myMSP API v2

System Interface - Specifications

21st Century Mobile
Marketing & Sales
+46 (0)8 21 21 55
sales@21st.se

www.21st.se
Research & Development
+46 (0)8 21 21 55
tech@21st.se

1 Purpose of Document 3
1.1 Author 3
1.2 Document Status 3
1.2.1 Document Changes 3
1.3 Document Perimeter 4
2 Overview 5
3 Communication with myMSP using an API 6
3.1 The send message sequence 6
3.2 Message status 7
3.3 Result codes 7
3.4 Send message example 8
3.5 Retrieve message example 9
4 Available APIs 10
4.1 HTTPS 10
4.2 Web service 10
4.3 SMPP 10
4.4 HTTP push (for delivery reports and incoming messages) 10
4.5 Device Push Notifications (DPN) 11
4.6 URL shortener 11
5 Glossary 12

1 Purpose of Document

The purpose of this document is to define the API interfaces made available by myMSP to access the application functionality. Accessing myMSP can be done from the computer systems of external parties/customers. This document has several appendixes which describe each of the available API:s in more detail.

1.1 Author

21st Century Mobile
Mikael Rosvall
mikael.rosvall@21st.se
+46 (0)8 21 21 55

1.2 Document Status

- Published 2007-01-08

1.2.1 Document Changes

- Since API v1
  - Separated the API document into several separate documents:
    - myMSP API v2 (an overview of the myMSP APIs)
    - myMSP API v2 Appendix ws (describes the Web Service API)
    - myMSP API v2 Appendix http (describes the HTTP API)
    - myMSP API v2 Appendix xml (describes the XML API)
  - Expanded the Web Service API document greatly
- B [2007-06-15]
  - Minor changes and improvements
- C [2007-08-20]
  - Added a note that the maximum length of a sender alias is ten characters.
- D [2007-12-14]
  - Updated the headers and footers with new information.
  - Updated the result codes in chapter 3.3
  - Added a note that the maximum length of a sender alias is eleven characters.
- E [2008-03-28]
  - Updated all URL-references from the old domain (www.mymsp.eu) to the new domain (mymsp.21st.se).
- F [2009-04-29]
  - Added a short description of the SMPP-API to 4.4
- G [2009-12-09]
  - Added a description of the push API to 3.1, 3.4, 3.5 and 4.5.
- H [2011-08-24]
  - Added a new result code: 503 (3.3)
- I [2012-10-26]
  - Added information regarding Device Push Notifications (DPN) to the document.
- J [2019-03-11]
  - Updated graphic profile and URL:s.
- K [2020-04-14]
  - Added information regarding URL shortener to the document.

1.3 Document Perimeter

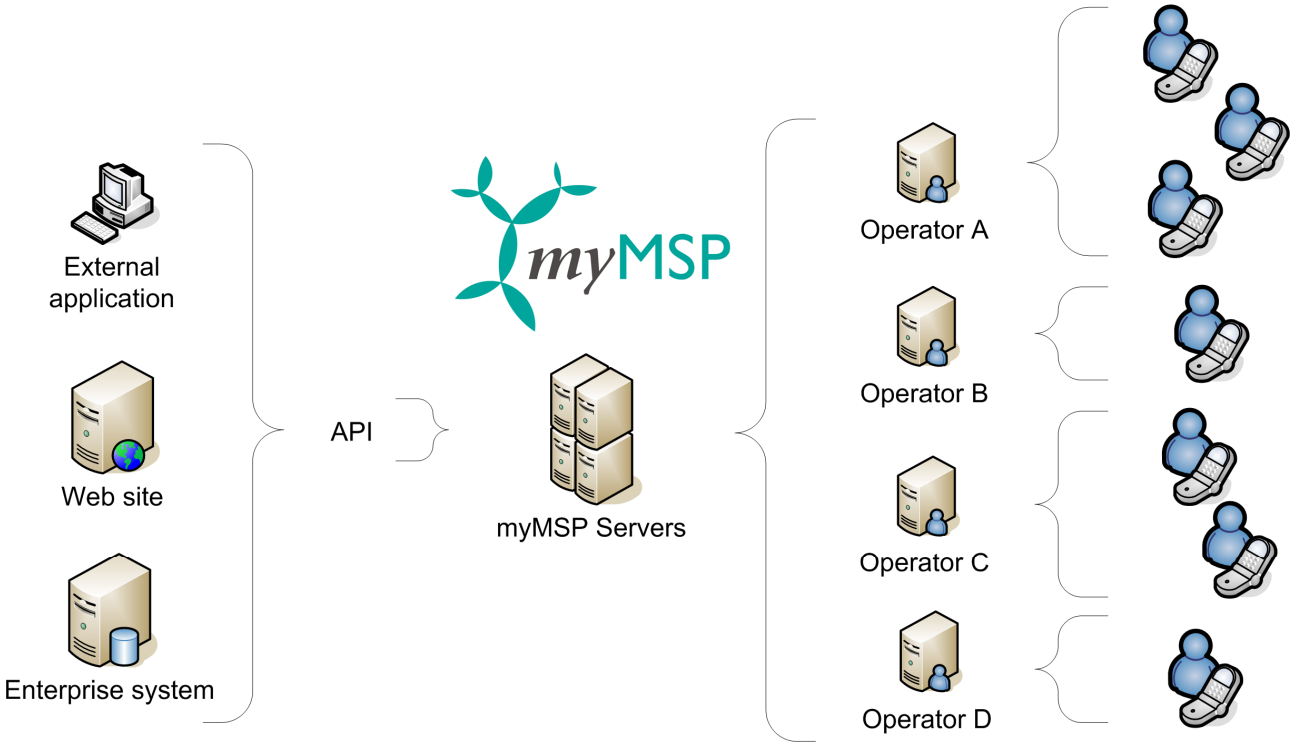
Neither economical questions nor questions regarding agreements/contracts will be dealt with in this document.

The information given in this document may change without notice and describes only the matters defined in the general part of this document. Please verify that your company has the most recent version. This information is intended for the use of customers and partners of 21st Century Mobile. The information or statements given in this document concerning the suitability, capacity or performance of the mentioned service cannot be considered binding but shall be defined in the agreement concluded between 21st Century Mobile and the customer, if applicable. 21st Century Mobile shall not be responsible in any event for errors in this document or for any damages, incidental or consequential (including monetary losses), that might arise from the use of this publication or the information in it. This material and the service described in this document are copyrighted in accordance with the applicable laws.

2 Overview

The myMSP application is a platform for structured mobile message management that enables communication between multiple mobile phones and different types of computer systems, using both two-way SMS, voice messages (VOI) and Device Push Notification (DPN) communication. myMSP is directly connected to the major domestic mobile operators that handle the actual sending and receiving of the mobile messages over the phone net.

Anyone using myMSP can easily connect their own computer systems, e.g. an ERP or CRM system, to myMSP through several Application Programming Interfaces (API).



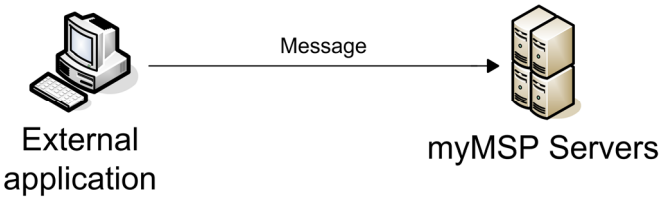
3 Communication with myMSP using an API

When accessing myMSP through an API it is important to have a basic understanding of how myMSP operates. The three central functions of myMSP are: *send messages*, *check the status of a sent message* and *retrieve incoming messages*. Different API:s can access these services in different ways. However, all API:s have a core set of concepts in common which are described here.

3.1 The send message sequence

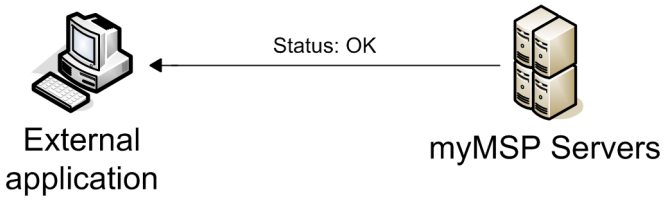
The time it takes for a message to be sent by the operator can be considerable and therefore the transmission sequence is a two-step process:

1. The message, with its intended recipients, is sent to myMSP. If the message is accepted (i.e. it is correctly formatted) then it is assigned a message number by myMSP and placed in a send queue.

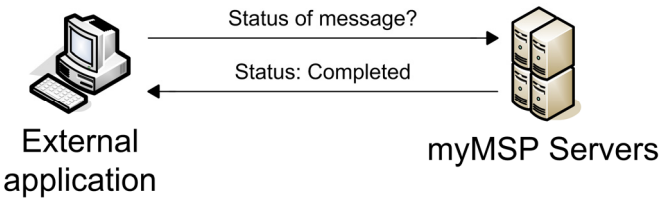


2. The send status of a message can be obtained by the external application in two ways:

    a. myMSP transmits the send status of a message to the external application. Updates are only sent when the send status has changed (e.g. a message has been delivered).



    b. The external application sends a request to myMSP querying the send status of a message. A response is returned containing the current send status of the message (e.g. delivered). This step can be repeated until it has been confirmed that the message has been sent to all recipients.



3.2 Message status

A message has a status (state) indicating the send progress of a message:

- 0 Created (myMSP has received the send request and accepted it)
- 1000 Sending (the message is being sent)
- 1100 Completed (the message is no longer being sent)
- 1200 Failed (the message could not be sent)

It is important to realize that the message may be sent to the operator in fragments, and that the acceptance is received per recipient (mobile number). A message with multiple recipients will not always be delivered to all recipients (for instance if a recipient's mobile number is invalid). When a message is no longer being transmitted to its recipients it will achieve a message status of 1100 (Completed), even if not all recipients have received the message. Information regarding the number of messages that have been successfully sent to recipients, delivered, and read is reported by myMSP. The message status should not be confused with result codes described below.

3.3 Result codes

The result code indicates the success of invoking a myMSP service. Example: a send request is sent to myMSP and the result code 0 is returned indicating that myMSP has accepted the request. If the result code is not 0 then something is wrong, e.g. the password is invalid (code 202). Note that the HTTP-API uses a slightly different method of indicating the success of a request, and that different codes apply (these are described in the HTTP appendix document).

The result codes are:

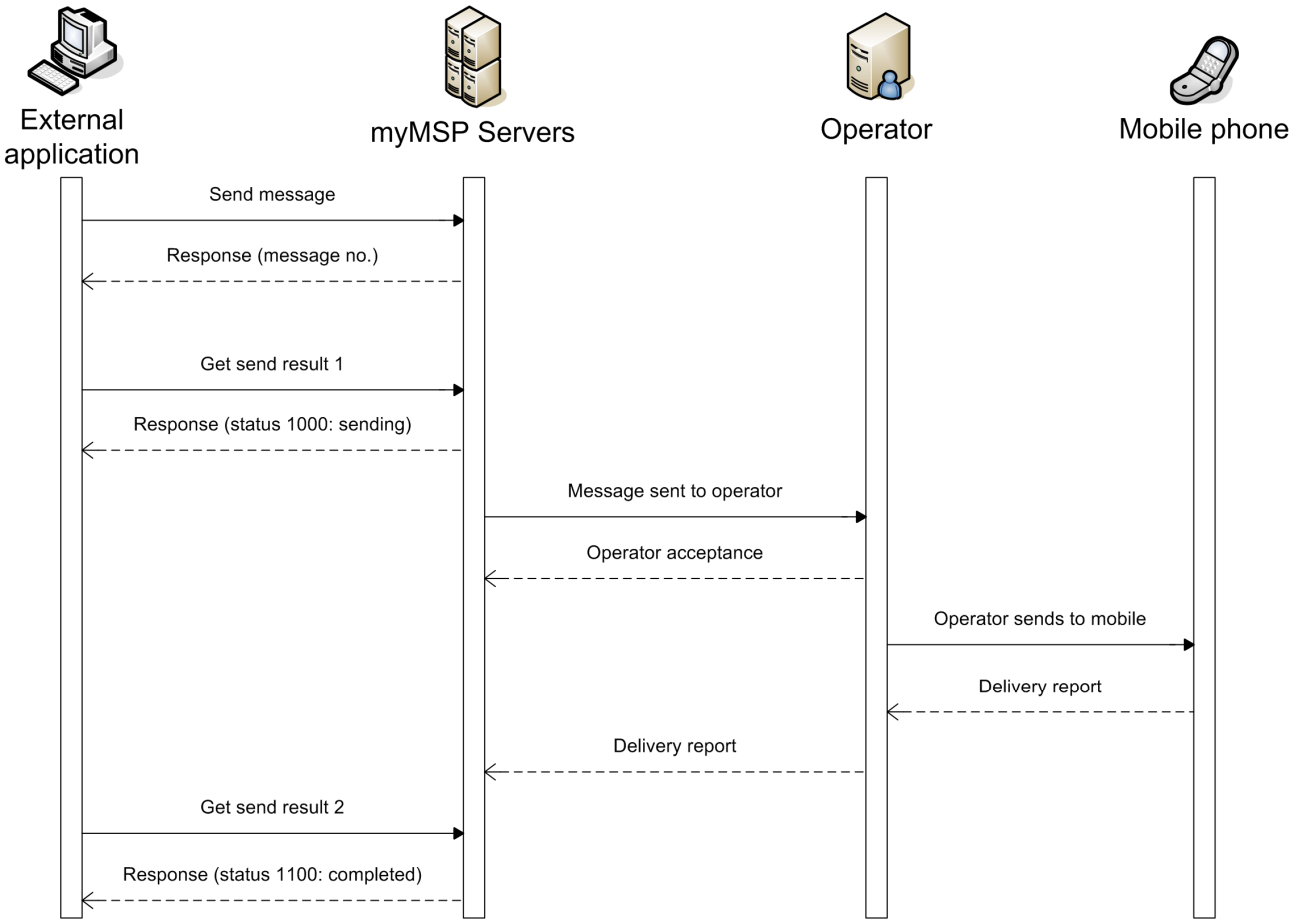
- 0 OK
- 100 User not owner of message!
- 101 User information missing!
- 102 Unknown message type
- 103 Recipient information missing!
- 104 Missing message text!
- 105 Message text longer than 765 characters!
- 106 Missing message data!
- 107 Message subject longer than max 40 characters!
- 108 Wrong message format!
- 109 Invalid time format (expected yyyyMMddHHmmssSSS)!
- 110 Invalid time (expected value later then 2004-01-01 00:00:00)!
- 201 Failed authorization, missing user
- 202 Failed authorization, bad password for user
- 211 Failed client validation, missing client type!
- 212 Failed client validation, invalid client type
- 213 Failed client validation, expected client version string, found
- 221 Failed client validation, missing client version!
- 222 Client validation, a newer version is available!
- 223 Failed client validation, not allowed version
- 301 Failed adding recipients to message!
- 401 Failed adding content to message!
- 501 Failed saving message!
- 502 Limit maximum messages exceeded!
- 503 Not enough credits left to send message!
- 601 Unknown message number!
- 701 Unknown request!

- 901 Unknown myMSP error, contact support!
- 902 Unknown error, contact support!

3.4 Send message example

The figure below illustrates a typical transmission sequence using the querying method for obtaining the send status (3.1 2a). For an example on how the sequence changes when delivery reports are pushed by myMSP to the external application see the push appendix document.

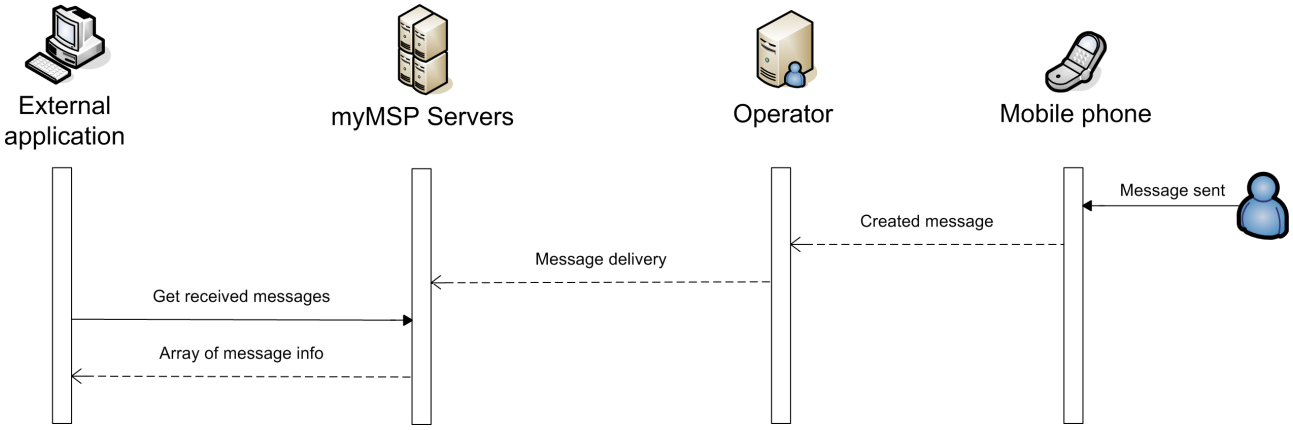
An external application connects to myMSP through an API and transmits a *send message*-request containing a message and a list of recipients. A myMSP server receives the message and returns a *response* that includes the message number that the new message has been given. The external application then checks the status of the message by sending the message number to the server in a *send result*-request, and receives a response from the server in the form of a status number. The status number 1000 indicates that myMSP has placed the message in the send queue. myMSP sends the message to the recipients through their respective mobile phone-operators. When an operator has accepted a message to a recipient's number an *operator acceptance*-response is sent to myMSP. The operator then sends the message to the recipient's phone and a *delivery report* to myMSP. When the message has been sent to all operators it is considered completed and receives a status of 1100. This status, along with the delivery status of each recipient, is returned to the external application when it transmits a second send result-request to myMSP.



3.5 Retrieve message example

The figure below illustrates an incoming message sequence using the querying method for obtaining the send status. For an example on how the sequence changes when incoming messages are pushed by myMSP to the external application see the push appendix document.

The figure below illustrates an incoming message sequence. A mobile phone user that received the message from the previous example decides to send a reply. The operator receives the message and sends it to myMSP. In the text-body of the message the user has included an incoming ID (In-ID) which makes it possible for myMSP to direct the message to the right myMSP-account(s). When the external application send a *retrieve received messages*-request the mobile phone-user's message is returned to the application along with any other messages that have been redirected to the myMSP user's account.



4 Available APIs

The following APIs are currently available. Note that the APIs are constantly being improved and new functionality added in accordance with customer needs. Please contact us (tech@21st.se) if you find that the functionality of an API is lacking or if you would like to access myMSP through an API that is not listed here, and we will do our up most to accommodate you.

Appendix documents with more detailed descriptions of the APIs listed below can be downloaded from our file library which you can find here: https://assets.21st.se/docs/API_v2/

4.1 HTTPS

An external system can connect to myMSP through HTTPS by sending a series of parameters using the POST method (POST is a HTTPS-method and is not related to e-mail).

Currently supports:

- Send SMS
- Send VOI (voice messages)
- Check send status of a message
- Retrieve incoming SMS

4.2 Web service

Web Services are application services or functions that are made available through the Internet. The information exchange is carried out using XML-documents that are sent through the HTTP-protocol and WSDL (Web Services Description Language) is used to describe what services are available and how they are accessed. WSDL is constructed to enable an automatic information and service exchange between computers.

Currently supports:

- Send SMS
- Check send status of a message
- Retrieve incoming SMS

4.3 SMPP

The Short Message Peer-to-Peer (SMPP) protocol is the most common protocol for sending SMS messages in use today. myMSP supports SMPP V3.4.

Currently supports:

- Send SMS
- Receive send status of a message
- Receive incoming SMS

4.4 HTTP push (for delivery reports and incoming messages)

myMSP can transmit to an external system through HTTP or HTTPS using the GET method to submit the data. The push interface is used to send data, like incoming messages and delivery status updates, directly to an external system thus eliminating the need for that system to constantly poll myMSP for updates. The myMSP HTTP push interface is designed to be compatible with the Web service interface, i.e. it directly

maps the Web service classes to request parameters. However, it is also possible to use the push interface in conjunction with the HTTPS-API:s, but not with the SMPP interface.

