METADATA	Response from the operator, the operation is rejected
50:CHARGING_ERROR	The charging operation cannot be performed
60:SUBSCRIBER_BLOCKED	Response from operator, the subscriber is blocked for this service.
00: NOT_APPLICABLE	Reason code is not applicable
70:SUBSCRIBER_NOT_REGISTERED	Response from operator, the subscriber must register at the operator to enable the service.
80:EXPIRED	Message could not be terminated within valid interval
90:DELETED	Message was deleted from the operator
100:HANDSET_ERROR	Message could not be terminated due to a failure of the mobile device, e.g. SIM card full
110:NETWORK_ERROR	Message could not be terminated due to a failure of the mobile operator

2.1.3. SMS receiving (MO)

Incoming SMS (MO) are redirected to the receiver's application per http POST request. As a response (response) to the request the http status codes 200 (OK) or 202 (Accepted) are expected. Additionally, the body of the response has to contain "OK", otherwise the delivery of the SMS will be restarted by a retry-mechanism.

Name	Meaning
Msg	(Alternative to the parameter data) The content of the text message.
Data	(Alternative to the parameter msg) The content of the binary message.
Snr	The source identifier of the Message
Dnr	The receiver's number of the message, e.g. the short code

Example for receiving a text message:

http://destinationApplicationURL?msg=TestNachricht&snr=491711234567&dnr=81234

2.2. SMS via SMPP

The SMPP interface of the MMG can be used for sending SMS MT messages, receiving SMS MO messages and delivery reports (DLR) when requested.

2.2.1. Confirmation

2.2.1.1. SMPP account details

There is one bind per SMPP account. If you need more connections please apply for more accounts.

Host	<MMG IP address>
Port	2775
System-ID	<system-id>
Password	<password>

System_Type	<system-type>

2.2.1.2. IP Access

To access the 4Pay SMPP server your client IP range must be announced and added to the client configuration.

2.2.2. Supported SMPP PDUs

Payload SMPP PDU types	Control SMPP PDU types
submit_sm / submit_resp	bind_transmitter / bind_transmitter_resp
deliver_sm / deliver_resp	bind_receiver / bind_receiver_resp
	bind_transceiver / bind_transceiver_resp
	enquire_link / enquire_link_resp
	unbind / unbind_resp

2.2.3. Parameter Description

2.2.3.1. Source Adress / Destination Adress

Adress Value	Type of Number (TON)		Number Plan Indicator	
49123456789	0x01	International	0x01	MSISDN
123456789	0x02	National	0x02	MSISDN
4Pay	0x05	Alphanumeric	0x00	
12345	0x00	Shortcode	0x09	Shortcode

The Type of the source address is defined by the TON and NPI parameters. Alpha numeric senders are limited to a maximum length of eleven characters MSISDN sender maximum length is 15 (see E.164).

2.2.3.2. Data Coding

Data Coding	
0x00	Default GSM
0x01	US-ASCII
0x02	Binary
0x03	Iso 8859-1 (Latin-1)
0x04	Binary
0x06	Cyrillic
0x08	UCS2/UTF-16BE

2.2.3.3. MSISDN Format

The MSISDN (maximum length is 15 digits) is built up as:

MSISDN	CC + NDC + SN
CC	Country Code

NDC	National Destination Code
SN	Subscriber Number

2.2.3.4. Enquire link

The enquire link timeout is set to 120 seconds.

2.2.4. Reason Codes

According to Appendix B Table B-1[1] you can receive the following extended delivery status information in the field "**err**"

Reason Code	Status (field: stat)	Description
000	Not applicable	Reason code is not applicable
010	UNDELIV	Response from operator, the operator does not recognize the consumer
020	UNDELIV	Response from operator, the consumer cannot fulfil the purchase
030	REJECTD	Response from operator, the operation is rejected
040	UNDELIV	Response from the operator, the operation is rejected
050	UNDELIV	The requested charging operation cannot be performed
060	REJECTD	Response from operator, the subscriber is blocked for this service.
070	UNDELIV	Response from operator, the subscriber must register at the operator to enable the service.
080	EXPIRED	Message could not be terminated within valid interval
090	DELETED	Message was deleted from the operator
100	UNDELIV	Handset error
110	UNDELIV	Network error

3. MMS

3.1. MMS transmission (MT)

The transmission of MMS messages (exclusively) takes place with the help of HTTP multipart POST Request (multipart/form-data) according to RFC 2045

3.1.1. Service URL

http://<IPAddressMMG>:8080/merlix/rest/sendmms2

The interface expects parameters according to the following chart. The block M/O states if a parameter is optional or mandatory.

