# Table of Contents

1. Introduction to OTP SMS Service – HTTP Interface  
   Page 3

2. Requirements for OTP SMS Implementation with HTTP  
   Page 3

3. Technical Overview  
   3.1 General Interface Information  
   3.2 tyntec Specific Interface Information  
   Page 4

4. Detailed Communication Process  
   Page 4

5. OTP SMS HTTP Features  
   5.1. GSM Return codes in OTP SMS HTTP responses  
   5.2. Buffered Notifications  
   5.3. Protocol ID / Over the Air (OTA)  
   5.4. Validity Period  
   5.5. HTTPS Connection  
   5.6. swapDLR  
   5.7. TON / NPI settings  
   5.8. Geographic time Stamp  
   5.9. Alphanumeric sender ID  
   5.10. Dynamic Sender ID replacement Feature  
   5.11. Concat SMS Header  
   5.12. WAP Push  
   5.13. SMS Sorter  
   Page 7

6. Example Java Programs  
   6.1. Sending an SMS using the HTTP interface  
   6.2 Reading the OTP automatically from the SMS on Android  
   Page 11

7. Glossary of Terms  
   Page 17

8. Frequently Asked Questions  
   Page 17

9. Contact Information  
   Page 18
1. Introduction to OTP SMS Service – HTTP Interface

The One-Time Password SMS HTTP is a service which enables you to send high quality SMS to all operators listed on tyntec’s network list using the HTTP protocol. This service is appropriate for mid-level usage (50,000 – 200,000 SMS sent / month). The service also allows you to determine in real-time when a message is sent to invalid numbers and to notify your users to correct the numbers in order to receive the OTP.

![Figure 1: The OTP SMS communication process with HTTP](image)

The arrows show the direction of the data flow. Each arrow is labeled with the data type carrying the data. The essential data that is transported appears in brackets following the data type. A detailed overview of the communication process is provided in section 5 of this document.

2. Requirements for OTP SMS Implementation with HTTP

To implement and use the OTP SMS service with HTTP, the customer should have certain resources available that are not provided by tyntec. For example, a developer must be available to plan and implement:

- The sending of an HTTP request that transports the SMS to tyntec’s platform. ([http://www.w3.org/Protocols/rfc2616/rfc2616.html](http://www.w3.org/Protocols/rfc2616/rfc2616.html)).
- Optional: if Delivery Receipts are required, a service must be embedded into the customer’s web server (e.g. by using a PHP page or Java servlet or any other dynamic HTTP service).

There are sample Java programs at the end of this document which can be used as a guideline for the developer.

The customer requires a computer system with a fixed IP address from where the SMS sending is initiated, and where optionally the customer’s HTTP Server is running and accepts the Delivery Receipts. This computer system must be reachable from the tyntec IP.

For the fastest implementation and expert support from tyntec, it is strongly recommended that the OTP SMS with HTTP is implemented using:
3. Technical Overview

3.1 General Interface Information

HTTP...
...is an industry standard.
...it is unproblematic towards firewall conflicts.
...is easy to implement/maintain.
...is accepted worldwide.

3.2 tyntec Specific Interface Information

Using the HTTP protocol it is possible to submit SMS to tyntec’s messaging platform either via HTTP method GET or POST. Sender, receiver and content are transmitted via individual and mandatory parameters.

Optional features can be requested by using non-mandatory parameters. The optional features are the sending of DLRs [Delivery Receipts] to a URL after the message was delivered or failed and the inclusion of the message reference in the HTTP response (see Figure 1, Step 2). In case a DLR [Delivery Receipt] is requested, it will be submitted to the customer’s HTTP Server using a URL which is specified by the customer during the initial set up to the service. The HTTP method of this request is POST.

4. Detailed Communication Process

The following section describes each step of the communication process. Each step is described individually with examples of the possible URL that may be used. For an overview of the process, please refer to Figure 1.