Currently supports:

- Receive send status of a message
- Receive incoming SMS

4.5 Device Push Notifications (DPN)

Device push notifications (DPN) make it possible to send lightweight messages through the Internet to apps running on, primarily, mobile devices such as smart phones and tablet computers. What sets notifications apart from other forms of mobile communication methods such as SMS and e-mail is that they can be targeted at an individual app running on a specific device. Notifications can be used to send simple text messages or to inform the app that there is new data to be fetched from an external server. myMSP's notification technology enables developers to send notifications to apps through a HTTP API without having to know what operating system the app is running on.

Currently supports:

- Add/remove subscriptions (app)
- Send notifications (external server)

4.6 URL shortener

The URL shortener API is used to create short URLs that forwards all request to longer URLs. This is an efficient way to reduce the length of an SMS message. myMSP uses a default domain name when creating short URLS but custom domains can also be configured. Contact you 21st sales representative to activate this feature for your account.

5 Glossary

Content	The information to be communicated. For message type SMS the content is text.
DPN	A DPN (Device Push Notification) is a lightweight message sent over the Internet to an app running on a device (e.g. a smart phone).
In-ID	When a mobile phone user wants to send a message to myMSP an incoming ID (In-ID) is required. The ID enables myMSP to direct the incoming message to the correct user account.
Message	An entity (container) with some content and an originator who wishes to communicate with one or more recipients.
Message type	An indicator which restricts what content a message may contain. Examples are SMS, VOI and DPN.
Mobile phone number	Mobile Phone Numbers can be represented in various formats. For example, when dealing with a national Swedish number, like 0730430525, the following formats are valid: 0046730430525, +46730430525 and 46730430525.
myMSP	The Mobile Solutions Platform provided by 21st Century Mobile.
Sender Alias	Se Originator Text below.
Operator	The operator, or mobile operator, is the organization responsible for the actual transmission of messages from myMSP to recipients' mobile phones and from mobile phones to myMSP, utilizing their own telephone network. Examples are Tele2, TeliaSonera, Telenor and Tre.
Originator	An originator is the sender of a message, normally a mobile number or a short number.
Originator Text	When a message is received on a mobile phone the originators number is displayed. This number can be replaced by an alias (originator text). For example: 46778899 can be displayed as 'companyx' or another alias. However, recipients can not reply to a message that uses an alias. Only aliases that have been submitted, accepted and registered in myMSP can be used. This functionality is however operator dependent since not all of them supports originator text. The maximum length of an alias is eleven (11) characters.
Recipient	A recipient is the target of the message identified either by recipient id, recipient name or a mobile number. The name may have been created automatically from the mobile number by myMSP.
SMS	A SMS (Short Message Service) can only have content of type text.
VOI	Voice message. The message is delivered by telephone call and the text is read to the receiver.

myMSP API v2 Appendix HTTPS

System Interface - Specifications

21st Century Mobile
Marketing & Sales
+46 (0)8 21 21 55
sales@21st.se

www.21st.se
Research & Development
+46 (0)8 21 21 55
tech@21st.se

1 Purpose of Document 3
1.1 Author 3
1.2 Document Status 3
1.2.1 Document Changes 3
1.3 Document Perimeter 4
2 Technical description 5
3 Usage scenario 6
3.1 Send a message 6
3.2 Check the status of a sent message 6
3.3 Receive a message 7
4 HTTPS methods 9
4.1 Send SMS (sendSms) 9
4.1.1 Input parameters 9
4.1.2 Output parameters (positive result) 9
4.1.3 Output parameters (negative result) 9
4.2 Retrieve send results (getSmsResult) 10
4.2.1 Input parameters 10
4.2.2 Output parameters (positive result) 10
4.2.3 Output parameters (negative result) 11
4.3 Delivery time 11
4.3.1 Get delivery time (sendlater/get) 11
4.3.1.1 Input parameters 11
4.3.1.2 Output parameters (positive result) 11
4.3.1.3 Output parameters (negative result) 12
4.3.2 Get the delivery time for all scheduled messages (sendlater/getall) 12
4.3.2.1 Input parameters 12
4.3.2.2 Output parameters (positive result) 12
4.3.3 Output parameters (negative result) 12
4.3.4 Update delivery time (sendlater/update) 13
4.3.4.1 Input parameters 13
4.3.4.2 Output parameters (positive result) 13
4.3.4.3 Output parameters (negative result) 13
4.3.5 Cancel scheduled message (sendlater/cancel) 13
4.3.5.1 Input parameters 13
4.3.5.2 Output parameters (positive result) 13
4.3.5.3 Output parameters (negative result) 14
4.4 Retrieve received messages (getMsgReceived) 14
4.4.1 Input parameters 14
4.4.2 Output parameters (positive result) 14
4.4.3 Output parameters (negative result) 15
4.5 Retrieve uncollected delivery reports and received messages (getMsgUpdates) 15
4.5.1 Input parameters 15
4.5.2 Output parameters (positive result) 15
4.5.3 Output parameters (negative result) 16
5 Examples 17
5.1 ASP 17
5.2 HTML 17
5.2.1 Send SMS 17
5.2.2 Retrieve SMS 18

1 Purpose of Document

The purpose of this document is to define the HTTPS interface made available by myMSP to access the application functionality. Accessing myMSP can be done from the computer systems of external parties/customers.

1.1 Author

21st Century Mobile
Mikael Rosvall
mikael.rosvall@21st.se
+46 (0)8 21 21 55

1.2 Document Status

- Published 2007-01-08

1.2.1 Document Changes

- B [2007-06-15]
  - Minor changes and improvements
- C [2007-12-11]
  - New method added: Retrieve messages
    - 3.3: Receive a message
    - 4.3: Retrieve received messages
- D [2008-03-28]
  - Updated all URL-references from the old domain (www.mymsp.eu) to the new domain (mymsp.21st.se).
  - Changed the description of the recipients-field in chapter 4.1.1 to include recipient groups.
- E [2009-04-29]
  - New parameter added: delivery time (4.1.1 and 4.1.3)
- F [2009-12-07]
  - New parameter added: charset (4.1.1)
- G [2010-12-17]
  - New parameter added: isflash (4.1.1)
  - New parameter added: destinationport (4.1.1)
- H [2011-08-24]
  - New error code added: 12 (4.1.3)
- I [2012-10-26]
  - Updated error codes (4.1.3, 4.2.3, 4.4.3)
- J [2015-01-28]
  - New methods added: Delivery time (4.3)
- K [2015-09-09]
  - Added “GET” to the supported HTTP methods in the Technical description (2)
- L [2018-11-02]
  - Removed error code 301.
- M [2019-03-11]
  - Updated graphic profile.
  - New methods added: Retrieve uncollected delivery reports and received messages (4.5)

1.3 Document Perimeter

Neither economical questions nor questions regarding agreements/contracts will be dealt with in this document.

The information given in this document may change without notice and describes only the matters defined in the general part of this document. Please verify that your company has the most recent version. This information is intended for the use of customers and partners of 21st Century Mobile. The information or statements given in this document concerning the suitability, capacity or performance of the mentioned service cannot be considered binding but shall be defined in the agreement concluded between 21st Century Mobile and the customer, if applicable. 21st Century Mobile shall not be responsible in any event for errors in this document or for any damages, incidental or consequential (including monetary losses), that might arise from the use of this publication or the information in it. This material and the service described in this document are copyrighted in accordance with the applicable laws.

2 Technical description

This interface is realised through HTTPS. An external system can send data to myMSP through HTTPS using the GET or POST methods to submit data. The results are delivered as a HTTPS RESPONSE in the “text/plain” format. Each function is accessed through a specific URL, i.e. to send a SMS the following URL is used:

- https://mymsp.21st.se/external/sendSms

3 Usage scenario

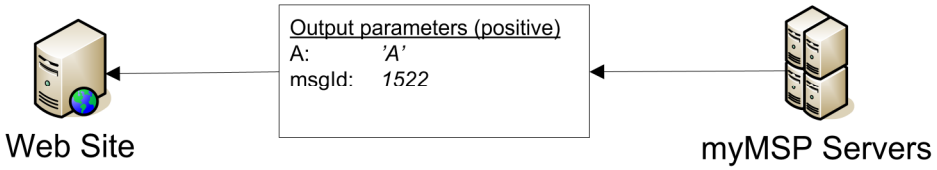
The following scenarios are intended to give a brief overview of how to use the myMSP HTTPS interface. In the next chapter the API is described in more detail.

3.1 Send a message

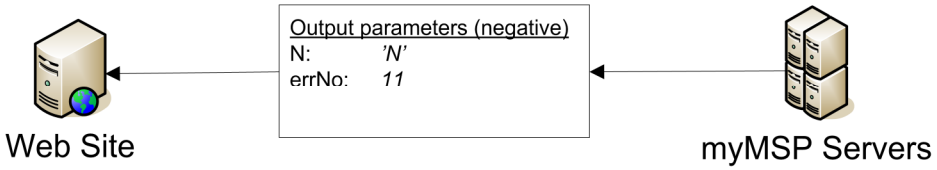
As previously mentioned in the main API-document (3.1 The send message sequence) sending a message through myMSP is done in steps.

1. First, a message *send request* is sent to myMSP. The request contains user identification (user name and password), a string of the recipients' numbers, and the message content.

2. After the send request has been sent, a set of parameters is returned.
   a. If myMSP has accepted the request then a set of positive parameters is returned, including the ID that myMSP has assigned the message.



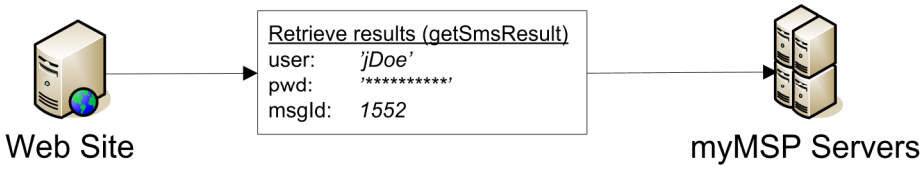
   b. If myMSP has rejected the request (e.g. the password for the user was incorrect) then a set of negative parameters are returned, including the specific code of the error.



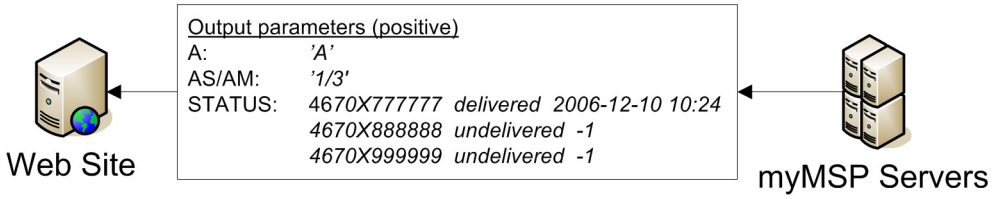
3.2 Check the status of a sent message

It can take some time before a message is delivered to all recipients. The status of a message can be checked while it is being delivered.

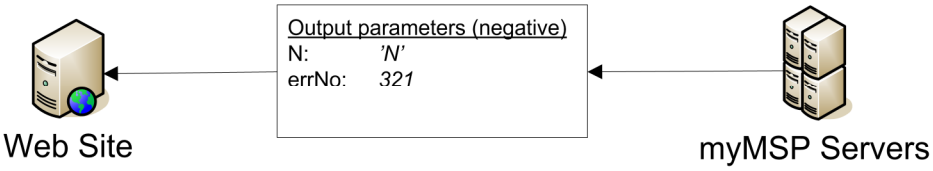
1. To check the current status of a message a *retrieve result request* has to be sent to myMSP. The request is composed of user identification, and the message number of the message whose status should be checked.



2. After the retrieve result request has been sent, a set of parameters is returned.
   a. If myMSP has accepted the request then a set of positive parameters is returned, including information regarding the send status for each recipient, e.g. if the message has been successfully sent to a specific recipient.



   b. If myMSP has rejected the request then a set of negative parameters are returned, including the specific code of the error.

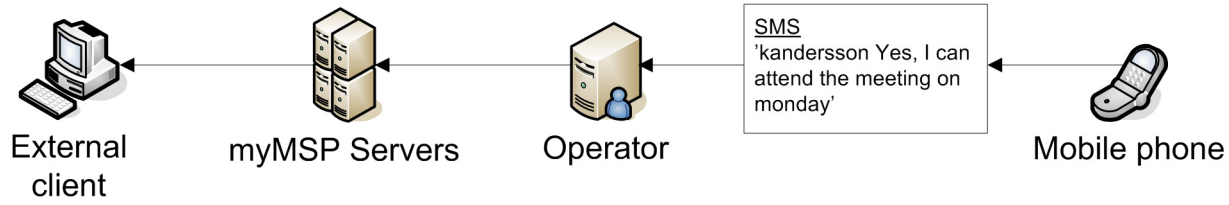


The process checking the send status of a message can be repeated as many times as is desirable, e.g. to check and make sure that all recipients have received the message.

3.3 Receive a message

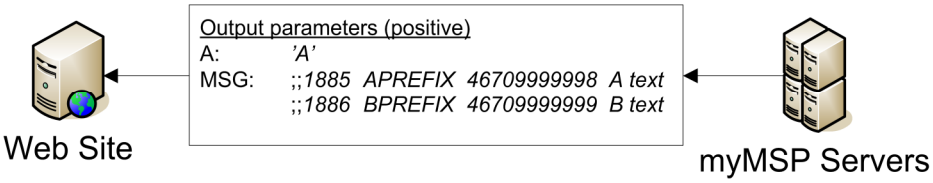
myMSP can also receive messages from mobile phones.

1. An initial ID is needed in order to send a message to a myMSP account from a mobile phone. The initial ID is placed in the beginning of the text in the message and should be followed by a blank space.

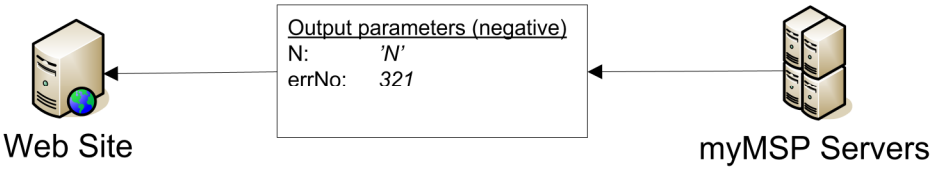


2. A *received message request* is needed in order to get new messages that have been sent to a myMSP account.

3. After the received message request has been sent, a set of parameters is returned.
   a. If myMSP has accepted the request then a set of positive parameters is returned, including the received messages.



   b. If myMSP has rejected the request then a set of negative parameters are returned, including the specific code of the error.



4 HTTPS methods

This chapter describes the various HTTPS methods in more detail.

4.1 Send SMS (sendSms)

Send a message to recipients. Note that the web page character encoding, e.g. “charset=UTF-8”, is used to set the character set of the message.

Connection: https://mymsp.21st.se/external/sendSms

4.1.1 Input parameters

user	User Account
pwd	Password
originator	Originator text of the message (see ‘Originator text’ in main API-document glossary). Note that only originator texts that have been pre-registered with myMSP can be used.
recipients	Phone Number of recipients or name/short name of a recipient group created in the myMSP web GUI (https://mymsp.21st.se/app). Should be in the form of a comma-separated string, e.g. “46709888888,46702111111,recipgroupA”.
msg	The message in plain text. Maximum of 765 characters.
deliverytime	The date and time that the message should be delivered. Should be formatted as follows: YYYYMMDDhhmmzz, e.g. 200904291230+1. The last field (zz) is the time zone difference in hours from Western European Time (UTC+0) i.e. GMT. +0 or -0 equals GMT, +1 equals Central European Time (UTC+1), etc.
charset	The character set that was used to encode the message text. If this parameter is not set then ‘ISO-8859-1’ used to decode the message.
isflash	Set to “true” if the message should be sent as a flash SMS.
destinationport	If set and larger then zero then the message is sent to the specified port on the receiver’s mobile phone.

4.1.2 Output parameters (positive result)

A	Character ‘A’
msgId	Message identification number in myMSP. Used when querying send results.

4.1.3 Output parameters (negative result)

N	Character ‘N’
errNo	The error code: 7: Authorization missing 11: Maximum number of messages exceeded 12: Not enough credits left to send message 13: Unknown error, contact support 22: No value in parameter ‘user’ 23: Parameter ‘user’ missing

	24: No value in parameter 'pwd' 25: Parameter 'pwd' missing 28: One or more of the recipient numbers are in the wrong format 282: No value in parameter 'recipients' 29: Parameter 'recipients' missing 30: No value in parameter 'msg' 31: Parameter 'msg' missing 33: Sender alias is not registered 34: Invalid format for delivery time 35: Invalid date for delivery time (must be after current time) 36: Invalid flash value 37: Invalid destination port value 38: Invalid express value 303: Invalid UCP message

4.2 Retrieve send results (getSmsResult)

Check the send result of a message.

Connection: https://mymsp.21st.se/external/getSmsResult

4.2.1 Input parameters

user	User Account
pwd	Password
msgId	Message identification number in myMSP.

4.2.2 Output parameters (positive result)

A	Character 'A'
R/S/D	The number of recipients (R), sent messages (S) and delivered messages (D). For example '10/10/9' means that a message with ten recipients has been sent to as ten SMS (one to each recipient) and that nine of the messages have been delivered.
STATUS	The delivery status for each recipient of the message is listed. The different fields of a delivery status are separated by a tab-character ( ). The fields are: *Recipient Number;* *Sent time (YYYY-MM-DD hh:mm);* *Delivery status;* *Delivery time (YYYY-MM-DD hh:mm);* If the message has still not been sent to a recipient, minus one (-1) is shown instead of the sent time. If the message has still not been delivered to a recipient, minus one (-1) is shown instead of the delivered time. Example: 4670X666666 2007-12-02 15:14 delivered 2007-12-02 15:15

	4670X777777	2007-12-02 15:15	delivered	2007-12-02 15:18
	4670X888888	2007-12-02 15:15	undelivered	-1
	4670X999999	-1	undelivered	-1

4.2.3 Output parameters (negative result)

N	Character 'N'
errNo	The error code: 7: Authorization missing 13: Unknown error, contact support 22: No value in parameter 'user' 23: Parameter 'user' missing 24: No value in parameter 'pwd' 25: Parameter 'pwd' missing 32: Can't find message with msgId 323: Message has been archived 391: Parameter 'msgId' is invalid 392: No value in parameter 'msgId'

4.3 Delivery time

4.3.1 Get delivery time (sendlater/get)

Gets the delivery time of a message that has been sent to myMSP.

Connection https://mymsp.21st.se/external/sendlater/get

4.3.1.1 Input parameters

user	User Account
pwd	Password
msgno	Message identification number in myMSP.

4.3.1.2 Output parameters (positive result)

A	Character 'A'
STATUS	The delivery status of the message. The fields are separated by a tab-character ( ). The fields are: *Message ID;* *Delivery time (YYMMDDhhmmssZZZZ);* If the message does not have a delivery time an empty string is returned in the delivery time field. Example: 12556 150112153000004+

4.3.1.3 Output parameters (negative result)

N	Character 'N'
errNo	The error code: 7: Authorization missing 13: Unknown error, contact support 22: No value in parameter 'user' 23: Parameter 'user' missing 24: No value in parameter 'pwd' 25: Parameter 'pwd' missing 32: Can't find message with msgno 391: Parameter 'msgno' is invalid 392: No value in parameter 'msgno'

4.3.2 Get the delivery time for all scheduled messages (sendlater/getall)

Gets the delivery times of all scheduled messages that have not yet been sent by myMSP.