[1] SMPP v3.4 Issue 1.2 -> http://www.smsforum.net/SMPP_v3_4_Issue1_2.zip

3.1.2. HTTP request parameter

Name	M/O	Bedeutung
dnr	M	The receiver's number, e.g. +491701234567
snr	M	The source identifier of the message.
subject	O	The subject heading of the MMS.
login	M	The login name of the account which is used for the transmission (is delivered by 4Pay).
pin	M	The password of the user's account (is delivered by 4Pay).
service	M	The service which is used for the transmission (is delivered by 4Pay).
callbackurl	O	URL to which the DLRs have to be reported.
externaldata	O	The data which was given by the initiator and which is included in the DLR reports; the setting of this parameter is only sensible if the callback url is set, too.

The contents of the MMS (SMIL-file, pictures, audio-files, videos, texts, etc.) are transmitted with arbitrary names in attachments.

3.1.3. MMS MT Response format

Response Code	Example http-Response	Description
200	<?xml version="1.0"> <message-events> <message-event> <type>MT_ACCEPTED</type> <time>04.09.2008 11:45:08 CEST</time> <message-id>987654321ABCD</message-id> <external-data>1234567890</external-data> </message-event> </message-events>	Message has been accepted by the MMG.
202	<?xml version="1.0"> <message-events> <message-event> <type>MT_BLACKLISTED</type> <time>04.09.2008 11:45:08 CEST</time> </message-event> </message-events>	Message will not be sent since it is on the blacklist.

Response Code	Example http-Response	Description
400	<?xml version="1.0"> <message-events> <message-event> <type>MT_INVALID</type>	An address cannot be parsed.

	<time>04.09.2008 11:45:08 CEST</time> </message-event> </message-events>	
400	<?xml version="1.0"> <message-events> <message- event> <type>MT_ACCEPTION_EXCEPTION</type> <time>04.09.2008 11:45:08 CEST</time> </message-event> </message-events>	Further errors.
403	<?xml version="1.0"> <message-events> <message- event> <type>MT_ACCEPTION_FORBIDDEN</type> <time>04.09.2008 11:45:08 CEST</time> </message-event> </message-events>	Wrong login, pin or service.

3.2. DLR - Receiving

If a Callback-URL has been indicated during the transmission and if the SMS Gateway receives a DLR, this DLR will be reported to the Callback-URL. Therefore, the URL is called and meanwhile a XML document is delivered as the parameter “data”. If a data for externaldata has been indicated during the transmission it will be reported as well.

Example:

```
<?xml version="1.0">
<message-events>
   <message-
    event>
    <type>MT_DELIVERED_TO_PHONE</type>
    <time>04.09.2008 11:45:10 CEST</time>
    <message-id>AB10000</message-id>
    <external-data>A1231AF</external-data>
   </message-
     event>
</message-events>
```

Possible Data for type are:

Type	Description
MT_DELIVERED_TO_PHONE	The message has been delivered to the end device.
MT_NOT_DELIVERED_TO_PHONE	The message could not be delivered to the end device.
MT_DELIVERED_EXTERNAL	The SMS Gateway has delivered the message to an external system in order to send it.

The receiving system (the DLR-report-receiver) has to give back a http-status code in the range 2xx in order to confirm the receiving of the DLR. Other status codes lead to new delivery-attempts until either a configured, maximum number of attempts is exceeded or alternatively a new delivery-attempt leads to a status code in the range 2xx.

3.3. MMS receiving (MO)

The MMS MO will be send to the receiver's application (target-URL) as http multipart POST Request (multipart/form-data) according to RFC 2045.