Step 1 - Sending message (sender, receiver, content)

The SMS is transmitted by calling an URL with 5 mandatory parameters:

- user: the username provided by tyntec.
- password: the password provided by tyntec.
- sender: the value of this parameter will be shown as the originator of the message on the mobile handset.
- receiver: the receiver of the message in international format, i.e. with a leading “+” or “00”. Please keep URL encoding of the plus sign in mind.
- content: the content of the message.

There are optional parameters available upon request:

- requestDLR: to request a delivery receipt please add this parameter to the HTTP request.
showMsgIDs: to request a message id please add this parameter to the HTTP request.

messageType: possible values:
- "sms_flash": The message will be shown on the display of the handset right away and usually not be stored in the handset.
- "sms_unicode": The message is encoded in the unicode characterset. The characters of the content parameter of the HTTP request must already be Hex encoded. For OTA messages, the protocol id feature [see §5] should be activated.
- "sms_raw": used for all types of binary messages. The content of the content parameter of the HTTP request must already be Hex encoded. For OTA messages, the protocol id feature [see §5] should be activated.
- "sms_concat": the short message is a part of a concatenated message. The content of the message must contain a concat header to ensure that all short messages will be displayed on the handset as one message. Please see § 6 for details on the concat header.

dataCoding ( &dataCoding= ):
Default datacoding for messagetype sms_simple is Latin-1(ISO-8859-1).
For characters encoded in GSM7 encoding please use datacoding _0’.
For characters encoded in ASCII please use datacoding _1’.
For characters encoded in Latin-1 encoding please use datacoding _3’.
For unicode sms please use messagetype sms_unicode and datacoding _8’.
For binary sms please use messagetype sms_raw and datacoding _4’.

Notes on time format for parameters deliverStart and deliverEnd:

The format of the time must be given in the following format: YYYYMMDDhhmmss[+-]<GMT-Offset>, meaning:
- "YYYY" is the year in 4 digits
- "MM" is the month in digits, with _01_ being January
- "DD" is the day of the month, ranging from _01_ to _31_
- "hh" is the hour, based on a 24 hour time format, e.g. 10 PM being 22
- "mm" is the minute of the hour
- "ss" is the second
- [+-]<GMT-Offset> is the GMT-Offset of the specified time in _hhmm_, i.e. GMT+2 would be specified as _+0200_.
  The time has to be URL-Encoded, i.e. a _+“_ has to be converted to _%2b_ and a _-_ to _%2d_.

Time format example 1: Setting the deliverStart to January 2nd 2014, 10 AM UTC time would be:
20140102100000%2b0000.

Time format example 2: Setting the deliverStart to May 18th 2014, 2:36 PM CEST time would be:
20140518143600%2b0200.
This is an example URL including request of messageid and delivery receipt - please note that the login "testuser" with password "testpassword" is not valid on the tyntec system, also TON/NPI settings must be
activated in order to use TON/NPI parameters:

http://smpp.tyntec.com:4887/http/send?user=testuser&password=testpassword&messagetype=sms_simple&senderTON=1&senderNPI=1&receiver=0044762123123123&sender=00444762123123&content=This%20is%20a%20test&showMsgIDs&requestDLR

Please note that all parameters have to be properly URL encoded. For further information please refer to the following URL: http://www.w3schools.com/tags/ref_urlencode.asp

Step 2 - Acknowledge message (msg ref)

tyntec will acknowledge the receipt of a transmitted message. Please note that this response will be delivered via the HTTP response to the previous HTTP request.

If the transmitted message has been received successfully, tyntec will acknowledge the transmitted message with a message reference which should be stored within the customer’s implementation of the interface. The HTTP response will contain a small HTML answer which looks like this:

<HTML>
<HEAD><TITLE>Tech-On-Air sender</TITLE></HEAD>
<BODY>
OK
<p>MessageID(s):<br/>
55956-1352352532413+85291228760
<br/>
</BODY></HTML>

If the message has been received but contains an error, the HTTP response contains an error code instead of the string "OK". The full list of return codes can be seen in the following table:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not enough data</td>
<td>One or more of the mandatory parameters is not set correctly. Please check if all mandatory parameters are included in the URL and the characters are properly URL encoded.</td>
</tr>
<tr>
<td>Invalid password</td>
<td>The username or password supplied is not correct or the initial amount of test messages is used up. Please contact Customer Relations to check the username and password or the amount of test messages left.</td>
</tr>
<tr>
<td>Internal Error</td>
<td>tyntec's server is currently unable to process the message. Please retry later.</td>
</tr>
</tbody>
</table>

Step 3 - Sending Delivery Receipt (msg-ref, result)

tyntec employs five message states to determine the outcome of a message. The table below shows the message states and their descriptions.

