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

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.

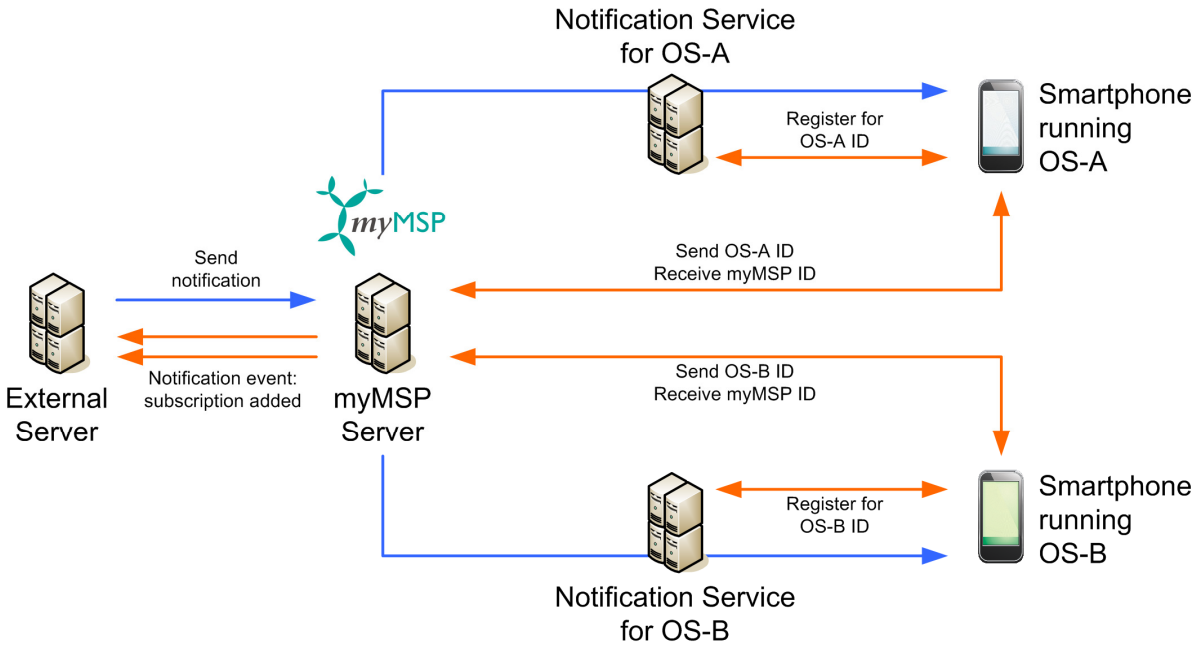
2 Technical description

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.

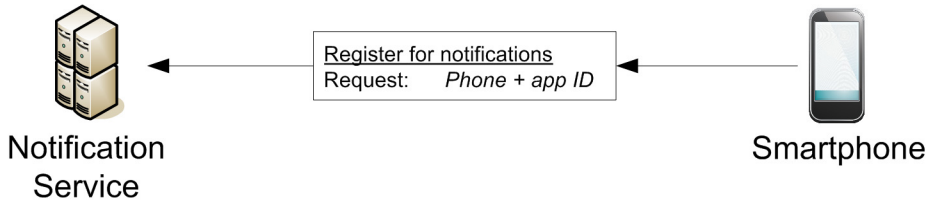
3 Usage scenarios

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.

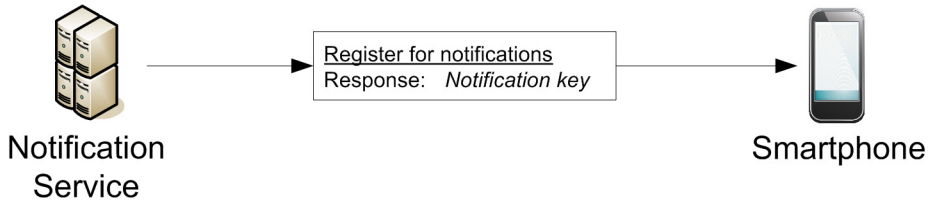
3.1 Register app with notification service provider

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.



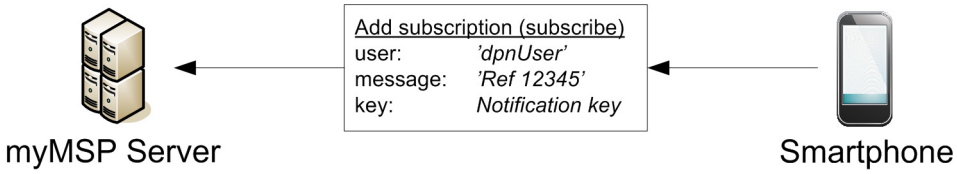
2. The notification service provider generates a *key* that acts as a unique ID for the app running on the device. The notification provider uses this key to route notifications to the specific app instance.