Connection https://mymsp.21st.se/external/sendlater/getall

4.3.2.1 Input parameters

user	User Account
pwd	Password

4.3.2.2 Output parameters (positive result)

A	Character 'A'
STATUS	The delivery status of all scheduled messages that have not yet been sent. The fields are separated by a tab-character ( ). The fields are: *Message ID;* *Delivery time (YYMMDDhhmmssZZZZ);* Example: 12556 150112153000004+ 12557 150115094500004+

4.3.3 Output parameters (negative result)

N	Character 'N'
errNo	The error code: 7: Authorization missing 13: Unknown error, contact support 22: No value in parameter 'user' 23: Parameter 'user' missing 24: No value in parameter 'pwd' 25: Parameter 'pwd' missing

4.3.4 Update delivery time (sendlater/update)

Updates the delivery time of a scheduled messages that has not yet been sent by myMSP.

Connection https://mymsp.21st.se/external/sendlater/update

4.3.4.1 Input parameters

user	User Account
pwd	Password
msgno	Message identification number in myMSP.
deliverytime	The date and time that the message should be delivered. See 4.1.1 for more details.

4.3.4.2 Output parameters (positive result)

A	Character 'A'

4.3.4.3 Output parameters (negative result)

N	Character 'N'
errNo	The error code: 7: Authorization missing 13: Unknown error, contact support 22: No value in parameter 'user' 23: Parameter 'user' missing 24: No value in parameter 'pwd' 25: Parameter 'pwd' missing 32: Can't find message with msgno 34: Invalid format for delivery time 35: Invalid date for delivery time (must be after current time) 351: Can't update delivery time, message already sent 391: Parameter 'msgno' is invalid 392: No value in parameter 'msgno'

4.3.5 Cancel scheduled message (sendlater/cancel)

Cancels a scheduled message.

Connection https://mymsp.21st.se/external/sendlater/cancel

4.3.5.1 Input parameters

user	User Account
pwd	Password
msgno	Message identification number in myMSP.

4.3.5.2 Output parameters (positive result)

A	Character 'A'

4.3.5.3 Output parameters (negative result)

N	Character 'N'
errNo	The error code: 7: Authorization missing 13: Unknown error, contact support 22: No value in parameter 'user' 23: Parameter 'user' missing 24: No value in parameter 'pwd' 25: Parameter 'pwd' missing 32: Can't find message with msgno 351: Can't cancel message, message already sent 391: Parameter 'msgno' is invalid 392: No value in parameter 'msgno'

4.4 Retrieve received messages (getMsgReceived)

Fetch received messages.

Connection: https://mymsp.21st.se/external/getMsgReceived

4.4.1 Input parameters

user	User Account
pwd	Password
lastMsgId	Message identification number in myMSP. All messages with a higher message ID (i.e. messages received later) are returned. Input 0 to fetch all received messages.
clean	Set to "true" to wash all return values, i.e. all new line characters (\r\n, \n, \r) are replaced with the string "\n" and all tab characters (\t) with a space.

4.4.2 Output parameters (positive result)

A	Character 'A'
STATUS	All received message with a msgID-value higher than the one supplied in the request are listed. Two semicolon (;;) are placed in front of each message. The different fields of a received message are separated by a tab-character ( ). The fields are: *Message ID;* *Received date (YYYY-MM-DD hh:mm);* *User name;* *Prefix used/Device notification action;* *Message type (SMS/VOI/DPN);* *Originator number;* *Originator alias/Recipient ID (if DPN);* *MMS-subject; (deprecated)* *Message text*

	Example: ;;12556 2007-12-02 15:14 jDoe APREFIX SMS 46769999999 null null A text ;;12554 2007-08-19 12:52 jDoe APREFIX SMS 46769999998 null null B text

4.4.3 Output parameters (negative result)

N	Character 'N'
errNo	The error code: 7: Authorization missing 13: Unknown error 22: No value in parameter 'user' 23: Parameter 'user' missing 24: No value in parameter 'pwd' 25: Parameter 'pwd' missing 391: Invalid 'msgno' value 392: Parameter 'msgno' missing

4.5 Retrieve uncollected delivery reports and received messages (getMsgUpdates)

Gets all uncollected delivery reports and received messages for an account. Subsequent calls to this method only returns updates that have happened since the last call. Contact support to activate this function.

Connection: https://mymsp.21st.se/external/getMsgUpdates

4.5.1 Input parameters

user	User Account
pwd	Password

4.5.2 Output parameters (positive result)

A	Character 'A'
STATUS	All uncollected delivery reports and received messages are listed. Two semicolon (;;) are placed in front of each delivery report/message. The different fields are separated by a tab-character ( ). The fields for delivery reports are: *Type (value "0" for delivery reports);* *Message ID;* *Recipient ID;* *Mobile no;* *Delivery status;* *Delivery time (YYMMDDhhmmssZZZZ);* The fields for received messages are: *Type (value "1" for received messages);* *Message ID;*

	Originator number; *Received date (YYYY-MM-DD hh:mm);* *Prefix used/Device notification action;* *Message type (SMS/VOI/DPN);* *Message text* Example: ;;0 12556 98765 46769999999 delivered 2007-12-02 15:14 ;;1 12554 46769999998 2019-03-01 12:52 APREFIX SMS B text

4.5.3 Output parameters (negative result)

N	Character 'N'
errNo	The error code: 7: Authorization missing 13: Unknown error 22: No value in parameter 'user' 23: Parameter 'user' missing 24: No value in parameter 'pwd' 25: Parameter 'pwd' missing

5 Examples

Here are some code examples that illustrate how to use the HTTPS API to communicate with myMSP.

5.1 ASP

The results are delivered as “HTTPS RESPONSE” in the ”text/plain” format directly after myMSP has received the message (and all recipients). In other words, the results are shown in text with information on how the message was handled. In order to process this information, for example save it to a database, some form of written code is necessary.

```
<%
'Uses WinHTTP v5.1 from Microsoft
'http://msdn.microsoft.com/library/en-us/winhttp/http/winhttp_start_page.asp

Dim objWinHttp
Dim strHTML
Dim strReq

strReq = "user=" & "myUsername" & _
        "&pwd=" & "myPWD" & _
        "&originator=" & "mySignature" & _
        "&recipients=" & "recipient phone numbers" & _
        "&msg=" & "message text"

Set objWinHttp = Server.CreateObject("WinHttp.WinHttpRequest.5.1")
objWinHttp.Open "POST", " https://mymsp.21st.se/external/sendSms"

objWinHttp.SetRequestHeader "Content-Type", "application/x-www-form-urlencoded"

objWinHttp.Send strReq

'gets/scans results
strHTML = objWinHttp.ResponseText
Set objWinHttp = Nothing
%>

<%
'prints results
= strHTML
%>
```

5.2 HTML

5.2.1 Send SMS

The results are delivered as “HTTPS RESPONSE” in the ”text/plain” format directly after myMSP has received the message (and all recipients). In other words, the results are shown in the browser used to send the SMS with information on how the message was handled. In order to process this information, for example save it to a database, some form of written code is necessary.

```
<form action=" https://mymsp.21st.se/external/sendSms" method="post">
<TABLE cellSpacing="0" cellPadding="5" width="300" border="0">
<TR>
<TD vAlign="middle" noWrap align="right">Username</TD>
<TD vAlign="middle" noWrap align="left">
<INPUT type="text" name="user"></TD>
<TD><FONT color="#ff0066">*</FONT></TD>
</TR>
<TR>
<TD vAlign="middle" noWrap align="right">Password</TD>
<TD vAlign="middle" noWrap align="left">
<INPUT type="password" name="pwd"></TD>
<TD><FONT color="#ff0066">*</FONT></TD>
</TR>
<TR>
<TD vAlign="middle" noWrap align="right">Sender</TD>
<TD vAlign="middle" noWrap align="left">
<INPUT type="text" name="originator"></TD>
<TD></TD>
</TR>
<TR>
<TD vAlign="middle" noWrap align="right">Recipient</TD>
<TD vAlign="middle" noWrap align="left">
<INPUT type="text" name="recipients"></TD>
<TD><FONT color="#ff0066">*</FONT></TD>
</TR>
<TR>
<TD vAlign="middle" noWrap align="right">Text</TD>
<TD vAlign="middle" noWrap align="left">
<INPUT type="text" name="msg"></TD>
<TD><FONT color="#ff0066">*</FONT></TD>
</TR>
<TR>
<TD vAlign="middle" noWrap align="left" colSpan="3">
<INPUT type="submit" value="Submit"></TD>
</TR>
</TABLE>
</form>
```

5.2.2 Retrieve SMS

The results are delivered as “HTTPS RESPONSE” in the ”text/plain” format. When sending messages *en masse* there will be a short delay. In other words, the result contains information on the send progress of the messages, which is displayed in text. In order to process this information, for example save it to a database, some form of written code is needed on your part

```
<form action=" https://mymsp.21st.se/external/getSmsResult" method="post">
<TABLE cellSpacing="0" cellPadding="5" width="300" border="0">
<TR>
<TD vAlign="middle" noWrap align="right">Username</TD>
<TD vAlign="middle" noWrap align="left">
<INPUT type="text" name="user"></TD>
<TD><FONT color="#ff0066">*</FONT></TD>
</TR>
<TR>
```

```
<TD vAlign="middle" noWrap align="right">Password</TD>
<TD vAlign="middle" noWrap align="left">
<INPUT type="password" name="pwd"></TD>
<TD><FONT color="#ff0066">*</FONT></TD>
</TR>
<TR>
<TD vAlign="middle" noWrap align="right">Message Identification Number</TD>
<TD vAlign="middle" noWrap align="left">
<INPUT type="text" name="msgId"></TD>
<TD><FONT color="#ff0066">*</FONT></TD>
</TR>
<TR>
<TD vAlign="middle" noWrap align="left" colSpan="3">
<INPUT type="submit" value="Submit"></TD>
</TR>
</TABLE>
</form>
```

myMSP API v2 Appendix Web service

System Interface - Specifications

21st Century Mobile
Marketing & Sales
+46 (0)8 21 21 55
sales@21st.se

www.21st.se
Research & Development
+46 (0)8 21 21 55
tech@21st.se

1 Purpose of Document 3
1.1 Author 3
1.2 Document Status 3
1.2.1 Document Changes 3
1.3 Document Perimeter 3
2 Technical description 5
3 Usage scenarios 6
3.1 Setup communication 6
3.2 Send a message 6
3.3 Check the status of a sent message 7
3.4 Check the status of several sent messages 8
3.5 Receive a message 9
4 Web service methods 10
4.1 Control Connection (ping) 10
4.1.1 Input Parameters (String) 10
4.1.2 Output Parameters (String) 10
4.2 Retrieve parameters (getParams) 10
4.2.1 Input Parameters (BasicRequest) 10
4.2.2 Input class Identification 10
4.2.3 Output Parameters (ParamsResponse) 10
4.3 Send Message (sendMsg) 11
4.3.1 Input Parameters (SendRequest) 11
4.3.2 Additional SMS parameters 11
4.3.3 Input class Recipient 12
4.3.4 [deprecated] Input class MessageContent 12
4.3.5 Output Parameters (SendResponse) 12
4.4 Retrieve Result (getSendResult) 13
4.4.1 Input Parameters (SendResultRequest) 13
4.4.2 Output Parameters (SendResultResponse) 13
4.4.3 Output class RecipientResult 13
4.5 Retrieve Results (getSendResults) 14
4.5.1 Input Parameters (SendResultsRequest) 14
4.5.2 Output Parameters (SendResultsResponse) 14
4.5.3 Output class MsgSendResultInfo 14
4.6 Retrieve Received Messages (getReceivedMessages) 14
4.6.1 Input Parameters (ReceivedMsgRequest) 14
4.6.2 Output Parameters (ReceivedMsgResponse) 15
4.6.3 Output class MessageReceived 15
4.7 [deprecated] Retrieve Message Content (getMsgContent) 15
4.7.1 Input Parameters (MsgContentRequest) 15
4.7.2 Output Parameters (MsgContentResponse) 15
5 Examples 16
5.1 C# 16
5.2 Java 17

1 Purpose of Document

The purpose of this document is to define the web service interface made available by myMSP to access the application functionality. Accessing myMSP can be done from the computer systems of external parties/customers.

1.1 Author

21st Century Mobile
Mikael Rosvall
mikael.rosvall@21st.se
+46 (0)8 21 21 55

1.2 Document Status

- Published 2007-01-08

1.2.1 Document Changes

- B [2007-06-15]:
  - Java code example modified.
  - Minor changes and improvements
- C [2007-07-17]
  - New method added: Retrieve send results
    - 3.4: usage scenario
    - 4.5: method description
  - Added *recommended usage patterns* to some method descriptions in chapter 4.
- D [2007-08-20]
  - Added a note that the maximum length of a sender alias is ten characters.
- E [2007-12-11]
  - Updated the headers and footers with new information.
  - Added a note that the maximum length of a sender alias is eleven characters.
- F [2008-03-28]
  - Updated all URL-references from the old domain (www.mymsp.eu) to the new domain (mymsp.21st.se).
- G [2012-01-24]
  - Added a new section: 4.3.2 Additional SMS parameters.
- H [2012-12-14]
  - Removed parameter “Charset” from section 4.3.1.
- I [2019-03-11]
  - Updated graphic profile.
  - Deprecated MMS methods.

1.3 Document Perimeter

Neither economical questions nor questions regarding agreements/contracts will be dealt with in this document.

The information given in this document may change without notice and describes only the matters defined in the general part of this document. Please verify that your company has the most recent version. This information is intended for the use of customers and partners of 21st Century Mobile. The information or statements given in this document concerning the suitability, capacity or performance of the mentioned

service cannot be considered binding but shall be defined in the agreement concluded between 21st Century Mobile and the customer, if applicable. 21st Century Mobile shall not be responsible in any event for errors in this document or for any damages, incidental or consequential (including monetary losses), that might arise from the use of this publication or the information in it. This material and the service described in this document are copyrighted in accordance with the applicable laws.

2 Technical description

This interface is realized through Web Services. Web Services are software systems that are designed to support machine-to-machine communication over a network, e.g. the Internet. The information exchange is carried out using SOAP-formatted XML-envelopes that are sent through the HTTP-protocol. The web service URL for accessing myMSP is:

- https://mymsp.21st.se/external/ws2/myMSP

WSDL (Web Services Description Language) is used to describe the system interface, i.e. what services are available and how they are accessed. WSDL is constructed to enable an automatic information and service exchange between computers. To hide the complexity of invoking the web services a development tool (e.g. Microsoft Visual Studio, Eclipse, Netbeans) is normally used to automatically generate the required parameter classes as specified by the WSDL file. It is important to note that the address to the WSDL-file should only be used to generate the required parameter classes in the development tool. It should not be used access the web service itself. The WSDL file that describes the myMSP web service can be found at:

- https://mymsp.21st.se/external/ws2/myMSP?wsdl

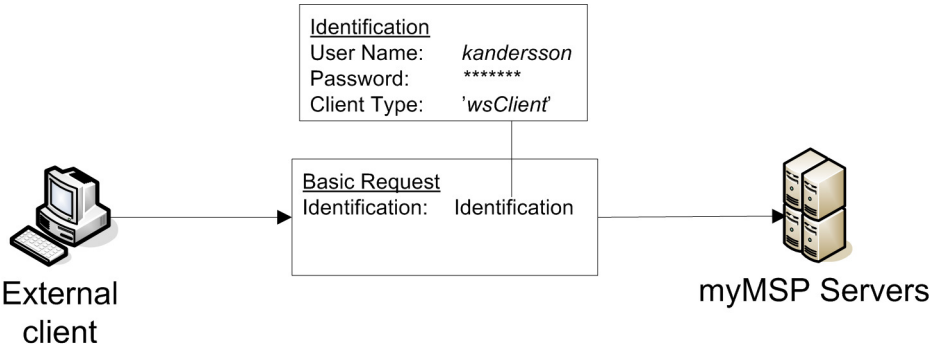
3 Usage scenarios

The following scenarios are intended to give a brief overview of how to use the myMSP web services. In the next chapter the myMSP Web Services are described in more detail.

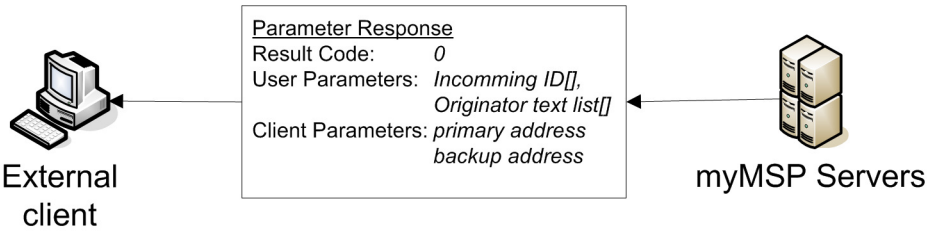
3.1 Setup communication

Before a client that utilizes the myMSP web service can send and receive messages the user identification has to be validated and some configuration information needs to be fetched.

1. In order to initialize communications with myMSP a *basic request*, including user identification and type of client, is used to fetch setup parameters:



2. The *response* includes two sets of parameters: *user-* (e.g. allowed originator texts and incoming ids) and *client-parameters* (e.g. primary address to be used when communicating with the web service and a backup address that can be used if the primary address is inaccessible).

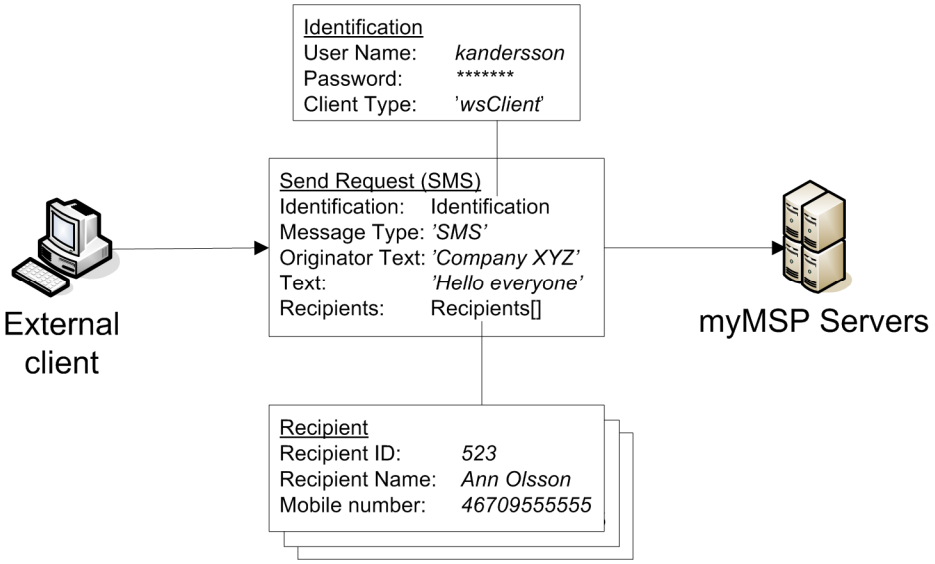


After the client has sent a request to myMSP and has received a response with a *result code* of 0 (indicating that the web service address, user name, and password are all valid) the actual sending and receiving of messages can commence.

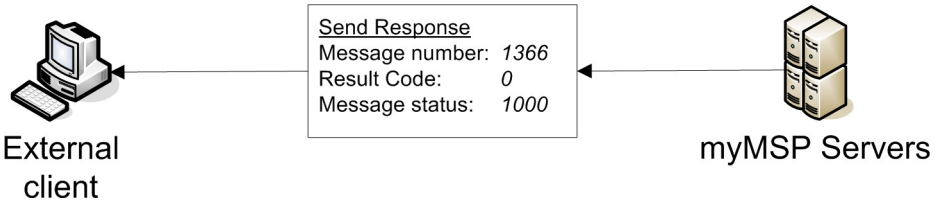
3.2 Send a message

As previously mentioned in the main API-document (3.1 The send message sequence) sending a message through myMSP is done in steps.

1. First, a message *send request* is sent to myMSP. The request contains user identification, a list of *recipients*, and the message content.