Name	Meaning
messageId	Message ID
Sender	Source identifier
Receiver	Target-number
Subject	Subject
Attachment00	Appendix 1 (picture, Soundfile, SMIL, etc.)

The attachments of the MMS MO (SMIL-file, picture, audio files, videos, texts, etc.) are taken over directly from the Multipart-Request. As a response (response) to the request the http status code 200(OK) or 202(Accepted) are expected. Additionally, the body of the response must begin with "OK" or "<status>OK</status>", otherwise the delivery of the SMS will be restarted by a retry-mechanism.

Example: Receiving MMS MO from MSISDN +491701234567 to 20002

Request:

```
POST /deliverMmsMo HTTP/1.1
User-Agent: Jakarta Commons-HttpCli-
ent/3.1 Host: localhost
Content-Length: 8244
Content-Type: multipart/form-data; bound-
ary=BUihzThAGydQUwHl7xwaWi35lfattxLzOWc

--BUihzThAGydQUwHl7xwaWi35lfattxLzOWc
Content-Disposition: form-data; name="mes-
sageId" Content-Type: text/plain; char-
set=US-ASCII Content-Transfer-Encoding: 8bit

8a95c14118987224011898758697000 3
--BUihzThAGydQUwHl7xwaWi35lfattxLzOWc
Content-Disposition: form-data;
name="sender" Content-Type: text/plain;
charset=US-ASCII Content-Transfer-Encod-
ing: 8bit

+491701234567
--BUihzThAGydQUwHl7xwaWi35lfattxLzOWc
Content-Disposition: form-data; name="re-
ceiver" Content-Type: text/plain; char-
set=US-ASCII Content-Transfer-Encoding:
8bit

20002
--BUihzThAGydQUwHl7xwaWi35lfattxLzOWc
Content-Disposition: form-data; name="sub-
ject" Content-Type: text/plain; char-
set=US-ASCII Content-Transfer-Encoding:
8bit
```

```
Test-MMS MO
--BUihzThAGydQUwHl7xwaWi35lfattxLzOWc
Content-Disposition: form-data; name="oper-
ator" Content-Type: text/plain; charset=US-
ASCII Content-Transfer-Encoding: 8bit

262-02
--BUihzThAGydQUwHl7xwaWi35lfattxLzOWc
Content-Disposition: form-data; name="attach-
ment00"; filename="fit_fill.txt"
Content-Type: text/plain; name=fit_fill.txt; charset=ISO-
8859-1 Content-Transfer-Encoding: binary


Hallo Test
--BUihzThAGydQUwHl7xwaWi35lfattxLzOWc
Content-Disposition: form-data; name="attachment01"; filename="blue-
a.jpg" Content-Type: image/jpeg; name=blue-a.jpg; charset=ISO-8859-1
Content-Transfer-Encoding:

binary ÿØÿà....<binary jpg

data>

--BUihzThAGydQUwHl7xwaWi35lfattxLzOWc
Content-Disposition: form-data; name="attach-
ment02"; filename="SMILSample10.smil"
Content-Type: application/smil; name=SMILSample10.smil; charset=ISO-
8859-1 Content-Transfer-Encoding: binary

<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<head>
    <layout>
      <region id="a" left="10%" top="10%" width="150"
        height="80" backgroundColor="green" fit="fill"/>
      <region id="text" left="10%" top="0"/>
    </layout>
 </hea
   d>
<body>
    <par dur="indefinite">
      <img src="blue-a.jpg" region="a" />
      <text src="fit_fill.txt" region="text"/>
    </par>
  </bod
    y>
</smil>

--BUihzThAGydQUwHl7xwaWi35lfattxLzOWc--
```

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache, must-revali-
date Content-Type: text/html; char-
set=ISO-8859-1 Content-Length: 21
Date: Mon, 10 Mar 2008 09:14:07 GMT