Note: The four digit GSM return codes are also returned, please consult your GSM return codes document for code descriptions.
Following an example HTTP request including a Message State; this HTTP request will be sent to an URL provided by the customer.

http://www.exampleurl.com/submitdate=0701201055&sender=00444762123123&stat=DELIVRD&err=0000&msgid=5555-77665544%2b44762123123123&receiver=%2b44762123123123&donedate=0701201005&text=Hello%20World

Please note that customer is responsible for the availability of this URL.

Step 4 - Acknowledge Delivery Receipt

Receipt of DLRs must be acknowledged to tyntec via a HTTP 200 (OK) response. Please note that this response must be delivered via the HTTP response to the previous HTTP request.

If there is an issue in the processing of the response from tyntec, a message indicating an error should be returned. For example, returning the following message will trigger a retry from tyntec. Any error code of the 4xx range will do.

Receipt of DLRs must be acknowledged by your HTTP server by terminating the stream. Failure to terminate the stream will result in a re-delivery from tyntec.

5. OTP SMS HTTP Features

There are additional features available for OTP SMS HTTP that can be activated upon customer request. For an overview of all features provided in the OTP SMS HTTP service (included by default or activated upon customer request), please refer to the Appendix 1 – Feature List.

5.1. GSM Return codes in OTP SMS HTTP responses

This feature gives a detailed answer to why a message failed to be delivered; for example, the message could not be delivered because the mobile phone was switched off.

The GSM return codes are four-digit hexadecimal codes. A list of GSM error code values and their associated descriptions is provided in a separate document. The following example shows the error for a phone that was switched off the whole delivery period:

http://www.exampleurl.com/msgid=77665544%2b44762123123123&submitdate=0701201055&donedate=070122
You can use the GSM return codes to identify invalid phone numbers. This way you can determine in real-time when your users provide a wrong number, e.g. as a result of a typo or their landline which is not SMS enabled. This feature gives you the opportunity to display a message in your app or on your website asking your user to correct the number in order to receive the OTP. Please refer to the GSM return codes document provided by tyntec for more details.

5.2. Buffered Notifications

The buffered notification informs the customer of the reason for the unsuccessful delivery after the first delivery attempt. In conjunction with the GSM Return codes, (see 6.1), it provides a good hint at the reason why a message was not delivered. The buffered notification follows the format of the DLR. In case of a buffered notification the Message State is set to “BUFFERD”.

5.3. Protocol ID / Over the Air (OTA)

This feature enables the setting of the protocol id as per GSM 3.40 documentation. By using this feature, customers can send over the air configuration messages such as software updates, configuration settings and security locks.

5.4. Validity Period

Customers can use the Validity Period feature for time-restricted SMS messages, e.g. One-Time Passwords, which can be valid only for a certain short time period. The Validity Period will be specified by setting an absolute time when to start and when to stop delivering the message. The names of the parameters are:

"deliverStart": when to start delivering the message
"deliverEnd": when to stop delivering the message

The format of the time is YYYYMMDDhhmmss[+-] <GMT-Offset>, meaning:

- "YYYY" is the year in 4 digits
- "MM" is the month, with "01" being January
- "DD" is the day of the month, ranging from "01" to "30"
- "hh" is the hour, based on a 24 hour watch
- "mm" is the minute
- "ss" is the second
- <GMT offset> is the GSM-offset of the specified item in "hhmm", i.e. GMT+2 would be specified as "+0200". The time has to be URL-encoded, i.e. a "+" has to be converted to "%2b" and a "-" to "%2d".