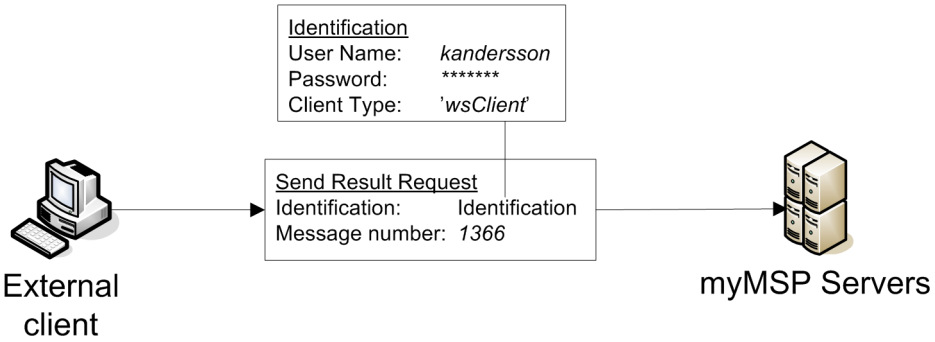
2. After the send request has been sent, a *send response* is returned containing, among other things, the *status* of the message, i.e. how far it has come in the transmission sequence.



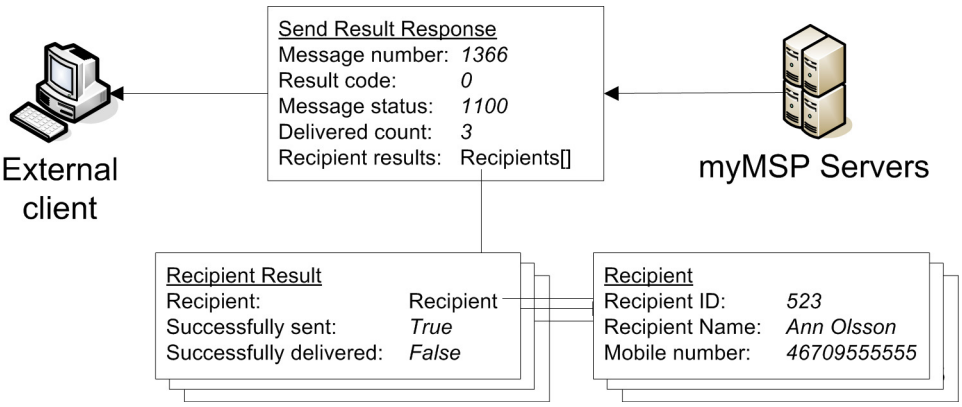
3.3 Check the status of a sent message

It can take some time before the message is delivered to all recipients. The status of a message can be checked while it is being delivered.

1. To check the current status of a message a *send result request* has to be sent to myMSP. The request is composed of user identification, and the message number of the message whose status should be checked.



2. myMSP returns a response containing the status of the message (i.e. how far it has come in the transmission sequence). The result response also contains information regarding the send status for each recipient, e.g. if the message has been successfully sent and delivered to a specific recipient.

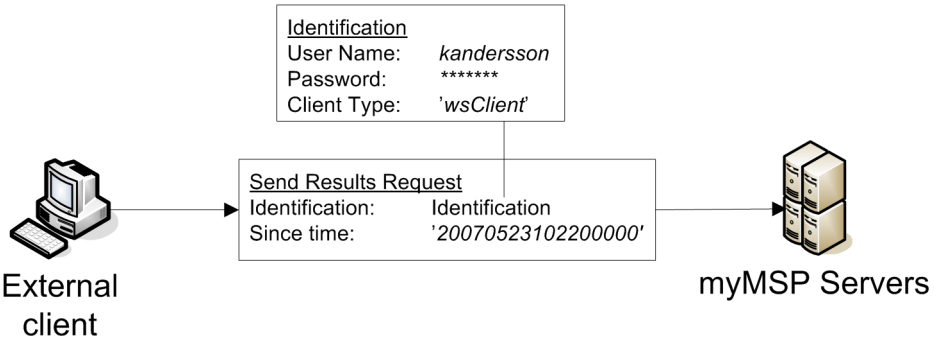


The process checking the send status of a message can be repeated as many times as is desirable, e.g. to check and make sure that all recipients have received the message.

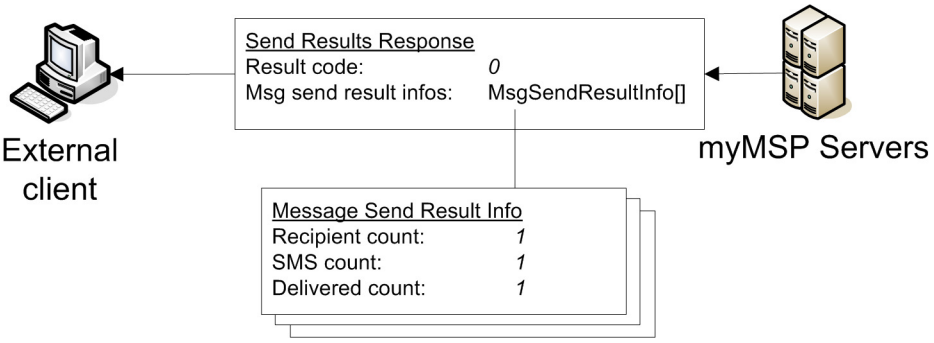
3.4 Check the status of several sent messages

Sometimes it is preferable to check the status of several sent messages at once. Use this method to reduce the number of requests sent to myMSP, e.g. if you regularly send many messages with few recipients per message.

1. To check the current status of several messages a *send results request* has to be sent to myMSP. The request is composed of user identification and a time stamp.



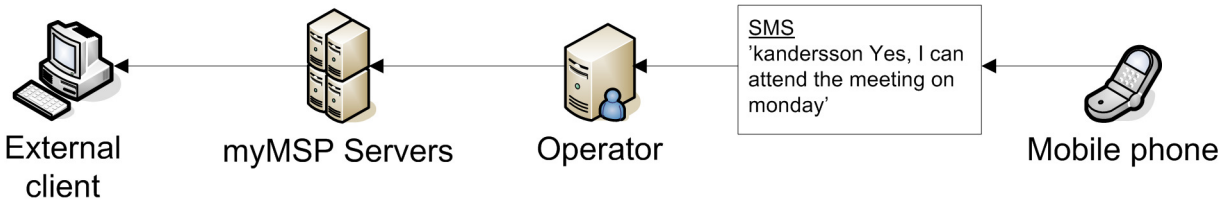
2. myMSP returns a response containing the aggregated send status (e.g. recipient count, sent count, delivered count) of those messages whose status has changed since the time that was specified in the request. However, the response does not contain information regarding the send status of the each recipient of each message. Submit a *send result request* (see 3.3) in order to check the send status of each recipient of a message.



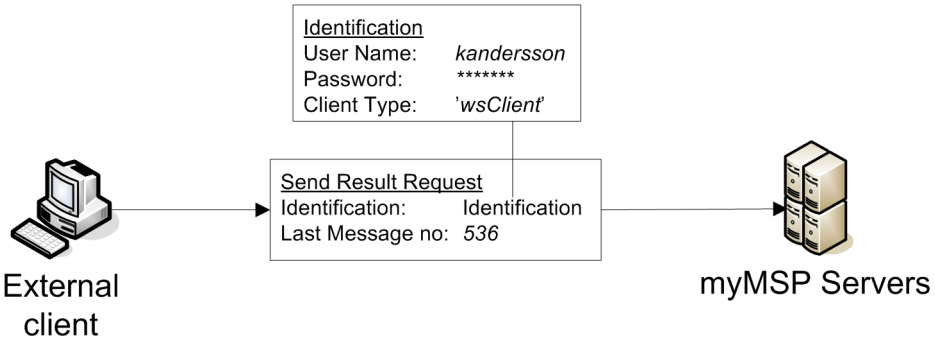
3.5 Receive a message

myMSP can also receive messages from mobile phones.

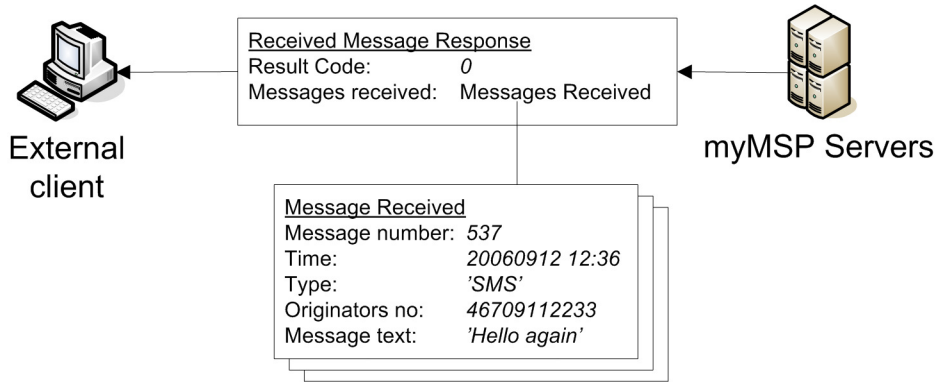
1. An initial ID is needed in order to send a message to a myMSP account from a mobile phone. The initial ID is placed in the beginning of the text in the message and should be followed by a blank space.



2. A *received message request* is needed in order to get new messages that have been sent to a myMSP account.



3. A *received message response* is returned by myMSP containing a set of received messages.



4 Web service methods

This chapter describes the various methods of the myMSP Web service in more detail.

4.1 Control Connection (ping)

Checks if there is a connection to the myMSP-server. The input and output parameters are pretty much dummies.

Recommended usage pattern: ping is useful when developing your application and later when initializing your application in order to check that you connection to myMSP is working properly.

4.1.1 Input Parameters (String)

Who	A text string

4.1.2 Output Parameters (String)

pingReturn	A text string with the value "Alive"

4.2 Retrieve parameters (getParams)

The method makes it possible to customize the remote application based on the current user and client type. The input parameter is a basic request object that contains an instance of the identification class (4.2.2). The identification class is included in all requests to myMSP, accept for the ping.

Recommended usage pattern: Since parameters rarely change they need only be fetched about once a day and when initializing your application.

4.2.1 Input Parameters (BasicRequest)

Identification	A class containing information that identifies the user and the type of client that is used (see 4.2.2 Identification).

4.2.2 Input class Identification

Username	User Account
Pwd	Password
clientType	Type of client (use 'wsClient')
clientVersion	Not applicable

4.2.3 Output Parameters (ParamsResponse)

resultCode	See result codes in main API-document (3.3)!
resultDescription	Text describing the results.
userParams	An array of parameters (Param) for the user. A parameter contains a key and a value.

clientParams	An array of parameters (Param) for the client. A parameter contains a key and a value.
originatorTexts	An array of allowed optional sender aliases (se 'Originator text' in main API-document glossary) to be displayed instead of the originating number.
initialIds	An array of initial ids. An initial id has to be included when a recipient replies to a message sent through myMSP. This enables myMSP to direct the incoming message to the correct user account.

Example of user parameters:

- locale (defines the users country and language 'sv_SE', 'us_EN')
- charset (the users preferred character encoding, 'ISO-8859-1, 'UTF-8')

Example of client parameters:

- primaryAddress (primary address to web service)
- backupAddress (secondary address to web service)
- homepage (Web address to myMSP home page with production status)

4.3 Send Message (sendMsg)

Send a message to recipients. Recipients are sent in the form of an array of recipient objects (4.3.2).

4.3.1 Input Parameters (SendRequest)

Identification	A class containing information that identifies the user and the type of client that is used (see 4.2.2 Identification).
msgType	Type of message ('SMS')
originatorText	Originator text of the message (see 'Originator text' in main API-document glossary). Note that only pre-registered originator texts acquired through the 'getParams' request (userParams in 4.2.3) can be used. The maximum length of an alias is eleven (11) characters.
Recipients	An array of recipients (see 4.3.2 class Recipient below).
smsText	Text Message. Maximum of 765 characters for SMS. See also 4.3.2 on how to add additional SMS parameters to the message using smsText.
subject	[deprecated]
Contents	[deprecated]
externalRef	An external reference for the user of myMSP. Statistics may be available based on this reference.
requestReadReceipt	[deprecated]

4.3.2 Additional SMS parameters

It is possible to set some additional parameters when sending an SMS. This can be done by adding parameter tags at the beginning of the message text ("smsText" in 4.3.1) and placing the actual parameters and their values in between: <param>param1=value1&param2=value2</param>. All parameters and the start and stop tags are removed from the message text once they have been processed and do not add to the total number of characters in the message.

timetolive	How long the SMS should "live", i.e. how long the operator should attempt to send

	the message to a mobile phone before giving up and deleting it. The default value is 72 hours. A timetolive value consists of a number and one of three time units: minutes (m), hours (h) and days (d). The max value is 3 days, and the min value is 5 minutes. Time units cannot be mixed, e.g. a value of one and a half hours should be set in minutes (90m), NOT hours and minutes (1h30m).
deliverytime	The date and time that the message should be delivered. Should be formatted as follows: yyMMddHHmmsszzzz, e.g. 120125183525004+. The last field (zzzz) is the time zone difference in quarters of an hour from Western European Time (UTC+0) i.e. GMT. A value of 000+ or 000- equals GMT, 004+ equals Central European Time (UTC+1), etc.
isflash	Set to "true" if the message should be sent as a flash SMS. A flash SMS is displayed in the recipient's mobile phone as soon as it is delivered, i.e. does not have to be opened by the user. It is also deleted immediately after being read, although some mobile phones allow the user to save the message to their SMS inbox.

Example: to send a flash SMS with the text "This is a test message" that should live for 15 minutes the "smsText" parameter should be set to: "<param>timetolive=15m&isflash=true</param>This is a test message".

4.3.3 Input class Recipient

recipientId	The myMSP internal ID of the recipient
recipientName	The name of the recipient
mobileNumber	The mobile number of the recipient
externalRef	An external reference for the recipient in myMSP. If the user of myMSP services uses another way of identifying a recipient, other from recipientName or mobileNumber, it can be specified here.

4.3.4 [deprecated] Input class MessageContent

slideNo	[deprecated]
duration	[deprecated]
contentType	[deprecated]
contentCharset	[deprecated]
contentBytes	[deprecated]

4.3.5 Output Parameters (SendResponse)

resultCode	All results other then zero means that there has been an error. See result codes in main API-document (3.3)!
resultDescription	Text describing the results.
msgNo	Message identification number in myMSP.
msgStatus	Message Status (state) in myMSP. See message status in main API-document (3.2)!
createTime	The time (yyyyMMddHHmmssSSS) the message was created

4.4 Retrieve Result (getSendResult)

Check the send result of a message. The individual delivery status for each recipient is included in an array of recipient result-objects (4.4.3).

Recommended usage pattern: Typically, the send result of a message changes frequently at first (1-2 min) and later less frequently. It is therefore recommended that the send status of a message be check less frequently the longer it has been since the message was sent.

4.4.1 Input Parameters (SendResultRequest)

Identification	A class containing information that identifies the user and the type of client that is used (see 4.2.2 Identification).
msgNo	Message identification number in myMSP.

4.4.2 Output Parameters (SendResultResponse)

resultCode	See result codes in main API-document (3.3)!
resultDescription	Text describing the results.
msgStatus	See message status in main API-document (3.2)!
sendRequestTime	The time (yyyyMMddHHmmssSSS) the message was requested to be sent.
recipientCount	The number of recipients of the message.
sentOkCount	The number of messages that where successfully sent to the operator.
deliveredOkCount	The number of successful deliveries to recipients.
readOkCount	[deprecated]
recipientResults	A result per recipient represented by an array of the class RecipientResult described below.

4.4.3 Output class RecipientResult

recipient	A class containing the recipient information (see 4.3.2 class Recipient).
operatorResultCode	The result code (error code) from the operator. Provided as guidance for an eventual correction of the message.
operatorResultDescription	The result description (error text) from the operator. Provided as guidance for an eventual correction of the message.
sentOk	Indicates if the operator has accepted the message and the recipient.
sentTime	The time (yyyyMMddHHmmssSSS) the message was accepted by the operator.
deliveredOk	Indicates if the operator has successfully delivered the message to the recipient.
deliveredTime	The time (yyyy-MM-dd HH:mm) the message was delivered to the recipient (mobile phone).
readOk	[deprecated]
readTime	[deprecated]

4.5 Retrieve Results (getSendResults)

Check the send result of several messages.

Recommended usage pattern: If your application normally sends many messages with a few recipients each (e.g. 100 messages a day with one recipient per message) then it is usually better to send a *sendResultsRequest* (that fetches a batch of send results) every few minutes instead of sending a *sendResultRequest* for each message.

4.5.1 Input Parameters (SendResultsRequest)

Identification	A class containing information that identifies the user and the type of client that is used (see 4.2.2 Identification).
sinceTime	A time stamp, format: yyyyMMddHHmmssSSS. All send status changes that have occurred after this time will be returned.

4.5.2 Output Parameters (SendResultsResponse)

resultCode	See result codes in main API-document (3.3)!
resultDescription	Text describing the results.
msgSendResultInfos	An array of the class MsgSendResultInfo described below.

4.5.3 Output class MsgSendResultInfo

msgNo	Message identification number in myMSP.
sendRequestTime	The time (yyyyMMddHHmmssSSS) the message was requested to be sent.
recipientCount	The number of recipients of the message.
sentOkCount	The number of messages that where successfully sent to the operator.
smsCount	myMSP splits long SMS (>160 characters) into several SMS which are sent separately and later reassembled in the recipient's mobile phone. smsCount specifies the total number of SMS that are sent. For example: an SMS with a 170 characters long text and two recipients will generate a smsCount of four (2 recipients x 2 text segments). The smsCount for normal SMS (=<160 characters) is always the same as the recipient count.
deliveredOkCount	The number of successful deliveries to recipients.
readOkCount	[deprecated]

4.6 Retrieve Received Messages (getReceivedMessages)

The method fetches received messages.

4.6.1 Input Parameters (ReceivedMsgRequest)

Identification	A class containing information that identifies the user and the type of client that is used (see 4.2.2 Identification).
lastMsgNo	The message identification number of the message last received, input 0 to retrieve all.

4.6.2 Output Parameters (ReceivedMsgResponse)

resultCode	See result codes in main API-document (3.3)!
resultDescription	Text describing the results.
messagesReceived	An array (MessageReceived []) that contains received messages. The class "MessageReceived" is described below.

4.6.3 Output class MessageReceived

msgNo	Message identification number in myMSP.
createTime	Time and date when the message arrived in myMSP.
creatorName	Name of the myMSP user that received the message.
initialId	The initial Id used to identify the recipient of the incoming message.
msgType	Type of message ('SMS')
originator	The originators mobile number
originatorText	Originator text of the message. See glossary above.
smsText	The SMS text
subject	[deprecated]

4.7 [deprecated] Retrieve Message Content (getMsgContent)

Deprecated method for getting the content of a MMS message.

4.7.1 Input Parameters (MsgContentRequest)

Identification	A class containing information that identifies the user and the type of client that is used (see 4.2.2 Identification).
msgNo	Message identification number in myMSP

4.7.2 Output Parameters (MsgContentResponse)

resultCode	See result codes in main API-document (3.3)!
resultDescription	Text describing the results.
contents	[deprecated]

5 Examples

The following code examples illustrate how to use the parameter classes in a development environment. More complete code samples are also available for Visual Studio 2005 and for Java development environments. The samples are meant to illustrate the how to use the functionality of the Web Service, i.e. creating requests, sending requests, receiving responses, reading the data in the responses, and displaying the results. You are free to view, modify, copy and use the code in any way that you like. The code samples can be downloaded from our file library. Go to https://mymsp.21st.se/app and log in to access the library.

Note that the myMSP support classes that are used in the code examples have been automatically generated by a development environment using the myMSP WSDL-file (see chapter 2). These classes look a bit different depending on which development environment is used.