<status>OK</status>
```

4. Number Lookup

The Number Lookup service is an asynchronous http service for identifying the mobile network of a given mobile phone number (MSISDN).

4.1. Number Lookup via HTTP

4.1.1. Service - URL

The number lookup service is provided at the following

URL:(POST): http://<IPAddressMMG>/merlix/rest/numberlookup

4.1.2. HTTP Request Parameter

Name	Use	Meaning	Example
login	M	4Pay customer account name	customer42
pin	M	4Pay customer account password	qHg6cZ728
service	M	Service profile	Nltest47
msisdn	M	Mobile phone number in international format	+491701234567
callbackurl	M	The lookup result will be delivered to this URL of the customer	http://12.34.56.78/nlresult
extid	O	Custom identifier which will be included in the callback	AZ5342

4.1.3. Number Lookup Response Format

http response code	Example response	Meaning
200	+OK	Request accepted
403	Forbidden	Unauthorised acces
400	<error message>	Request rejected

4.1.4. Callback process

The result of the number lookup is sent by a HTTP POST request to the URL passed as parameter callbackurl. It consists of a simple xml document (content type is text/xml) with the elements *msisdn, mcc, mnc* and optionally *error*.

The receiving system has to give back a http-status code in the range 2xx in order to confirm the receiving of the NL callback. Other status codes lead to new delivery-attempts until either a configured, maximum number of attempts is exceeded or alternatively a new delivery-attempt leads to a status code in the range 2xx.

Element name	Meaning
MSISDN	Mobile phone number
Mcc	Mobile Country Code
Mnc	Mobile Network code
Ported	Portability status of the MSISDN (true, false, unknown)
Error	Error code (attribute) and short description, if mcc-mnc cannot be determined for a MSISDN

Example 1:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<lookupResultList>
    <lookupResult>
        <msisdn>+491701234567</msisdn>
        <mcc>262</mcc>
        <mnc>01</mnc>
        <ported>false</ported>
    </lookupResult>
</lookupResultList>
```

Example 2:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<lookupResultList>
    <lookupResult>
        <msisdn>+491701234568</msisdn>
        <error code="330">unknown subscriber</error>
    </lookupResult>
</lookupResultList
```

4.1.5. Callback error codes

Error code	Error Message	Meaning
310	Lookup Fail	Technical problem
320	Invalid MSISDN	The specified msisdn is not valid
330	Unknown Subscriber	The MSISDN is not registered in HLR

4.2. Number Lookup via SMPP

The number lookup service can be used via SMPP by sending a virtual SMS (submit_sm PDU) to the mobile phone number (MSISDN) in question. The <system-type> of the connection has to be configured exclusively for number lookup. The result of the NL will be delivered as DLR (deliver_sm PDU) in the “text” field of the shortMessage. It can contain the status “lookup failed” in case of an error, the status “unknown subscriber” if the MSISDN is not registered or the MCC, MNC and ported status of the MSISDN.

Port	Description
T	MSISDN is ported
F	MSISDN is not ported
U	Ported status is unknown

Example for non ported NLR results

shortMessage of DLR

id:0010153721 sub:001 dlvrd:001 submit date:1105271458 done date:1105271747 stat:DELIVRD err:000 text: MCC=262 MNC=01 PORT=F

4.3. Bulk Number Lookup via Mail

The Bulk Number Lookup Service is an asynchronous mail (SMTP) service for identifying the mobile network of a list of mobile phone numbers (MSISDN). The mail must contain a list of MSISDNs as attachment. The result will be sent back to the originator of the mail and contains a CSV document as attachment.

4.3.1. Mail Format

The originator's email address must be registered for Number Lookups in the MMG by 4PAY. The destination address will be provided by 4PAY after registration. The subject, body and the name of the attachment does not matter. The mail must contain one attachment: a text file with one MSISDN (international format) per line.

Example for attachment textfile:

```
+491701234567 // +491701234568 // +491771234569
```

4.3.2. Mail response

The response contains a text file in CSV format as attachment. The value separator is a comma and each line consists of 4 fields: MSISDN, MCC, MNC and portability status (true / false / unknown). If the network cannot be identified for a MSISDN then the MCC field contains the error message and the following fields are empty.

Example for mail response:

```
123456, invalid msisdn, , // +491731234567, 262,02, false
+491771234568, 262,03, false // +491731234569, unknown subscriber, ,
+491701234561, 262, 07, true
```