Example:
The time "2008-04-29 13:30:00" specified in GMT+2 would be "20081232235959%2d0800"
The time "2008-12-31 23:59:59" specified in GMT-8 would be "20081232235959%2d0800"

5.5. HTTPS Connection

On request the customer can connect to a SSL-encrypted HTTP service, thus ensuring secure transmission of sensitive data. This requires the customer’s application to support SSL encryption [already built in into Java and the HTTP Client library]. Furthermore the following connection parameters change:

- the host to connect to is smpp.tyntec.com. Be aware not to use the IP address since this will lead to a domain mismatch error in your application.
the port to connect to is 8443.

All other parameters that are used for sending a message remain the same. This means that the example from chapter 5 step 1 changes in case of a HTTPS connection to:

https://smpp.tyntec.com:8443/http/send?user=testuser&password=testpassword&receiver=0044762123123123&sender=00444762123123&content=Hello%20World&showMsgIDs

5.6. swapDLR

In the tyntec DLRs, the content of the sender and receiver parameters are set to the same values as the matching message. If this feature is activated, the contents are swapped, i.e. the content of the sender parameter in the original message becomes the receiver in the DLR.

5.7. TON / NPI settings

If this feature is activated, tyntec checking of the TON / NPI settings for plausible values is de-activated. This allows the customer to set the originator settings - the number as national format (07852 900460) instead of international +447852 900460. This feature is useful to recognize the characteristics of the number (international, national, alphanumeric, ISDN, etc.).

5.8. Geographic time Stamp

This feature adapts the time stamp to the local time of the home network. It is particularly useful for international mobile service providers and enterprises, which need to have the exact sending time adapted to the local time. (Please note this is not linked to the MSC, so if the subscriber is roaming the message will not show local time, it only shows the subscribers home network time).

5.9. Alphanumeric sender ID

This feature allows specifying the originator field freely for each message. This feature is needed in order to send messages with an alphanumeric sender ID e.g. tyntec 1. The length of an alphanumeric sender ID is between 1-11 characters.

5.10. Dynamic Sender ID replacement Feature

This feature can change an alphanumeric Sender ID or a shortcode to a normal MSISDN with international TON/ NPI (1/1). It is particularly useful for networks who handle alphanumeric Sender IDs or shortcodes more sensitively. Please contact your account manager for further details.

5.11. Concat SMS Header

The first six bytes of the content of a concatenated message contain information on the parts of the message.

<table>
<thead>
<tr>
<th>Byte</th>
<th>Content</th>
<th>Value, Hex encoded</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Total header length</td>
<td>05</td>
</tr>
<tr>
<td>2</td>
<td>Header type: concat 8 bit</td>
<td>00</td>
</tr>
<tr>
<td>3</td>
<td>Header length of 8 bit concat header</td>
<td>03</td>
</tr>
<tr>
<td>4</td>
<td>Reference number, used by the handset for identifying the same message</td>
<td>Anything, as long it fits into one byte and is Hex encoded. It has to be the same throughout all parts of one concatenated message</td>
</tr>
<tr>
<td>5</td>
<td>Total number of parts in the concat message</td>
<td>Total number of parts, Hex encoded</td>
</tr>
<tr>
<td>6</td>
<td>Part number of the short message</td>
<td>Part number of the short message, Hex encoded.</td>
</tr>
</tbody>
</table>
A typical concat header would look like this: 050003010201 for part one of the message and 050003010202, leaving 153 characters in the first part and 153 in the second part.

**Example requests:**

Part 1:
http://smpp.tyntec.com:4887/http/send?user=testuser&password=testpassword&messagetype=sms_concat&showMsgIDs=true&receiver=0044762123123123&sender=00444762123123123&content=050003010201aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
hex encoding for “tyntec”

00 terminate string with 00
01 <indication>
01 <si> true

5.13. SMS Sorter

If this feature is activated, the customer can provide an identifier (HTTP parameter) of their choice for each message sent to tyntec. The “SMS Sorter” feature stores this identifier while the SMS is transmitted. This identifier is then returned in the DLR for each message. This feature can assist customers in billing their partners. It also assists large enterprises in tracking, differentiating and sorting their SMS traffic per for example, traffic type, country, department, product, etc.