5.1 C#

```
private void sendSMS()
{
	// The web service connection-object
	eu.mymsp.www.MyMSP2 _webService = new eu.mymsp.www.MyMSP2();

	// Create and fill a identification object
	eu.mymsp.www.Identification id = new eu.mymsp.www.Identification();
	id.username = "johnDoe";
	id.pwd = "xyz123";
	id.clientType = "wsClient";
	id.clientVersion = "";

	// Create a recipient array
	eu.mymsp.www.Recipient[] recipients = new eu.mymsp.www.Recipient[1];

	// Create and fill a recipient object
	eu.mymsp.www.Recipient recipient = new eu.mymsp.www.Recipient();
	recipient.mobileNumber = "46709888888";
	recipient.recipientName = "Jane Doe";
	recipient.externalRef = "";

	// Add the recipient to the recipient array
	recipients[0] = recipient;

	// Create a send request object
	eu.mymsp.www.SendRequest sendRequest = new eu.mymsp.www.SendRequest();

	// Fill the send request
	sendRequest.identification = id;
	sendRequest.recipients = recipients;
	sendRequest.msgType = "SMS";
	sendRequest.smsText = "Hello, this is a test message.";
	sendRequest.requestReadReceipt = true;
	sendRequest.subject = "";
	sendRequest.originatorText = "";
	sendRequest.externalRef = "";
```

```
        sendRequest.contents = null;

        // Create a sendMsg object that will contain all the data that should
        // be sent to the web service
        eu.mymsp.www.sendMsg sendSms = new eu.mymsp.www.sendMsg();

        // Add the send request to the sendMsg-object
        sendSms.SendRequest_1 = sendRequest;

        // Create a response object that stores the result of the send-attempt
        eu.mymsp.www.sendMsgResponse response = null;

        try
        {
            // Send the message
            response = _webService.sendMsg(sendSms);
        }
        catch
        {
            response = null;
        }
}
```

5.2 Java

The following sample code was created using NetBeans 5.5 and depends on support classes generated by the Web Service framework JAX-WS (a NetBeans wizard). The support classes might look different if you use another development environment (e.g. Eclipse) and/or Web Service framework (e.g. JAX-RPC, Axis) to generate them. For more information we refer you to the support documents of your development environment that details how to create a Web Service client.

```
private void sendSMS() throws Exception {
        // The web service connection-object
        eu.mymsp.www.MyMSPInterface1 webServicePort =
            new eu.mymsp.www.MyMSP2().getMyMSPInterface1Port();

        // Create and fill a identification object
        eu.mymsp.www.Identification id = new eu.mymsp.www.Identification();
        id.setUsername("johnDoe");
        id.setPwd("xyz123");
        id.setClientType("wsClient");
        id.setClientVersion("");

        // Create and fill a recipient object
        eu.mymsp.www.Recipient recipient = new eu.mymsp.www.Recipient();
        recipient.setMobileNumber("46709888888");
        recipient.setRecipientName("Jane Doe");
        recipient.setExternalRef("");

        // Create a send request object
        eu.mymsp.www.SendRequest sendRequest = new eu.mymsp.www.SendRequest();

        // Fill the send request
        sendRequest.setIdentification(id);
```

```
        sendRequest.getRecipients().add(recipient);
        sendRequest.setMsgType("SMS");
        sendRequest.setSmsText("Hello, this is a test message.");
        sendRequest.setRequestReadReceipt(true);
        sendRequest.setSubject("");
        sendRequest.setOriginatorText("");
        sendRequest.setExternalRef("");

        // Create a response object that stores the result of the send-attempt
        eu.mymsp.www.SendResponse response = null;

        try {
            // Send the message
            response = webServicePort.sendMsg(sendRequest);
        } catch (Exception ex) {
            ex.printStackTrace();
            throw ex;
        }

        System.out.println(response.getMsgNo());
        System.out.println(response.getMsgStatus());
        System.out.println(response.getResultCode());
        System.out.println(response.getResultDescription());
        System.out.println(response.getCreateTime());
}
```

myMSP API v2 Appendix SMPP

System Interface - Specifications

21st Century Mobile
Marketing & Sales
+46 (0)8 21 21 55
sales@21st.se

www.21st.se
Research & Development
+46 (0)8 21 21 55
tech@21st.se

1 Purpose of Document 3
1.1 Author 3
1.2 Document Status 3
1.2.1 Document Changes 3
1.3 Document Perimeter 3
2 SMPP Specifications 4
2.1 Account details 4
2.2 Activate delivery 4
2.3 Supported SMPP functions 4
2.4 SMPP TON/NPI Parameters 4
2.5 Data coding 4
2.6 Enquire link 5
2.7 User Data Header (UDH) 5
2.7.1 Concatenated message 5
2.7.2 Destination and source port 5
3 SMPP functions 6
3.1 BIND_TRANSMITTER 6
3.2 BIND_RECEIVER 6
3.3 BIND_TRANSCEIVER 6
3.4 ENQUIRE_LINK 6
3.5 UNBIND 6
3.6 GENERIC_NACK 6
3.7 SUBMIT_SM 7
3.8 DELIVER_SM 7
3.8.1 Delivery report format 8

1 Purpose of Document

The purpose of this document is to define the SMPP interface made available by myMSP to access the application functionality. Accessing myMSP can be done from the computer systems of external parties/customers.

1.1 Author

21st Century Mobile
Mikael Rosvall
mikael.rosvall@21st.se
+46 (0)8 21 21 55

1.2 Document Status

- Published 2009-12-21

1.2.1 Document Changes

- B [2019-03-11]
  - Updated graphic profile.

1.3 Document Perimeter

Neither economical questions nor questions regarding agreements/contracts will be dealt with in this document.

The information given in this document may change without notice and describes only the matters defined in the general part of this document. Please verify that your company has the most recent version. This information is intended for the use of customers and partners of 21st Century Mobile. The information or statements given in this document concerning the suitability, capacity or performance of the mentioned service cannot be considered binding but shall be defined in the agreement concluded between 21st Century Mobile and the customer, if applicable. 21st Century Mobile shall not be responsible in any event for errors in this document or for any damages, incidental or consequential (including monetary losses), that might arise from the use of this publication or the information in it. This material and the service described in this document are copyrighted in accordance with the applicable laws.

2 SMPP Specifications

This interface is realised through the Short Message Peer-to-Peer (SMPP) protocol. SMPP is the most commonly used protocol for sending SMS messages in use today.

2.1 Account details

host	smpp1.mymsp.21st.se
port	4300
system_id	<myMSP user name>
password	<myMSP password>
message mode	Store and Forward

2.2 Activate delivery

myMSP does not automatically send delivery reports or MO message to a client application that binds a receiver or a transceiver. If the client wants to be able to receive messages then this needs to be specified when setting up the account with 21st Century Mobile.

2.3 Supported SMPP functions

BIND_TRANSMITTER /_RESP
BIND_RECEIVER /_RESP
BIND_TRANCEIVER /_RESP
ENQUIRE_LINK /_RESP
UNBIND /_RESP
SUBMIT_SM /_RE
DELIVER_SM /_RESP
GENERIC_NACK /_RESP

2.4 SMPP TON/NPI Parameters

The *Type Of Number* (TON) and *Number Plan Indicator* (NPI) values do not have to be specified since myMSP sets them automatically.

SMPP parameter	Type of address	Max length	Valid input	Output example
Source address	International	20	0-9, +, -,	+46709111111
Source address	Alphanumeric	11	a-z, A-Z, 0-9 (guaranteed)	myMSP
Destination address	Always international	20	0-9, +, -	+46709111111

2.5 Data coding

The following character encoding regimes are supported.

0x00	GSM 03.38 (default GSM encoding)
0x01	ASCII (IA5)
0x03	Latin-1 (ISO-8859-1)

2.6 Enquire link

The client should send an ENQUIRE_LINK request every 60 seconds to keep the link alive.

2.7 User Data Header (UDH)

In order to use some of the more advanced features of SMPP a *User Data Header* (UDH) needs to be included in a message. The UDH should be placed before the message text in the *short_message* parameter of a *SUBMIT_SM* request. A UDH can also be included in a *mobile originated* (MO) message that is sent to the client from myMSP through a *DELIVER_SM* request. The *ems_class* of a message that includes an UDH should always be set to 64.

An UDH consists of the following fields:

- Length of User Data Header.
- Information Element Identifier (IEI). Used to identify what type of UDH is being submitted.
- Length of the header, excluding the first two fields.
- Content

2.7.1 Concatenated message

The IEI of a concatenated message is 0. The content must contain three octets: a reference number, the total number of parts, and the current message's number in the sequence.

Example:
Part 1: 0x5, 0x0, 0x3, 0x1, 0x2 0x1
Part 1: 0x5, 0x0, 0x3, 0x1, 0x2 0x2

2.7.2 Destination and source port

The IEI of a message which specifies a destination and a source port is either 4 (for 8 bit addresses) or 5 (for 16 bit addresses). The content must contain two (8 bit) or four octets (16 bit): a destination port and a source port.

Example:
8-bit [destination=50, source=0]: 0x4, 0x4, 0x2, 0x32, 0x0
16-bit [destination=50000, source=0]: 0x6, 0x5, 0x4, 0xc3, 0x50, 0x0, 0x0

3 SMPP functions

The following section specifies which parameters should be set for each SMPP request sent to myMSP, and which parameters myMSP sets when sending requests to the client. Parameters that are not listed below will be ignored if they are set.

3.1 BIND_TRANSMITTER

Parameter	Value
system_id	<myMSP user name>
password	<myMSP password>

3.2 BIND_RECEIVER

Parameter	Value
system_id	<myMSP user name>
password	<myMSP password>

3.3 BIND_TRANSCEIVER

Parameter	Value
system_id	<myMSP user name>
password	<myMSP password>

3.4 ENQUIRE_LINK

According to the SMPP 3.4 specification.

3.5 UNBIND

According to the SMPP 3.4 specification.

3.6 GENERIC_NACK

According to the SMPP 3.4 specification.

3.7 SUBMIT_SM

Parameter	Value
destination_addr	The mobile number that the message should be sent to.
data_coding	Indicates which character encoding regiment was used to encode the message text (see 2.5). This value should be set to 240 when sending flash SMS.
esm_class	The ESM class of the message. Should be set to 64 if the message has a UDH.
schedule_delivery_time	When the message should be delivered. Set to NULL for immediate delivery. Should be set as an absolute time.
short_message	The message body. Contains the message text and/or a UDH (see 2.7). The data coding value is used do encode the text.
source_addr	The originator (numeric or alphanumeric) that should be set for the message.
validity_period	The validity period of the message. Set to NULL to request the default validity period.

3.8 DELIVER_SM

Delivery reports and MO messages are sent from myMSP to the client as a DELIVER_SM requests.

Delivery report	
Parameter	**Value**
destination_addr	The mobile number that the message was sent to.
dest_addr_ton	The destination TON that was set for the message.
dest_addr_npi	The destination NPI that was set for the message.
source_addr	The originator (numeric or alphanumeric) of the sent message.
source_addr_ton	The source TON that was set for the message.
source_addr_npi	The source NPI that was set for the message.
esm_class	Always set to 4.
short_message	The message body. Contains the information content of the delivery report (see 3.8.1).

Mobile originated message	
Parameter	**Value**
destination_addr	The mobile number of the handset that sent the message.
source_addr	The number that the message was sent to.
esm_class	Always set to 2.
short_message	The message body. Contains the message text.

3.8.1 Delivery report format

The information content of the delivery report is contained in the short_message parameter. The following parameters are always present:

id	The ID that was assigned to the message when it was submitted to myMSP.
dlvrd	Always set to "1".
submit date	The date and time when the message was first submitted to myMSP. Format: yyMMddHHmm.
done date	The date and time when the message was delivered to the recipient. Format: yyMMddHHmm.
stat	The delivery status of the message. myMSP can format the delivery status in two ways: default and short. Short delivery statuses are always 8 characters long, while default statuses vary in length. Please specify which version is preferable when setting up the account with 21st Century Mobile. stat can have the following values: **Default** **Short** DELIVERED DELIVRD RETRIEVED DELIVRD EXPIRED EXPIRED REJECTED REJECTD DELETED DELETED UNDELIVERED UNDELIV FAILED UNDELIV ERROR UNDELIV
err	Operator error code.
text	Always empty.

Example:
"id:123456789 dlvrd:1 submit date:0901011201 done date: 0901011205 stat:DELIVERED err:0 Text:"

myMSP API v2 Appendix HTTP Push

System Interface – Specifications

21st Century Mobile
Marketing & Sales
+46 (0)8 21 21 55
sales@21st.se

www.21st.se
Research & Development
+46 (0)8 21 21 55
tech@21st.se

1 Purpose of Document 3
1.1 Author 3
1.2 Document Status 3
1.2.1 Document changes 3
1.3 Document Perimeter 3
2 Technical description 4
3 Usage scenarios 5
3.1 Receive delivery info 5
3.2 Receive delivery report 5
3.3 Receive an incoming message 6
3.4 Ping 7
4 Transmission sequence scenario 8
5 HTTP push methods 9
5.1 Ping 9
5.2 Delivery info 9
5.3 Delivery report 10
5.4 Incoming message 10

1 Purpose of Document

The purpose of this document is to define the HTTP push interface made available by myMSP to send data to the computer systems of external parties/customers.

1.1 Author

21st Century Mobile AB
Mikael Rosvall
mikael.rosvall@21st.se
+46 (0)8 21 21 55

1.2 Document Status

- Published 2009-12-08

1.2.1 Document changes

- B [2012-02-23]
  - New parameter added: destination (5.4)
  - New parameter added: externalRef (5.4)
  - New parameter added: isPremium (5.4)
- C [2019-03-11]
  - Updated graphic profile.

1.3 Document Perimeter

Neither economical questions nor questions regarding agreements/contracts will be dealt with in this document.

The information given in this document may change without notice and describes only the matters defined in the general part of this document. Please verify that your company has the most recent version. This information is intended for the use of customers and partners of 21st Century Mobile. The information or statements given in this document concerning the suitability, capacity or performance of the mentioned service cannot be considered binding but shall be defined in the agreement concluded between 21st Century Mobile and the customer, if applicable. 21st Century Mobile shall not be responsible in any event for errors in this document or for any damages, incidental or consequential (including monetary losses), that might arise from the use of this publication or the information in it. This material and the service described in this document are copyrighted in accordance with the applicable laws.

2 Technical description

myMSP can transmit to an external system (client) through HTTP or HTTPS using the POST method to submit the data. The push interface is used to send data, like incoming messages and delivery status updates, directly to a client thus eliminating the need for that system to constantly poll myMSP for updates. The client system must set up a HTTP listener that can process the request and return a response. The listener must be accessible through an URL, for example:

- http://www.mysite.com/listener

It is possible to specify optional parameters that should be included in every request send from myMSP, e.g. a password to increase security:

- http://www.mysite.com/listener?pwd=123456

The myMSP HTTP push interface is designed to be compatible with the Web service interface, i.e. it directly maps the Web service classes to request parameters. This relationship is more thoroughly discussed in the next chapter. However, it is also possible to use the push interface in conjunction with both the HTTPS- and XML-API:s, but not with the SMPP interface.

To activate HTTP-push send an e-mail to support@21st.se and include the following information:

- The **user name** of the account that the push functionality should be activated for.
- The **target URL** to which the push messages should be sent.
- Any **optional parameters** that should be included in every request. Not required.

3 Usage scenarios

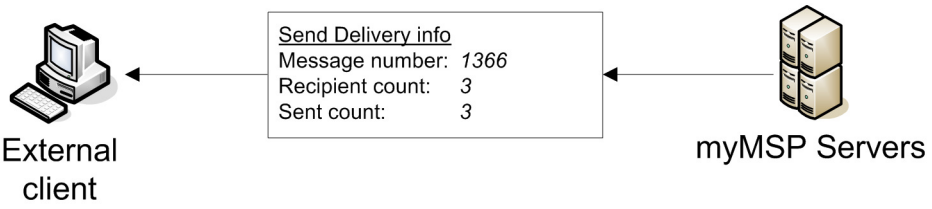
The following scenarios are intended to give a brief overview of how to use the myMSP push interface.

Note that all messages sent from myMSP require the client application to **always** respond with HTTP response code 200 if the message has been correctly received. If myMSP does not receive a response code with the value 200 then the message will be sent again. The client application should acknowledge the message as soon as it arrives as the connection has a read timeout of 10 seconds. Any processing of the message, like updating the database, should be done after the acknowledgment.

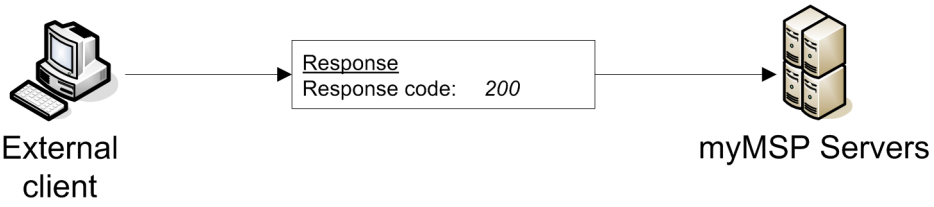
3.1 Receive delivery info

It can take some time for a message to be sent and delivered to all recipients. When myMSP has finished sending a message a *delivery info-message* is sent to the external client. A delivery info-message contains the same parameters as the Web service class *SendResultResponse* (see 4.4.2 in the Web service appendix document), with the exception of *msgStatus*, *recipientResults*, *deliveredOkCount* and *readOkCount* which are not included since they are redundant. A new parameter, *smsCount*, is added.

1. myMSP sends a delivery status report containing the message number, the send status which will be either 1100 (completed) or 1200 (failed), and some aggregated information regarding recipients.



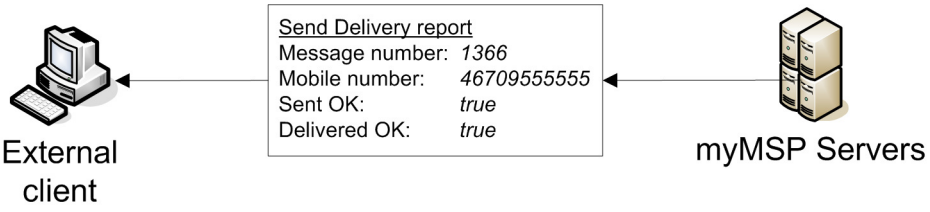
2. The client application responds with HTTP response code 200.



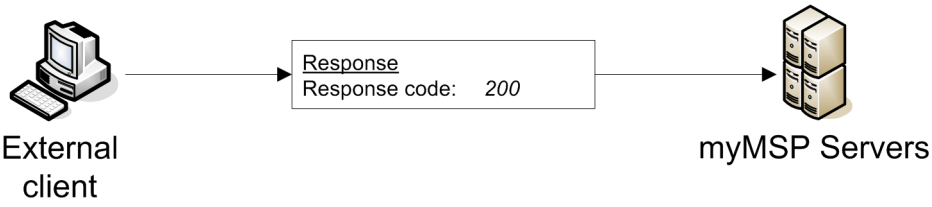
3.2 Receive delivery report

After the delivery info-message has been successfully sent to the client myMSP will begin sending *delivery reports* for each recipient of the message. A delivery report will be sent each time the delivery status has been updated, e.g. if the message has been delivered to a recipient or if it has expired. A delivery report corresponds with the *RecipientResult* and *Recipient* classes in the Web service interface (see 4.3.2 and 4.4.3. in the Web service appendix document).

1. myMSP sends a delivery report containing a message number, recipient information, and the delivery status of the message for the current recipient.



2. The client application responds with HTTP response code 200. The message number is used to link the delivery report with a previously sent message.

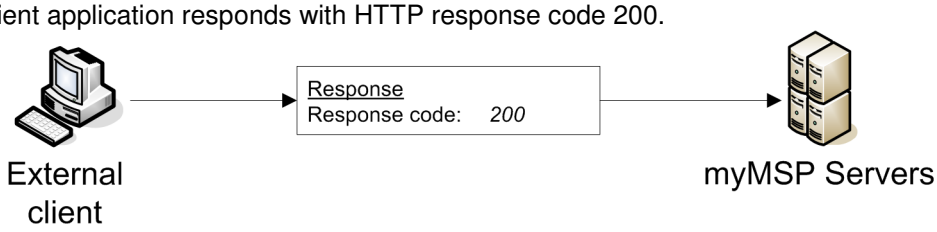


3.3 Receive an incoming message

myMSP can receive SMS sent from mobile phones and send these messages to a client application. An *incoming message* corresponds with the *MessageReceived* class of the Web Service interface (see 4.6.3 in the Web service appendix document).

1. myMSP sends an incoming message containing the message number, In-ID, text etc. to the client.

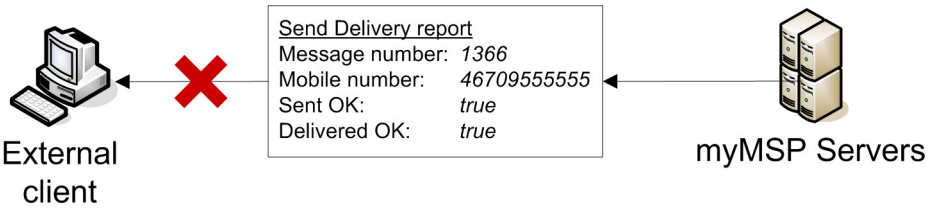
2. The client application responds with HTTP response code 200.



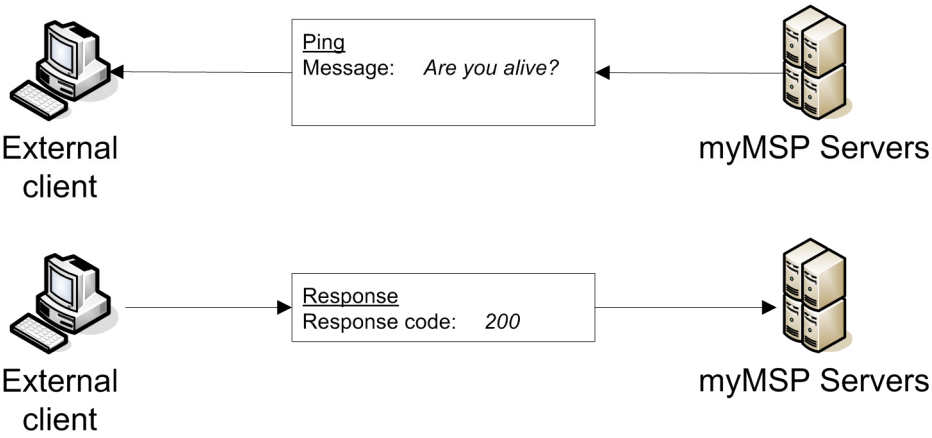
3.4 Ping

If myMSP does not receive any responses after sending several messages to a client application, that client's push queue will be stopped. myMSP will start sending *ping messages* to the client until it receives a positive response.

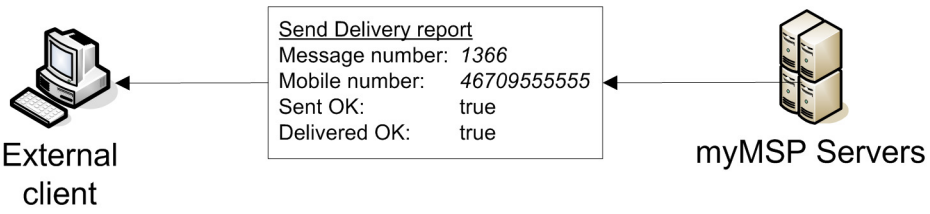
1. myMSP puts a client's push queue on hold after 10 failed attempts to send messages.



2. myMSP sends a ping message every 20 seconds until it receives a positive response from the client.



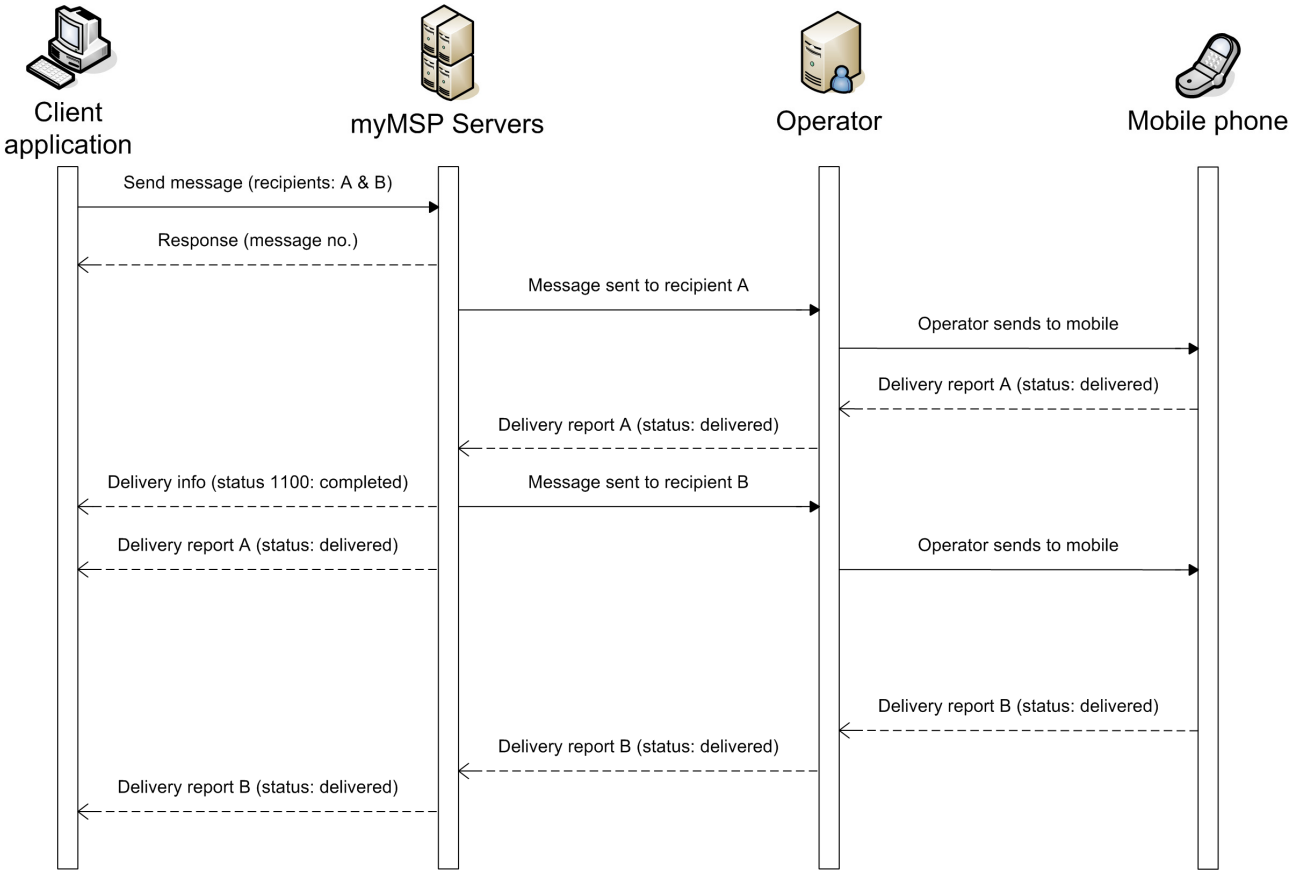
3. The push queue is re-started.



4 Transmission sequence scenario

The figure below illustrates a typical transmission sequence where a client application uses an API (e.g. HTTPS or Web service) to send a message to myMSP and receive the send status through the push interface.

A client application connects to myMSP through an API and transmits a *send message*-request containing a message and a list of two recipients. myMSP receives the message and returns a *response* that includes the message number that the new message has been given. myMSP sends the message to the recipients through their respective mobile phone-operators, beginning with recipient A. The operator sends the message to recipient A's phone and a *delivery report* (A) is returned to myMSP. myMSP does not send the delivery report to the client application yet since the message has not yet been sent to all recipients. When the message has been sent to recipient B it is considered finished (message status: 1100) and a *delivery info-message* is sent to the client. The delivery info-message also specifies that the number of successfully sent messages is two, i.e. that it has been successfully sent to all recipients. The client uses the message no. in the delivery info-message to match the report to a previously sent message. Delivery report A is immediately sent by myMSP once the delivery info-message is accepted by the client. A little while later recipient B switches on his phone and delivery report B is sent from the operator to myMSP and finally to the client application.



5 HTTP push methods

This chapter describes the various messages that the myMSP push interface can send in more detail.

The parameter *messageType* is included in every message and indicates which type of message is being sent. It can have the following values:

0	Ping
1	Deliver info
2	Delivery report
3	Incoming message

If any optional parameters have been specified then they are also included in every message.

5.1 Ping

A ping is sent to check if the client application is responding.

messageType	0
pingMessage	Are you alive?

Example:
http://www.mysite.com/listener?messageType=0&pingMessage=Are%20you%20alive%3F

5.2 Delivery info

When a message has been sent to all its' recipients a delivery info-message is sent.

messageType	1
msgNo	Message identification number in myMSP.
sendRequestTime	The time (yyyyMMddHHmmssSSS) the message was requested to be sent.
recipientCount	The number of recipients of the message.
smsCount	If the message text of a SMS is longer then 160 characters then the message is split in to several messages that are linked together. The SMS count indicates how many messages in total where sent (2 recipients + 2 parts = SMS count 4). For normal SMS the SMS count and the recipient count will always be identical.
sentOkCount	The number of messages that where successfully sent to the operator. If all messages have been sent then the sent ok count and the SMS count will have the same value.

Example:
http://www.mysite.com/listener?messageType=1&msgNo=1234567&sendRequestTime=20090101153422000&recipientCount=2&smsCount=4&sentOkCount=4

5.3 Delivery report

After the delivery info-message has been successfully sent to the client myMSP will begin sending delivery reports for each recipient of the message.

messageType	2
msgNo	Message identification number in myMSP.
recipientId	The myMSP internal ID of the recipient.
recipientName	The name of the recipient.
mobileNumber	The mobile number of the recipient.
externalRef	An external reference for the recipient in myMSP.
operatorResultCode	The result code (error code) from the operator. Provided as guidance for an eventual correction of the message.
operatorResultDescription	The result description (error text) from the operator. Provided as guidance for an eventual correction of the message.
sentOk	Indicates if the operator has accepted the message and the recipient.
sentTime	The time (yyyyMMddHHmmssSSS) the message was accepted by the operator.
deliveredOk	Indicates if the operator has successfully delivered the message to the recipient.
deliveredTime	The time (yyyyMMddHHmmssSSS) the message was delivered to the recipient (mobile phone).
readOk	Indicates if the message is opened (read) by the recipient. Not always available, is dependent on operator (not all operators support the feature) and message type.
readTime	The time (yyyyMMddHHmmssSSS) the message was opened (read) by the recipient. Not always available, is dependent on operator (not all operators support the feature) and message type.

Example:
http://www.mysite.com/listener?messageType=2&msgNo=1234567&recipientId=99991&recipientName=John%20Doe&mobileNumber=46709111111&externalRef=1000&operatorResultCode=0&operatorResultDescription=&sentOk=true&sentTime=20090101153425000&deliveredOk=true&deliveredTime=20090101153455000&readOk=false&readTime=

5.4 Incoming message

When myMSP receives a message sent from mobile phone it forwards the information to the client application.

messageType	3
msgNo	Message identification number in myMSP.
createTime	Time and date when the message arrived in myMSP.
creatorName	Name of the myMSP user that received the message.
initialId	The initial Id used to identify the recipient of the incoming message.

msgType	Type of message (1=SMS, 2=DPN, 3=VOI).
originator	The originator's mobile number.
originatorText	Originator text of the message.
destination	The phone number that the message was sent to.
smsText	The SMS text.
subject	[deprecated]
externalRef	An external reference for the message.
isPremium	True if the message is a premium message.

Example:
http://www.mysite.com/listener?messageType=3&msgNo=1234568&createTime=2009%2D01%2D01%2010%3A56&creatorName=aUsername&initialId=anInID&msgType=1&originator=46709111111&originatorText=&smsText=A%20test%20message&subject=&externalRef=&isPremium=false

myMSP API Appendix
Device Push Notifications

System Interface – Specifications

21st Century Mobile
Marketing & Sales
+46 (0)8 21 21 55
sales@21st.se

www.21st.se
Research & Development
+46 (0)8 21 21 55
tech@21st.se

1 Purpose of Document 3
1.1 Author 3
1.2 Document Status 3
1.2.1 Document changes 3
1.3 Document Perimeter 3
2 Technical description 4
3 Usage scenarios 5
3.1 Register app with notification service provider 5
3.2 Add subscription to myMSP 5
3.3 Update notification key 7
3.4 Remove subscription from myMSP 7
3.5 Remove all subscriptions from myMSP 8
3.6 Send notification 9
3.7 Get notification content 10
4 DPN methods for apps 12
4.1 Add subscription (subscribe) 12
4.1.1 Input parameters 12
4.1.2 Output parameters (positive result) 12
4.1.3 Output parameters (negative result) 12
4.2 Update notification service key (updatekey) 14
4.2.1 Input parameters 14
4.2.2 Output parameters (positive result) 14
4.2.3 Output parameters (negative result) 14
4.3 Remove subscription (unsubscribe) 15
4.3.1 Input parameters 15
4.3.2 Output parameters (positive result) 15
4.3.3 Output parameters (negative result) 15
4.4 Remove all subscriptions (unsubscribeall) 16
4.4.1 Input parameters 16
4.4.2 Output parameters (positive result) 16
4.4.3 Output parameters (negative result) 16
4.5 Get notification content (get) 17
4.5.1 Input parameters 17
4.5.2 Output parameters (positive result) 17
4.5.3 Output parameters (negative result) 17
5 DPN methods for external server 19
5.1 Send notification (send) 19
5.1.1 Input parameters 19
5.1.2 Output parameters (positive result) 19
5.1.3 Output parameters (negative result) 20
5.2 Get notification events through HTTP push 21
5.3 Get notification events through HTTPS retrieve received messages 22
5.3.1 Input parameters 22
5.3.2 Output parameters (positive result) 22
5.3.3 Output parameters (negative result) 22
5.4 Get notification events through WebService retrieve received messages 24
5.4.1 Input Parameters (ReceivedMsgRequest) 24
5.4.2 Output Parameters (ReceivedMsgResponse) 24
5.4.3 Output class MessageReceived 24

1 Purpose of Document

The purpose of this document is to describe and define the API interface made available by myMSP to access the Device Push Notification (DPN) functionality.

1.1 Author

21st Century Mobile AB
Mikael Rosvall
mikael.rosvall@21st.se
+46 (0)8 21 21 55

1.2 Document Status

- Published 2012-10-24

1.2.1 Document changes

- B [2015-01-26]
  - Replaced C2DM with GCM
- C [2019-03-11]
  - Updated graphic profile.
  - Added FCM information.

1.3 Document Perimeter

Neither economical questions nor questions regarding agreements/contracts will be dealt with in this document.

The information given in this document may change without notice and describes only the matters defined in the general part of this document. Please verify that your company has the most recent version. This information is intended for the use of customers and partners of 21st Century Mobile. The information or statements given in this document concerning the suitability, capacity or performance of the mentioned service cannot be considered binding but shall be defined in the agreement concluded between 21st Century Mobile and the customer, if applicable. 21st Century Mobile shall not be responsible in any event for errors in this document or for any damages, incidental or consequential (including monetary losses), that might arise from the use of this publication or the information in it. This material and the service described in this document are copyrighted in accordance with the applicable laws.

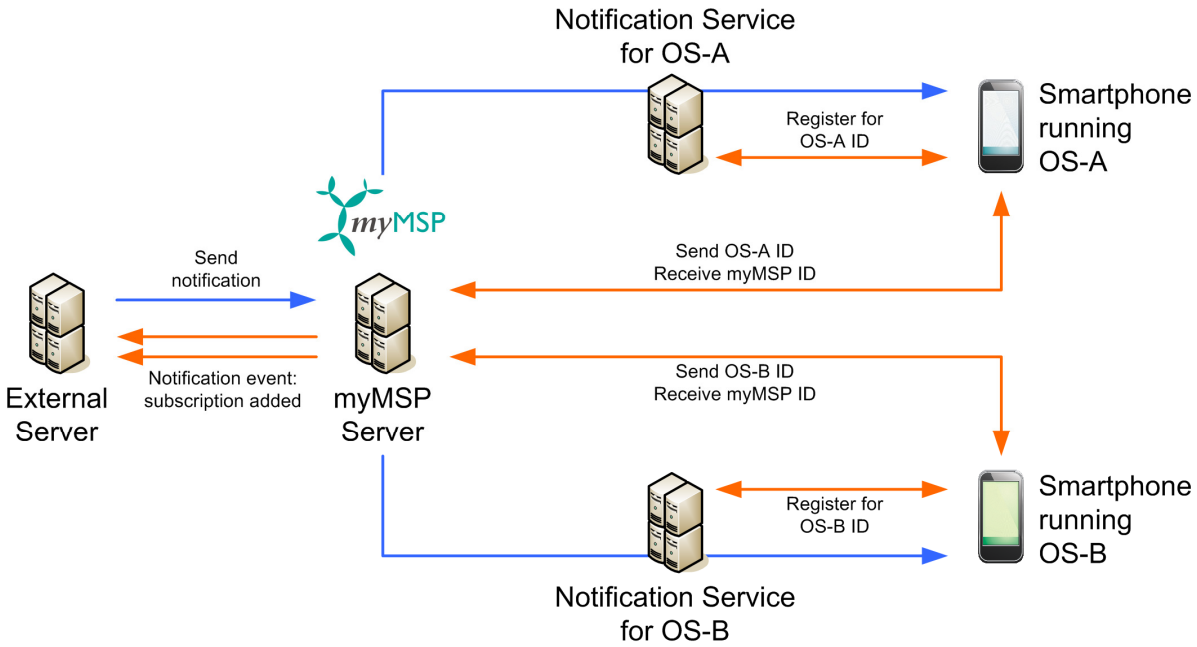
2 Technical description

Device push notifications (DPN) make it possible to send lightweight messages through the Internet to apps running on, primarily, mobile devices such as smart phones and tablet computers. What sets notifications apart from other forms of mobile communication methods such as SMS and e-mail is that they can be targeted at an individual app running on a specific device. Notifications can be used to send simple text messages or to inform the app that there is new data to be fetched from an external server.

Many operating systems support notification technology but each implementation is different, unlike e.g. SMS which is based on a common protocol. This adds complexity to the development of apps that are intended run on more than one operating system, and to the development of server applications that service these apps. myMSP's notification technology enables developers to send notifications to apps through a simple and universal API without having to know what operating system the app is running on.

myMSP currently supports notifications for the following operating systems:

- Apple iOS (APNS)
- Google Android (GCM)
- Google Firebase (FCM)
    - Firebase can replace both GCM and APNS. GCM is deprecated and will not be supported after 2019-04-01.



A simplified overview of the subscription- and the notification-process.

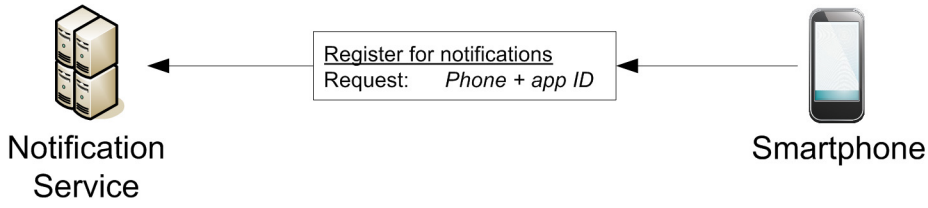
3 Usage scenarios

The following scenarios are intended to give a brief overview of how an *app* and an *external server*, that services the app, can use the myMSP DPN interface.

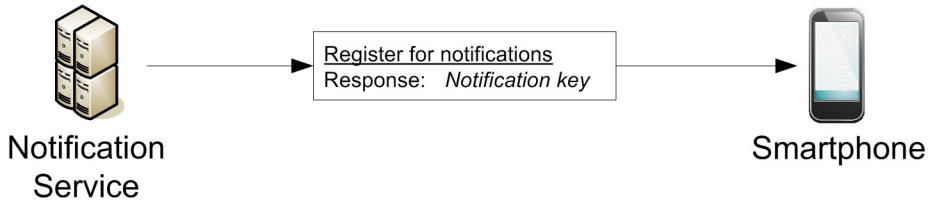
3.1 Register app with notification service provider

Before an app can received notifications it needs to register with a *notification service provider*. There are different providers depending on which device and operating system the app is running on, but the registration procedures are similar.