In order to provide a self selected identifier, customers have to add the parameter „userContext” to the HTTP request.

6. Example Java Programs

6.1. Sending an SMS using the HTTP interface

HTTP request:
import java.io.*;
import java.net.*;
import java.util.*;
import java.util.regex.Pattern;

import sun.misc.BASE64Encoder;

/**
 * Java program that sends SMS using the tyntec HTTP interface
 */
public class HTTPSenderServletCaller {

    public static void main(String[] args) throws Exception{
        String username = args[0];
        String password = args[1];
        String sender = args[2];  // prefixed with +
        String receiver = args[3]; // prefixed with +, if number
        String content = args[4];
        String ip = args[5];
        String port = args[6];

        // Building Request-String from given data
        StringBuffer dataBuffer = new StringBuffer();
dataBuffer.append(“user=“);
dataBuffer.append(URLEncoder.encode(username, “ISO-8859-1“));
dataBuffer.append(“password=“);
dataBuffer.append(URLEncoder.encode(password, “ISO-8859-1“));
dataBuffer.append(“receiver=“);
dataBuffer.append(URLEncoder.encode(receiver, “ISO-8859-1“));
dataBuffer.append(“content=“);
dataBuffer.append(URLEncoder.encode(content, “ISO-8859-1“));
dataBuffer.append(“sender=“);
dataBuffer.append(URLEncoder.encode(sender, “ISO-8859-1“));
dataBuffer.append(“ip=“);
dataBuffer.append(ip);
dataBuffer.append(“port=“);
dataBuffer.append(port);

    }
dataBuffer.append(URLEncoder.encode(receiver, "ISO-8859-1"));
dataBuffer.append("&content=");
dataBuffer.append(URLEncoder.encode(content, "ISO-8859-1"));
dataBuffer.append("&sender=");
dataBuffer.append(URLEncoder.encode(sender, "ISO-8859-1"));
dataBuffer.append("&showMsgIDs=true");

// Creating the connection
URL url = new URL("http://" + ip + ":" + port + "/http/send");
URLConnection con = url.openConnection();
con.setDoOutput(true);

// Sending the data
OutputStreamWriter writer =
    new OutputStreamWriter(con.getOutputStream());
writer.write(dataBuffer.toString());
writer.flush();

// Reading the whole answer from the URL to a string
BufferedReader reader =
    new BufferedReader(new InputStreamReader(con.getInputStream()));
StringBuffer output = new StringBuffer();
String line = null;
do {
    line = reader.readLine();
    if (line != null) output.append(line);
} while (line != null);

// Search MessageID, deleting all HTML tags with a regexp
Pattern pattern = Pattern.compile("<\/[a-zA-Z]*>");
String[] responses = pattern.split(output.toString());

// Searching for OK and the text introducing the IDs
List responseList = Arrays.asList(responses);
if (responseList.contains("OK")){
    int idx = responseList.indexOf("MessageID(s):");  
    System.out.println("MessageID: "+ responses[idx+1]);
}
else{
    System.err.println("Error: " + output.toString());
}

HTTP response:

import java.io.*;
import java.util.*;
import java.net.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.net.*;
import javax.servlet.http.*;

public class DeliveryReceiptServlet extends HttpServlet {
    /**
      *
* Handle an incoming POST request.
*/
public void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws IOException {
    doGet(req, resp);
}

/**
 * Handle an incoming GET request.
 */
public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws IOException {
    try {
        // The message ID that was given back on send time
        String msgID = URLDecoder.decode(req.getParameter("msgid"), "UTF-8");

        // The date when the message was delivered to the platform
        Date submit = new Date(Long.parseLong(req.getParameter("submitdate")));

        // The date when the message reached its final state (normally it
        // reached the handset)
        Date done = new Date(Long.parseLong(req.getParameter("donedate")));