1. The app sends a request to the notification service provider containing an ID for the device and for the app.



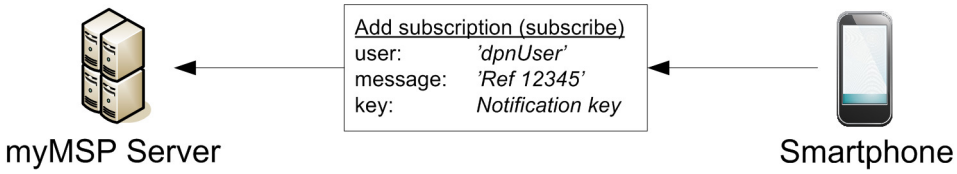
2. The notification service provider generates a *key* that acts as a unique ID for the app running on the device. The notification provider uses this key to route notifications to the specific app instance.



3.2 Add subscription to myMSP

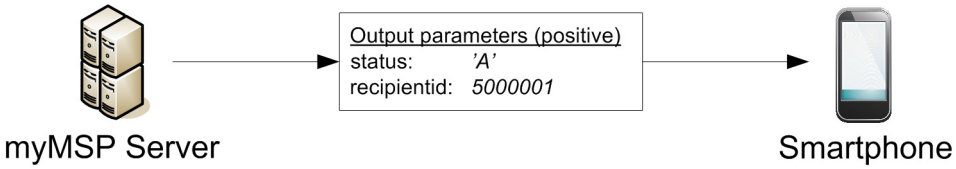
In order to receive notifications the app needs to register the key generated by the notification service provider with myMSP.

1. The app send a “add subscription”-request to myMSP and includes its’ notification key. The app can include a text *message* in the request that will be forwarded to the external server.



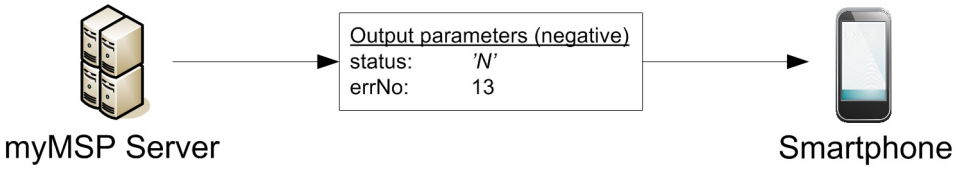
2. myMSP returns a response.

   a. If myMSP has accepted the request then it creates a new recipient and returns a reference called *recipient ID* to the app. The app should store this ID and include it in all future request sent to myMSP.

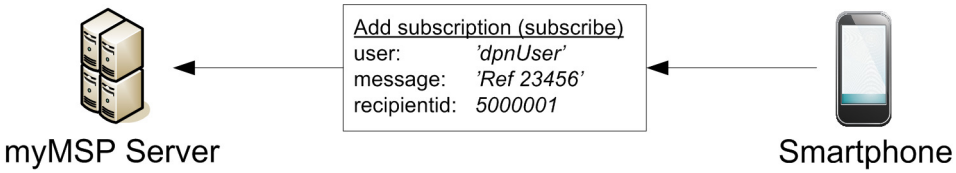


   myMSP also creates a *notification event message* informing the external server that a subscription has been added. The event message includes the recipient ID, the type of *action* that the event describes (add/remove a subscription) and the text message forwarded from the app. myMSP can either send the event message to a listener that is set up at the external server or wait for the external server to fetch it using one of myMSP's APIs.

   b. If myMSP has rejected the request (e.g. the key was not included) a specific error code is returned. A notification event message is not created.



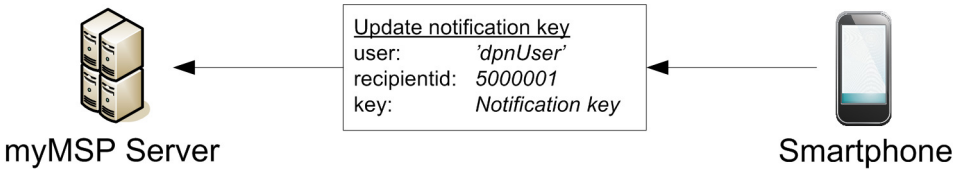
3. The app can add multiple subscriptions as a way to send messages to the external server, but after it has received a recipient ID from myMSP (in the response to the first subscription request) the ID must be included in future request. Each subsequent subscription generates a notification event message for the external server.



3.3 Update notification key

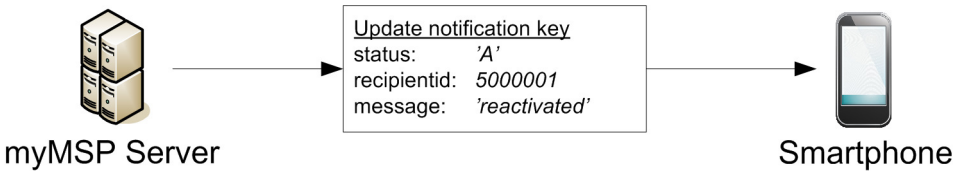
Notification service providers occasionally force apps to re-register for notifications which results in the generation of a new notification key. The new key must be sent to myMSP so that it can update the recipient to ensure that notifications sent to the app are not rejected or lost.

1. To update an existing recipient's key its' ID must be included in the request along with the new notification key. Since the external server only references the recipient ID when sending notifications myMSP does NOT create a event message informing the external server that the key has been updated.

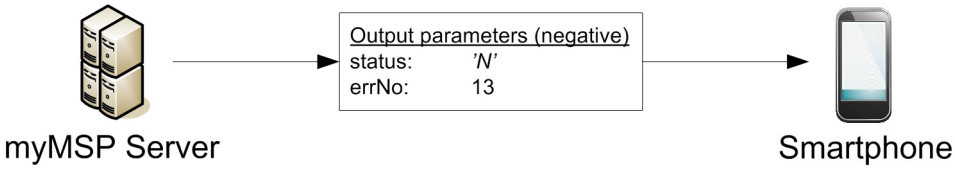


2. myMSP returns a response.

   a. If myMSP has accepted the request then the recipient ID and a *message* parameter are returned. If the recipient was inactive it is reactivated and the message parameter is set to "*reactivated*".



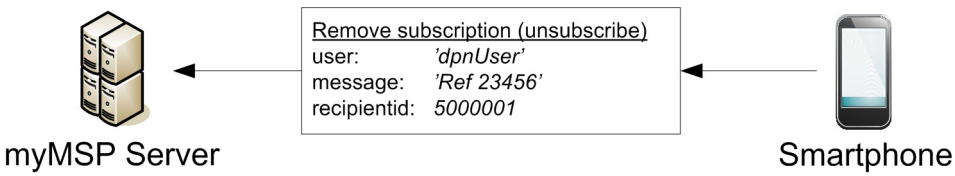
   b. If myMSP has rejected the request a specific error code is returned.



3.4 Remove subscription from myMSP

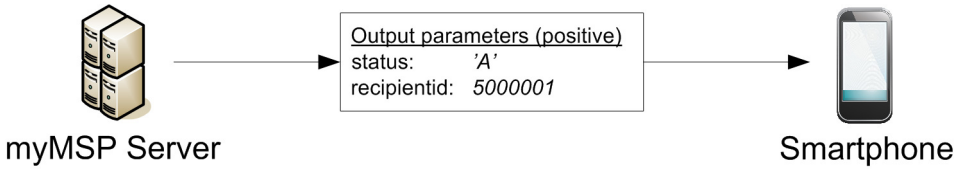
The app can notify the external server that it wants to remove a subscription by sending a "unsubscribe"-request to myMSP. The recipient is NOT deactivated as a result of this request.

1. The app send a "unsubscription"-request to myMSP and includes its' recipient ID. The app can include a text message in the request that will be forwarded to the external server.

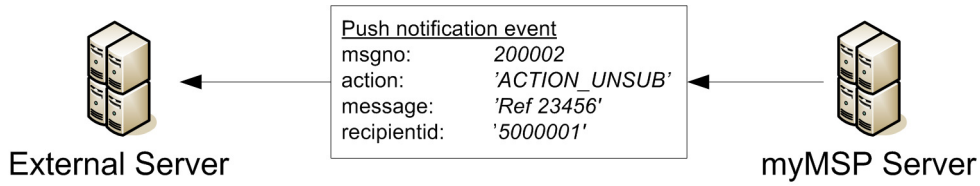


2. myMSP returns a response.

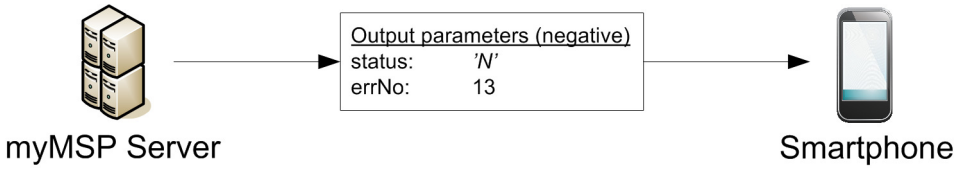
   a. If myMSP has accepted the request then the recipient ID is returned to the app.



   myMSP creates a notification event message informing the external server that a subscription has been removed.



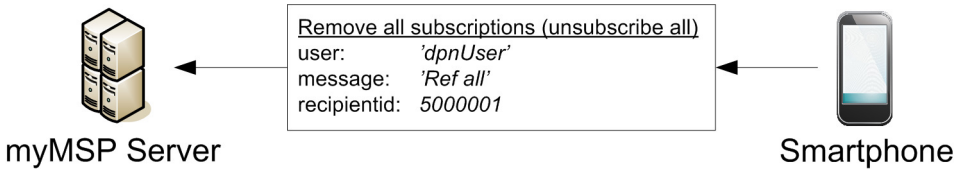
   b. If myMSP has rejected the request a specific error code is returned. A notification event message is not created.



3.5 Remove all subscriptions from myMSP

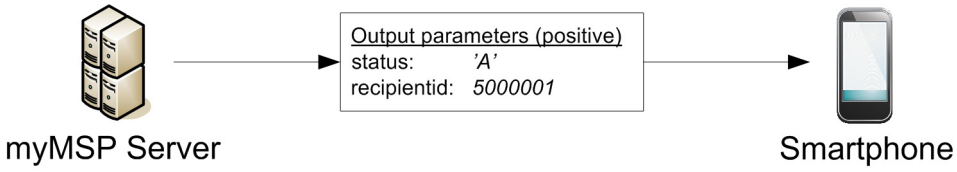
The app can notify the external server that it wants to remove all of its' subscriptions by sending a "unsubscribe all"-request to myMSP. The recipient is deactivated as a result of this request.

1. The app send a "unsubscribe all"-request to myMSP and includes its' recipient ID. The app can include a text message in the request that will be forwarded to the external server.

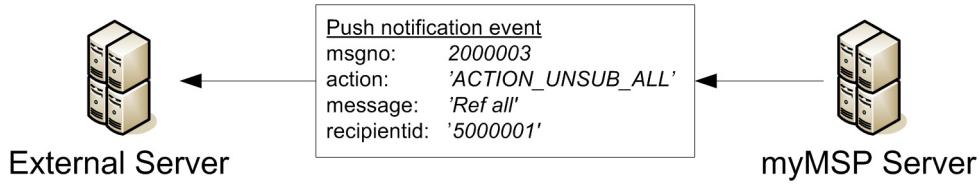


2. myMSP returns a response.

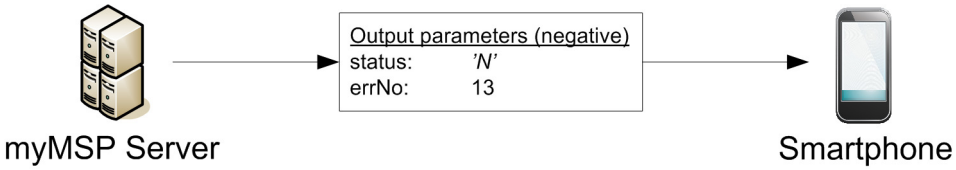
   a. If myMSP has accepted the request then the recipient ID is returned to the app.



myMSP creates a notification event message informing the external server that all subscriptions for a recipient have been removed.



b. If myMSP has rejected the request a specific error code is returned. A notification event message is not created.



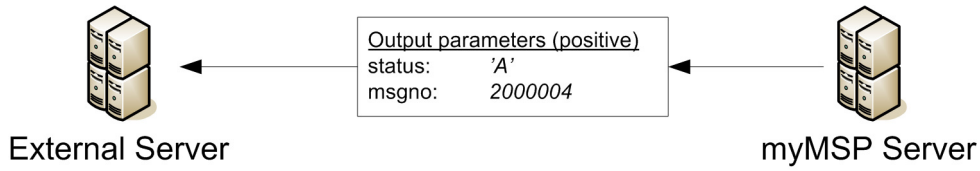
3.6 Send notification

When apps register subscriptions with myMSP notification event messages are created and sent to/fetched by the external server. The recipient ID of an app is included in each event message. When the external server wants to send notifications to apps that have subscribed it only need to include the recipient IDs associated with the subscriptions and does not need to keep track of which operating system and notification service the apps are using.

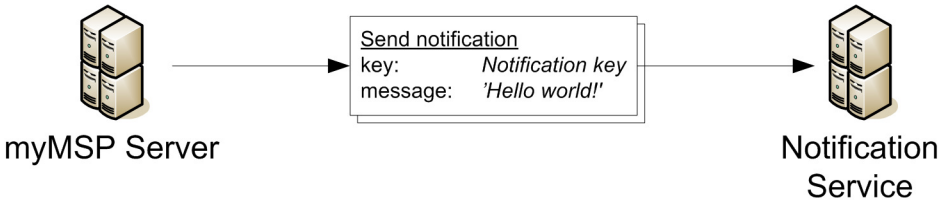
1. The external server sends a notification to myMSP and includes a list of recipients and a message.

2. myMSP returns a response.

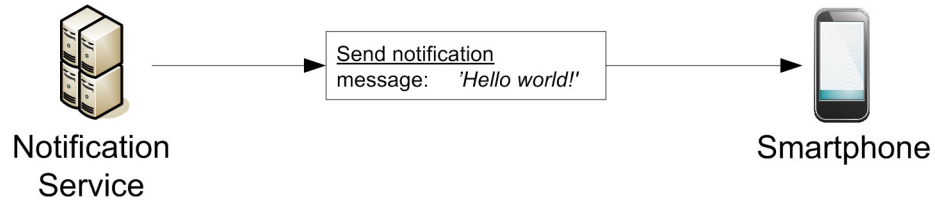
   a. If the request has been accepted then the *message number* that myMSP has assigned the notification is returned.



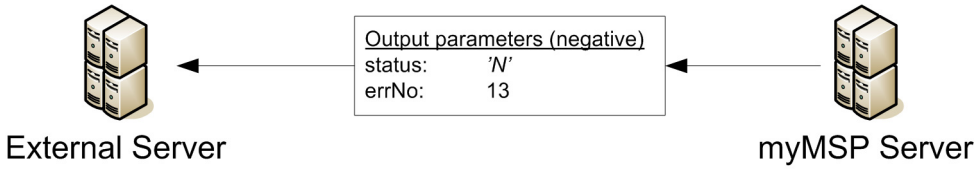
myMSP send the notification to each recipient through their respective notification services. The keys that the apps sent to myMSP (see 3.2) are included in the requests to the notification services.



The notification services route the notifications to the correct app using the key included in the request from myMSP. The notification appears in the device that is running the app. If the device is not connected to the Internet the notification service queues the notification for later delivery.



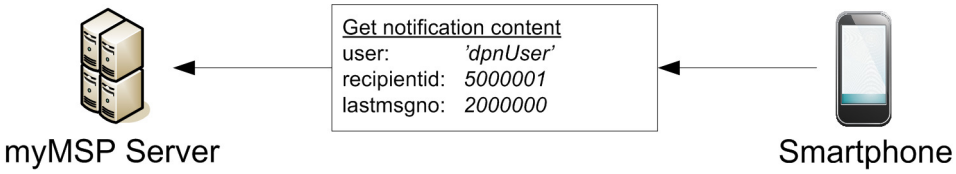
b. If myMSP has rejected the request a specific error code is returned to the external server.



3.7 Get notification content

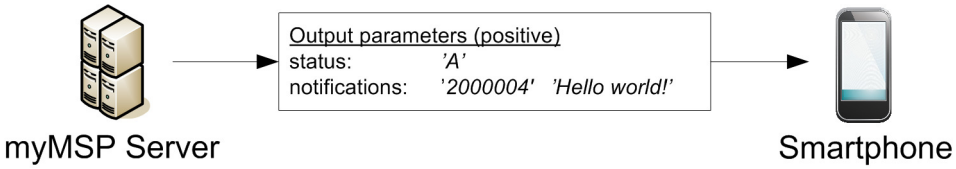
Since the maximum size of a notification varies for each notification service myMSP might truncate the text when the notification is sent. Therefore the app must fetch the content from myMSP after it has received a notification.

1. The app receives one or more notifications and sends a request to myMSP in order to get the contents. The app should include the message number of the last notification that was fetched so that only new notifications are returned.

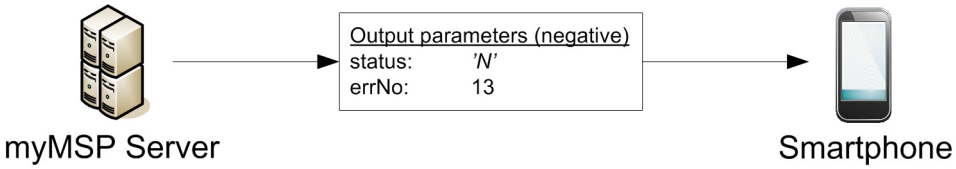


2. myMSP returns a response.

   a. If myMSP has accepted the request then all new notifications are returned.



   b. If myMSP has rejected the request a specific error code is returned.



4 DPN methods for apps

This chapter describes the various methods of the DPN HTTP interface that should be used by apps to communicate subscription and registration changes to myMSP.

4.1 Add subscription (subscribe)

Adds a new notification subscription and a new recipient, or adds a new notification subscription for an already existing recipient. Also creates an inbound DPN message that notifies the external server that a new subscription has been added.

Connection: https://mymsp.21st.se/external/notifications/subscribe

4.1.1 Input parameters

user	User account
message	[Optional] A plain text message that can be used to forward information from the app to the external server. The max length is 4000 bytes.
recipientid	[Optional] If the app has not registered a subscription before this parameter should be left empty. If the app has previously registered a subscription and received a recipient ID from myMSP then ID should be included in the new subscription request.
spid	The name of the notification service the app is registered to. Valid values are: *apns:* for an app running on an iOS device. *gcm:* for an app running on an Android device. *fcm:* for an app running on either an Android or iOS device.
key	The key/ID that the notification service assigns to the app instance. It is returned when the app is registered with the notification service. Each notification service uses a different name for their key values, which are: *Token:* for an app using APNS, i.e. running on an iOS device. *Registration ID:* for an app using FCM/GCM, i.e. running on an Android device.

4.1.2 Output parameters (positive result)

A	Character 'A'
recipientid	Recipient identification number, i.e. the ID that myMSP uses to identify the app instance.

4.1.3 Output parameters (negative result)

N	Character 'N'
errNo	The error codes: 7: Authorization missing 13: Unknown error 22: No value in parameter 'user'

	23: Parameter 'user' missing 412: Invalid 'recipientid' value 46: Cannot find a recipient with the supplied recipient ID 60: Invalid 'spid' value 61: Invalid 'key' value 63: The length of the 'message' value is exceeding the allowed amount of bytes (max 4000)

4.2 Update notification service key (updatekey)

When a notification service updates the key it is using to identify an app, the app must send the new key along with its' recipient ID to myMSP. If the recipient is inactive this request reactivates it.