        // One of DELIVRD, FAILED or EXPIRED
        String stat = URLDecoder.decode(req.getParameter("stat"), "UTF-8");

        // Numerical value according to stat
        String err = URLDecoder.decode(req.getParameter("err"), "UTF-8");

        // The first characters of the original message
        String text = URLDecoder.decode(req.getParameter("text"), "UTF-8");

        // The original sender of the message
        String src = URLDecoder.decode(req.getParameter("sender"), "UTF-8");

        // The original receiver of the message
        String dest = URLDecoder.decode(req.getParameter("receiver"), "UTF-8");
        resp.setStatus(HttpServletResponse.SC_OK);
    } catch (Exception e) {
        resp.setStatus(HttpServletResponse.SC_BAD_REQUEST);
    }
    resp.flushBuffer();
}
6.2 Reading the OTP automatically from the SMS on Android

On the Android OS it is possible to access the incoming SMS from an app and to read the OTP (One-Time Password) directly instead of asking the user to type it. This improves the customer experience and speeds up the OTP authentication process. It also increases the conversion rates, since you will remove an extra step that might have otherwise interrupted the authentication or registration process. You will find below some sample code that you can integrate into your Android app in order to read the OTP from the incoming SMS automatically.

/**
* @author tyntec
*
* Interface that is used to handle received SMS by a
* dedicated android receiver object
*
*/
private interface IMessageReceiver {
/**
* Called for each SMS received
*
* @param from originating phone number
* @param text message text
*/
    void onMessage(String from, String text);
}

/**
* Method that is called when user wants to pass and authorization
* @param v
*/
private void onAuthButtonClick(View v)
{
    setUiInitialState();
    // Get a phone number from either android app widget or settings
    String phoneNumber = getPhoneNumber();
    // Send auth request based on phone number
    sendAuthRequest(phoneNumber);
    // Show a message or start an indication of the auth processing
    notifyUserRequestSended();
}

/**
* This method must be called before the application exit
*/
private void unregisterReceiver()
{
    if (receiver == null)
        return;
```java
getApplicationContext().unregisterReceiver(receiver);
receiver = null;
}

/**
 * Registers special receiver object in the Android’s infrastructure and
 * configures the object to forward received messages into the app.
 */
private void registerReceiver()
{
    IntentFilter ifilter = new IntentFilter();
    // Use number higher than 999 if you want to be able to stop processing and not to put
    // auth messages into the inbox.
    ifilter.setPriority(1000);
    ifilter.addAction("android.provider.Telephony.SMS_RECEIVED");
    // Create and hold the receiver. We need to unregister on shutdown
    receiver = createReceiver();
    getApplicationContext().registerReceiver( receiver, ifilter );
}

private SmsReceiver createReceiver() {
    return new SmsReceiver(new IMessageReceiver() {
        @Override
        public void onMessage(String from, String text) {
            processMessage(from, text);
        }
    });
}

/**
 * Method that is called on every SMS receive.
 * It needs to accept only valid originating phone number
 * @param from originating phone number
 * @param text message text
 */
private void processMessage(String from, String text) {
    if (isValidSender(from))
        // Auth procedure
        performAuth(text);
}

/**
 * Class for receiving SMS
 * Required to be as a separate object for the Android’s infrastructure.
 * Once SMS received it forwards it to IMessageReceiver object passed in the class’s constructor
 * @author tyntec
 */
public class SmsReceiver extends BroadcastReceiver {
    private static final String SMS_EXTRA_NAME = "pdus";
```
private final IMessageReceiver messageReceiver;
public SmsReceiver(IMessageReceiver messageReceiver) {
    if (messageReceiver == null)
        throw new IllegalArgumentException("messageReceiver");

    this.messageReceiver = messageReceiver;
}

@Override
/**
 * Method that handles SMSs.
 * It iterates over all the messages and forward each method to IMessageReceiver object.
 */
public void onReceive(Context context, Intent intent) {
    Bundle extras = intent.getExtras();

    if (extras != null) {
        Object[] smsExtra = (Object[]) extras.get(SMS_EXTRA_NAME);
        for (int i = 0; i < smsExtra.length; ++i) {
            SmsMessage sms = SmsMessage.createFromPdu((byte[]) smsExtra[i]);
            String body = sms.getMessageBody().toString();
            String address = sms.getOriginatingAddress();
            messageReceiver.onMessage(address, body);
        }
    }
}

// WARNING!!!
// Uncomment this if you do not want the SMS put into the inbox (for priority > 999)
// this.abortBroadcast();
}
7. Glossary of Terms

- Asynchronous: An interaction is said to be asynchronous when the associated messages are chronologically and procedurally decoupled.
- MCC: Mobile Country Code consisting of 3 digits, e.g. 234 for UK.
- MNC: Mobile Network Code consisting of 2 or 3 digits, e.g. 58 for Manx Telecom.
- MSISDN: Mobile Subscriber ISDN Number.
- GSM 3.40: ETSI document for technical realization of the Short Message Service (SMS).
- HTTP: Hyper text transfer protocol.
- Unicode: A character encoding standard.
- DLR: Delivery Receipt. A message that is delivered by tyntec to the customer and contains information on a certain message, e.g. the time it was delivered to the handset.
- HTTP request: A request sent to a web server by a client using the HTTP protocol. In this case the HTTP request contains all necessary data for a short message to be delivered to a mobile phone.
- HTTP response: The response to an HTTP request sent to a client by a web server. It contains the status of the issued request. In this case, it may also contain the tyntec message reference.
- RFC: Request for Comments (RFC) documents are a series of memoranda encompassing new research, innovations, and methodologies applicable to Internet technologies. RFC documents are issued with a unique serial number.
- SMPP: Short Message Peer to Peer (P2P). A protocol used for transmitting short messages.
- OTP SMS: One-Time Password SMS

8. Frequently Asked Questions

Q: I cannot send the SMS messages via HTTP to the URL provided by tyntec.
A: Check firewall settings to ensure that requests to URLs with a port number other than 80 are indeed permitted. If in doubt, please connect to the following URL (http://support.tyntec.biz:4887) and report to Customer Relations the date and time of the attempted connection.

Q: Why do I receive an authorization failure?
A: An authorization failure can be caused by an incorrect username and / or password. Please contact Customer Relations to verify your user details.
If you have a test account and have previously been able to send messages successfully you may have reached your test message limit.

Q: How do I know when my test message limit has been reached?
A: If you attempt to send messages beyond your test account limit, you will receive a HTTP response stating «Invalid password» (see §4 step 2).

Q: I send messages but always get an error code back.
A: This can be caused by a number of reasons:
The supplied username and/or password is invalid: Please check if you have mistyped the username and/or password provided to you.
The supplied number is not a valid phone number: the number must only consist of digits (except a leading +).
Any other characters including spaces will not be accepted.
## Appendix 1 – Feature List