Connection: https://mymsp.21st.se/external/notifications/updatekey

4.2.1 Input parameters

user	User account
recipientid	The recipient ID that myMSP included in the response to the initial subscription request from the app.
spid	The name of the notification service the app is registered to. The value replaces the recipient's old spid. See 4.1.1 for valid values.
key	The new key/ID that the notification service has assigned the app. It replaces the recipient's old key. See 4.1.1 for more information regarding keys/IDs.

4.2.2 Output parameters (positive result)

A	Character 'A'
recipientid	Recipient identification number, i.e. the ID that myMSP uses to identify the app instance.
message	The text 'reactivated' if the recipient was reactivated by the update-request.

4.2.3 Output parameters (negative result)

N	Character 'N'
errNo	The error codes: 7: Authorization missing 13: Unknown error 22: No value in parameter 'user' 23: Parameter 'user' missing 412: Invalid 'recipientid' value 46: Cannot find a recipient with the supplied recipient ID 60: Invalid 'spid' value 61: Invalid 'key' value

4.3 Remove subscription (unsubscribe)

Removes a notification subscription by creating an inbound DPN message that notifies the external server that the app has removed one of its' subscriptions. Unsubscription information can be passed to the external server through the "message" parameter. The recipient is not deactivated as a result of this request. If the recipient is already inactive no unsubscription message is created.

Connection: https://mymsp.21st.se/external/notifications/unsubscribe

4.3.1 Input parameters

user	User account
message	[Optional] A plain text message that can be used to forward information from the app to the external server. The max length is 4000 bytes.
recipientid	The recipient ID that myMSP included in the response to the initial subscription request from the app.
spid	The name of notification service the app is registered to. See 4.1.1 for valid values.

4.3.2 Output parameters (positive result)

A	Character 'A'
recipientid	Recipient identification number, i.e. the ID that myMSP uses to identify the app instance.

4.3.3 Output parameters (negative result)

N	Character 'N'
errNo	The error codes: 7: Authorization missing 13: Unknown error 22: No value in parameter 'user' 23: Parameter 'user' missing 412: Invalid 'recipientid' value 46: Cannot find a recipient with the supplied recipient ID 60: Invalid 'spid' value

4.4 Remove all subscriptions (unsubscribeall)

Removes all notification subscription for a recipient by creating an inbound DPN message that notifies the external server that the app has removed all of its' subscriptions. Unsubscription information can be passed to the external server through the "message" parameter. The recipient is deactivated as a result of this request. If the recipient is already inactive no unsubscription message is created.

Connection: https://mymsp.21st.se/external/notifications/unsubscribeall

4.4.1 Input parameters

user	User account
message	[Optional] A plain text message that can be used to forward information from the app to the external server. The max length is 4000 bytes.
recipientid	The recipient ID that myMSP included in the response to the initial subscription request from the app.
spid	The name of notification service the app is registered to. See 4.1.1 for valid values.

4.4.2 Output parameters (positive result)

A	Character 'A'
recipientid	Recipient identification number, i.e. the ID that myMSP uses to identify the app instance.

4.4.3 Output parameters (negative result)

N	Character 'N'
errNo	The error codes: 7: Authorization missing 13: Unknown error 22: No value in parameter 'user' 23: Parameter 'user' missing 412: Invalid 'recipientid' value 46: Cannot find a recipient with the supplied recipient ID 60: Invalid 'spid' value

4.5 Get notification content (get)

Since the maximum size of a notification varies for each notification service myMSP might truncate the text when the notification is sent. Therefore the app should always send a “get”-request after it has received a notification in order to fetch the entire content from myMSP. See 5.1 of more information regarding sending notifications.

Connection: https://mymsp.21st.se/external/notifications/get

4.5.1 Input parameters

user	User account
recipientid	Recipient identification number, i.e. the ID that myMSP uses to identify the app instance.
lastmsgno	Message number in myMSP used to identify notifications. The app should store the message number with the highest value after getting the content of sent notifications and include it in the next request so that the same notifications are not returned again. A value of “0” fetches all notifications.
sentafter	[Optional] A time stamp, format: yyyyMMddHHmmssSSS. Messages sent before this time will not be returned. This parameter can be used to prevent the app from fetching outdated notifications that have accumulated if the app has not been active for a long time.

4.5.2 Output parameters (positive result)

A	Character ‘A’
notifications	A list of the contents of notifications sent to the app. Each notification is displayed on a separate line. The notification parameters are separated by a tab-character ( ). The fields are: *Message number (msgno)* *Sent time (format: yyyyMMddHHmmssSSS)* *Content* Example: 12556 20120101151617000 A notification message 12557 20120101161718000 Another notification message

4.5.3 Output parameters (negative result)

N	Character ‘N’
errNo	The error codes: 7: Authorization missing 13: Unknown error 22: No value in parameter ‘user’ 23: Parameter ‘user’ missing 391: Invalid ‘msgno’ value

	392: Parameter 'msgno' missing 412: Invalid 'recipientid' value 46: Cannot find a recipient with the supplied recipient ID 66: Invalid 'sentafter' value

5 DPN methods for external server

This chapter describes the various methods of the DPN interface that external servers can use to send notifications and receive subscription updates from myMSP.

5.1 Send notification (send)

Send a notification to one or more recipients. When the app receives the notification it must send a "get"-request (see 4.5) to myMSP in order to fetch the entire content of the message.

Connection: https://mymsp.21st.se/external/notifications/send

5.1.1 Input parameters

user	User account
pwd	Password
message	The message in plain text. The max length is 4000 bytes.
recipients	Recipient IDs and/or names/short names of a recipient group. Should be in the form of a comma-separated string, e.g. "2000001, 2000002,recipgroupA". All none DPN-recipients are filtered out.
delaywhileidle	[Only FCM/GCM, optional] If included, indicates that the message should not be sent immediately if the device is idle. The FCM/GCM server will wait for the device to become active, and then only the last message for each collapse_key (see below) value will be sent. Valid values: "true" or "false". The default value is "false".
collapsekey	[Only FCM/GCM, optional] An arbitrary string that is used to collapse a group of like messages when the device is offline, so that only the last message gets sent to the client. This is intended to avoid sending too many messages to the phone when it comes back online. FCM/GCM allows a maximum of 4 different collapse keys to be used by the FCM/GCM server at any given time. In other words, the FCM/GCM server can simultaneously store 4 different send-to-sync messages per device, each with a different collapse key. If you exceed this number FCM/GCM will only keep 4 collapse keys, with no guarantees about which ones they will be. The max length is 400 characters.
timetolive	[Only FCM/GCM, optional] Specifies how long (in seconds) the message should be kept on FCM/GCM storage if the device is offline. The default time-to-live is 4 weeks, i.e. 2419200 seconds. This is also the max value.
restrictedpackagename	[Only FCM/GCM, optional] Specifies a string containing the package name of your app. When set, messages are only sent to registration IDs that match the package name. The max length is 400 characters.

5.1.2 Output parameters (positive result)

A	Character 'A'
msgno	Message identification number in myMSP.

5.1.3 Output parameters (negative result)

N	Character 'N'
errNo	The error codes: 7: Authorization missing 11: Maximum number of messages exceeded 12: Not enough credits left to send message 13: Unknown error 22: No value in parameter 'user' 23: Parameter 'user' missing 24: No value in parameter 'pwd' 25: Parameter 'pwd' missing 28: No valid values in parameter 'recipients' 63: Invalid 'message' value: cannot be longer then 4000 bytes 64: Invalid 'delaywhileidle' value 65: Invalid 'collapsekey' value 67: Invalid 'timetolive' value 68: Invalid 'restrictedpackagename' value

5.2 Get notification events through HTTP push

When a subscription is added or updated myMSP creates a message of the event that can be sent to an external server through myMSP's push interface. The following are the relevant parameters that are sent as an HTTP request to a specified URL on the external server when a new event message has been created. See appendix *myMSP_API_v2_Appendix_http_push.pdf* for more information regarding the HTTP push interface.

Notification events can also be retrieved using myMSPs HTTPS- (see 5.3) or WebService-APIs (see 5.4).

messageType	3 (request with other message types can be ignored).
msgNo	Message identification number in myMSP.
createTime	Time and date when the event message was created in myMSP. Format: yyyy-MM-dd HH:mm
creatorName	Name of the myMSP account.
initialId	A code for the action that generated the event message. Possible values are: ACTION_SUB Generated by a "Add subscription"-request (see 4.1) from the app. ACTION_UNSUB Generated by a "Remove subscription"-request (see 4.3) from the app. ACTION_UNSUB_ALL Generated by a "Remove all subscriptions"-request (see 4.4) from the app or by myMSP if a "unregister"-request is received from the notification server for a recipient.
originatorText	The recipient ID of the app that generated the event.
destination	The URL that the request was sent to, e.g. "/external/notifications/subscribe". Empty if myMSP generated an ACTION_UNSUB_ALL-event.
subject	A message sent by the app to accompany the event. If myMSP generated an ACTION_UNSUB_ALL-event the message will be "unsub [recipientid]".

5.3 Get notification events through HTTPS retrieve received messages

When a subscription is added or updated myMSP creates a message of the event that can be fetched by an external server through myMSP's HTTPS interface. Since no message is sent from myMSP when an event is created the external server needs to check for updates regularly. See appendix *myMSP_API_v2_Appendix_https.pdf* for more information regarding the HTTPS interface.

Notification events can also be pushed to the external server using myMSPs HTTP push-API (see 5.2) or retrieved using the WebService-API (see 5.4).

5.3.1 Input parameters

user	User Account
pwd	Password
lastMsgId	Message identification number in myMSP. All event messages with a higher message number (i.e. event messages generated later) are returned. Input 0 to fetch all event messages.
clean	Set to "true" to wash all return values, i.e. all new line characters (\r\n, \n, \r) are replaced with the string "\n" and all tab characters (\t) with a space.

5.3.2 Output parameters (positive result)

A	Character 'A'
STATUS	All event messages with a msgno-value higher than the one supplied in the request are listed. Two semicolon (;;) are placed in front of each message. The different fields of a received message are separated by a tab-character ( ). The fields are: *Message number* *Received date* (format: YYYY-MM-DD hh:mm) *User name* *Device notification action* (see the "action"-parameter in 0 for possible values) *Message type* (will always return "DPN") *Originator number* (will always return "null") *Recipient ID* *Subject* (a message sent by the app to accompany the event. If myMSP generated an ACTION_UNSUB_ALL-event the message will be "unsub [recipientid]".) *Message text* (same value as "Subject" field) Example: ;;12556 2012-01-01 15:16 dpnuser ACTION_SUB DPN null 90001 Msg A Msg A ;;12557 2012-01-01 16:17 dpnuser ACTION_SUB DPN null 90002 Msg B Msg B

5.3.3 Output parameters (negative result)

N	Character 'N'
errNo	The error code: 7: Authorization missing 11: Maximum number of messages exceeded 13: Unknown error 22: No value in parameter 'user'

	23: Parameter 'user' missing 24: No value in parameter 'pwd' 25: Parameter 'pwd' missing 391: Invalid 'msgno' value 392: Parameter 'msgno' missing

5.4 Get notification events through WebService retrieve received messages

When a subscription is added or updated myMSP creates a message of the event that can be fetched by an external server through myMSP's WebService interface. Since no message is sent from myMSP when an update is added the external server needs to check for updates regularly. See appendix *myMSP_API_v2_Appendix_ws.pdf* for more information regarding the WebService interface.

Notification updates can also be pushed to the external server using myMSPs HTTP push-API (see 5.2) or retrieved using the HTTPS-API (see 5.3).

5.4.1 Input Parameters (ReceivedMsgRequest)

Identification	A class containing information that identifies the user and the type of client that is used. See WebService appendix for more information.
lastMsgNo	Message identification number in myMSP. All event messages with a higher message number (i.e. event messages generated later) are returned. Input 0 to fetch all event messages.

5.4.2 Output Parameters (ReceivedMsgResponse)

resultCode	See result codes in main API-document (3.3)!
resultDescription	Text describing the results.
messagesReceived	An array (MessageReceived []) that contains received messages. See WebService appendix for more information.

5.4.3 Output class MessageReceived

msgNo	Message identification number in myMSP.
createTime	Time and date when the event message was created in myMSP.
creatorName	Name of the myMSP account.
initialId	A code for the action that generated the event message. See the "action"-parameter in 0 for possible values.
msgType	Will always return "DPN".
originator	Will always return null.
originatorText	The recipient ID of the app that generated the event.
subject	A message sent by the app to accompany the event. If myMSP generated an ACTION_UNSUB_ALL-event the message will be "unsub [recipientid]".)
smsText	Same value as "subject" field.

myMSP API v2 Appendix URL shortener

System Interface - Specifications

21st Century Mobile
Marketing & Sales
+46 (0)8 21 21 55
sales@21st.se

www.21st.se
Research & Development
+46 (0)8 21 21 55
tech@21st.se

1 Purpose of Document 3
1.1 Author 3
1.2 Document Status 3
1.3 Document Perimeter 3
2 Technical description 4
3 HTTPS methods 5
3.1 Create shortened URL (create) 5
3.1.1 Input parameters 5
3.1.2 Output parameters (positive result) 5
3.1.3 Output parameters (negative result) 6
3.2 Get shortened URLs (get) 6
3.2.1 Input parameters 6
3.2.2 Output parameters (positive result) 6
3.2.3 Output parameters (negative result) 6
3.3 Get all shortened URLs (getall) 7
3.3.1 Input parameters 7
3.3.2 Output parameters (positive result) 7
3.3.3 Output parameters (negative result) 7
3.4 Remove shortened URL (remove) 7
3.4.1 Input parameters 7
3.4.2 Output parameters (positive result) 7
3.4.3 Output parameters (negative result) 8

1 Purpose of Document

The purpose of this document is to define the HTTPS interface made available by myMSP to access the URL shortening functionality. Accessing myMSP can be done from the computer systems of external parties/customers.

1.1 Author

21st Century Mobile
Mikael Rosvall
mikael.rosvall@21st.se
+46 (0)8 21 21 55

1.2 Document Status

- Published 2020-04-14

1.3 Document Perimeter

Neither economical questions nor questions regarding agreements/contracts will be dealt with in this document.

The information given in this document may change without notice and describes only the matters defined in the general part of this document. Please verify that your company has the most recent version. This information is intended for the use of customers and partners of 21st Century Mobile. The information or statements given in this document concerning the suitability, capacity or performance of the mentioned service cannot be considered binding but shall be defined in the agreement concluded between 21st Century Mobile and the customer, if applicable. 21st Century Mobile shall not be responsible in any event for errors in this document or for any damages, incidental or consequential (including monetary losses), that might arise from the use of this publication or the information in it. This material and the service described in this document are copyrighted in accordance with the applicable laws.

2 Technical description

The URL shortener API is used to create short URLs that forwards all request to longer URLs. This is an efficient way to reduce the length of an SMS message. myMSP uses a default domain name when creating short URLS but custom domains can also be configured. Contact you 21st sales representative to activate this feature for your account.

The interface is realised through HTTPS. An external system can send data to myMSP through HTTPS using the GET or POST methods to submit data. The results are delivered as a HTTPS RESPONSE in the “text/plain” format.

3 HTTPS methods

This chapter describes the various HTTPS methods in more detail.

3.1 Create shortened URL (create)

Creates a new shortened URL that will forward all requests to a long (destination) URL. It is recommended that a POST method is used with the media type set to “application/x-www-form-urlencoded” to ensure that the “destination” parameter is correctly encoded.

Connection: https://mymsp.21st.se/external/urlShortener/create

3.1.1 Input parameters

user	String (64)	User Account
pwd	String (64)	Password
destination	String (4000)	The long URL that the shortened URL should forward all requests to. Must be a valid URL.
domain	String (200)	(OPTIONAL) The domain (including subdomain) of the shortened URL. If empty they default system or (if set) account domain is used.
resource	String (64)	(OPTIONAL) The resource path of the shortened URL, i.e. the part of the URL that is after the domain but before the resource ID. For example, at.21st.se/**thisIsAResource**/ABC123. If empty the resource part of the shortened URL is excluded to that the ID follows directly after the domain (at.21st.se/ABC123).
resourceid	String (64)	(OPTIONAL) The ID that is used to uniquely identify the shortened URL. If empty a random ID is generated.
expires	Integer	(OPTIONAL) The number of days (request time + 24h x expires) until the shortened URL is automatically removed. If empty or zero the URL never expires.

3.1.2 Output parameters (positive result)

A	Character ‘A’
STATUS	The shortened URL along with parameters separated by a tab-character ( ). The parameters are (in order): *Shortened URL ID (unique ID for the shortened URL)* *Shortened URL (the shortened URL)* *Domain (domain part of the shortened URL)* *Resource (resource path part of the shortened URL)* *Resource ID (resource ID part of the shortened URL)* *Create time (when the shortened URL was created)* *Expiration time (when the shortened URL expires)* *Clicks (the number of times the shortened URL has forwarded a request)* Example: `e8cf(…)  at.21st.se/g/abc123  at.21st.se  g  abc123  20200414120530000  20200416120530000  0`

3.1.3 Output parameters (negative result)

N	Character ‘N’
errNo	The error code: 7: Authorization missing 75: URL shortening module not activated for user 13: Unknown error, contact support 22: No value in parameter ‘user’ 23: Parameter ‘user’ missing 24: No value in parameter ‘pwd’ 25: Parameter ‘pwd’ missing 951: Invalid ‘destination’ parameter 952: Parameter ‘domain’ too long 953: Parameter ‘resource’ too long 954: Parameter ‘resourceid’ too long 955: Invalid ‘expires’ parameter 956: A shortened URL with the same domain, resource and resourceid already exists 957: All available resourceids for the domain and resource have been taken.

3.2 Get shortened URLs (get)

Get one or more shortened URLs using IDs.

Connection: https://mymsp.21st.se/external/urlShortener/get

3.2.1 Input parameters

user	String (64)	User Account
pwd	String (64)	Password
id	String (64 per ID)	Shortened URL ID. Included in the response when a new shortened URL is created. Multiple IDs can be submitted as a comma-separated list.

3.2.2 Output parameters (positive result)

A	Character ‘A’
STATUS	See 3.1.2 Example: e8cf(…) at.21st.se/g/abc123 at.21st.se g abc123 20200414120530000 20200416120530000 2 h39p(…) at.21st.se/def456 at.21st.se def456 20200414134520000 1

3.2.3 Output parameters (negative result)

N	Character ‘N’
errNo	The error code: 7: Authorization missing 75: URL shortening module not activated for user 13: Unknown error, contact support 22: No value in parameter ‘user’ 23: Parameter ‘user’ missing 24: No value in parameter ‘pwd’ 25: Parameter ‘pwd’ missing 950: No shortened URL with ‘id’ exists for user

3.3 Get all shortened URLs (getall)

Get all shortened URLs created by a user.

Connection https://mymsp.21st.se/external/sendlater/getall

3.3.1 Input parameters

user	String (64)	User Account
pwd	String (64)	Password

3.3.2 Output parameters (positive result)

A	Character ‘A’
STATUS	See 3.2.2

3.3.3 Output parameters (negative result)

N	Character ‘N’
errNo	The error code: 7: Authorization missing 75: URL shortening module not activated for user 13: Unknown error, contact support 22: No value in parameter ‘user’ 23: Parameter ‘user’ missing 24: No value in parameter ‘pwd’ 25: Parameter ‘pwd’ missing

3.4 Remove shortened URL (remove)

Remove one or more shortened URLs using IDs.

Connection: https://mymsp.21st.se/external/remove

3.4.1 Input parameters

user	String (64)	User Account
pwd	String (64)	Password
id	String (64 per ID)	Shortened URL ID. Included in the response when a new shortened URL is created. Multiple IDs can be submitted as a comma-separated list.

3.4.2 Output parameters (positive result)

A	Character ‘A’

3.4.3 Output parameters (negative result)

N	Character ‘N’
errNo	The error code: 7: Authorization missing 75: URL shortening module not activated for user 13: Unknown error, contact support 22: No value in parameter ‘user’ 23: Parameter ‘user’ missing 24: No value in parameter ‘pwd’ 25: Parameter ‘pwd’ missing 950: No shortened URL with ‘id’ exists for user