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Needs activation?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery Receipt (DLR)</td>
<td>Provides a delivery report for sent messages.</td>
<td>No</td>
</tr>
<tr>
<td>Binary content</td>
<td>A binary SMS message has 140 bytes of used data. This user data can be split into two parts: the user data header and the actual data. The user data header can be used to inform mobile phones about certain message types. This way SMS can be used to carry operator logos, ringtones, telephone settings and WAP-Push messages.</td>
<td>No</td>
</tr>
<tr>
<td>Unicode</td>
<td>Provides SMS sending in different language alphabets.</td>
<td>No</td>
</tr>
<tr>
<td>Concatenated</td>
<td>Allows sending of concatenated messages. All parts of the concatenated message have to be sent independently to tyntec.</td>
<td>No</td>
</tr>
<tr>
<td>Flash SMS</td>
<td>Message appears directly on the phone's screen, instead of being allocated in the Inbox. It's a useful alternative to normal SMS when you want to catch the recipient's attention immediately.</td>
<td>No</td>
</tr>
<tr>
<td>WAP-Push SMS</td>
<td>Specially formatted SMS messages that display an alert message to the user, and gives the user the option of connecting directly to a particular URL via the mobile phone's browser. This allows the sending party to enable the viewing and/or downloading of complex data by specifying a web link within a regular SMS format.</td>
<td>No</td>
</tr>
<tr>
<td>MNP check for each message (before sending)</td>
<td>Checks the true home network of a mobile number before routing the message. It's an essential feature for countries where Mobile Number Portability is in place.</td>
<td>No</td>
</tr>
<tr>
<td>Flexible retry scheme</td>
<td>Minimum of 6 Retries conducted in the time frame set by deliverStart and deliverEnd parameters. The exact scheme can be explained by the Customer Relations manager.</td>
<td>No</td>
</tr>
<tr>
<td>Validity Period</td>
<td>Customers use the Validity Period feature for time-restricted SMS messages, e.g. One-Time Passwords, which can be valid only for a certain short time period. This feature is conducted in the time frame set by deliverStart and deliverEnd parameters.</td>
<td>No</td>
</tr>
<tr>
<td>swapdlr</td>
<td>In the tyntec DLRs, the content of the sender and receiver parameters are set to the same values as the matching message. If this feature is activated, the contents are swapped, i.e. the content of the sender parameter in the original message becomes the receiver in the DLR. If this feature is activated, the originator and receiver number in the DLR to match SMPP 3.4 documentation.</td>
<td>Yes</td>
</tr>
<tr>
<td>TON/NPI settings</td>
<td>Type of Number/Numbering Plan Indicator - This allows the customer to set the originator settings - the number as national format (+7852 900460) instead of international +447852 900460. This feature is useful to recognize the characteristics of the number (international, national, alphanumeric, ISDN, etc.).</td>
<td>Yes</td>
</tr>
<tr>
<td>Protocol ID / Over the Air (OTA)</td>
<td>Sends OTA configuration messages, such as software updates, configuration settings and security locks. It's particularly useful for Mobile Device Management purposes.</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Needs activation?</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>GSM return codes in DLRs</td>
<td>Give a more detailed answer to why a response failed. This feature gives standard GSM return codes and tyntec codes, giving more detailed information about the status of the message.</td>
<td>Yes</td>
</tr>
<tr>
<td>Buffered Notifications</td>
<td>Informs the customer of the reason for the unsuccessful delivery after the first SMS delivery attempt. The “Buffered Notification” is sent in the format of a DLR.</td>
<td>Yes</td>
</tr>
<tr>
<td>Geographic Time Stamps</td>
<td>Adapts the time stamp to the local time of the home network. It is particularly useful for international mobile service providers and enterprises, which need to have the exact sending time adapted to the local time. (Please note this is not linked to the MSC, so if the subscriber is roaming the message will not show local time, it only shows the subscribers home network time).</td>
<td>Yes</td>
</tr>
<tr>
<td>Numeric Originator,</td>
<td>Allows specifying the originator field freely for each message. This can allow a targeted, clear identity for sending to opt-in subscribers, and also facilitate 2-way exchange with them if a short code or a long number is set in this way. International e.g. +447852 910046, short codes e.g. 80080 and alphanumeric e.g. tyntec 1. The length is 1-11 characters.</td>
<td>Yes</td>
</tr>
<tr>
<td>Alphanumeric or Short Code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Originator individual on each message</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alphanumeric Sender ID</td>
<td>This feature can change an alphanumeric Sender ID to a normal MSISDN with international TON/NPI (1/1). It is particularly useful for networks who handle alphanumeric Sender IDs more sensitively. Please contact your account manager for further details.</td>
<td>Yes</td>
</tr>
<tr>
<td>Replacement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMS Sorter</td>
<td>If this feature is activated, the customer can provide an identifier (HTTP parameter) of their choice for each message sent to tyntec. - The SMS sorter feature stores this identifier while the SMS is transmitted. - This identifier is then returned in the DLR for each message. - This feature can assist customers in billing their partners. - It also assists large enterprises in tracking, differentiating and sorting their SMS traffic per for example, traffic type, country, department, product etc...</td>
<td>Yes</td>
</tr>
</tbody>
</table>