3.2 Add subscription to myMSP

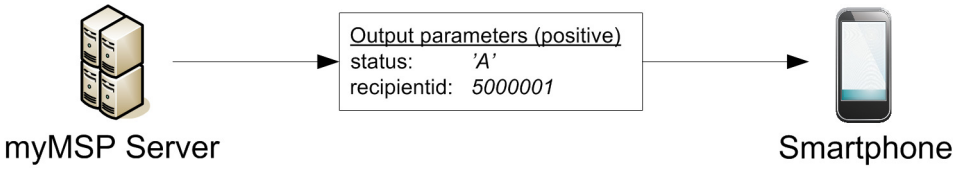
In order to receive notifications the app needs to register the key generated by the notification service provider with myMSP.

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.



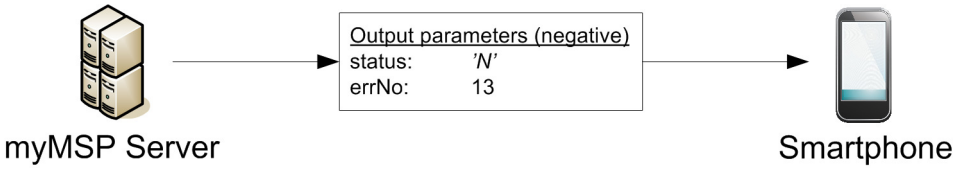
2. myMSP returns a response.

   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.

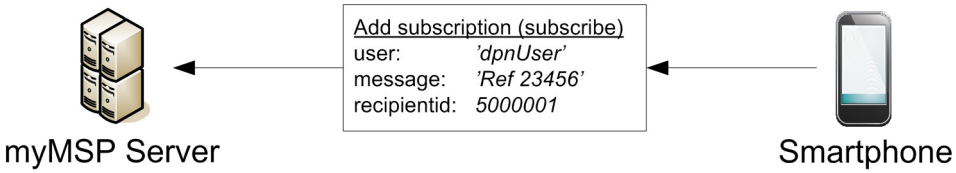


   myMSP also creates a *notification event message* informing the external server that a subscription has been added. The event message includes the recipient ID, the type of *action* that the event describes (add/remove a subscription) and the text message forwarded from the app. myMSP can either send 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.



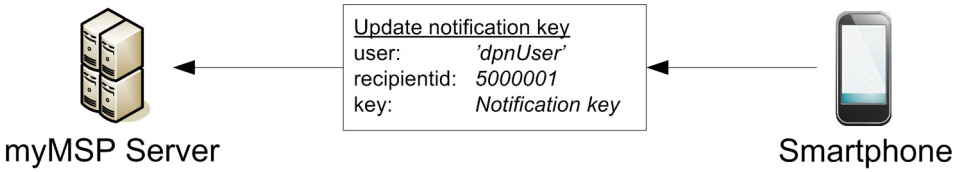
3. The app can add multiple subscriptions as a way to send messages to the external server, but after it has received a recipient ID from myMSP (in the response to the first subscription request) the ID must be included in future request. Each subsequent subscription generates a notification event message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



3.3 Update notification key

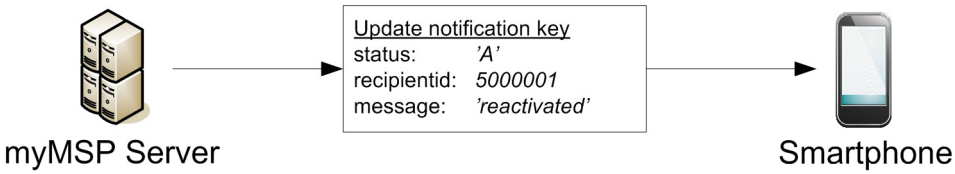
Notification service providers occasionally force apps to re-register for notifications which results in the generation of a new notification key. The new key must be sent to myMSP so that it can update the recipient to ensure that notifications sent to the app are not rejected or lost.

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.

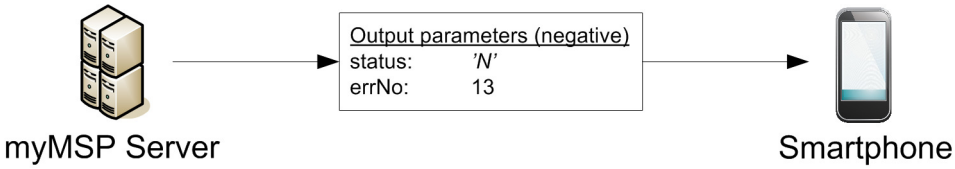


2. myMSP returns a response.

   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*".



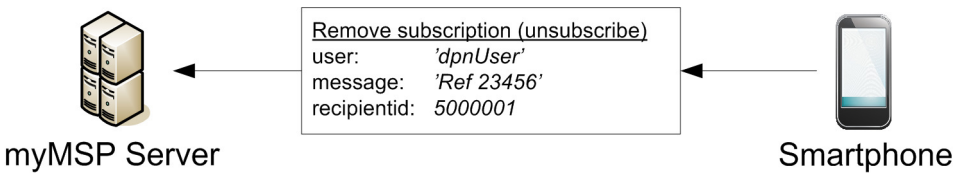
   b. If myMSP has rejected the request a specific error code is returned.



3.4 Remove subscription from myMSP

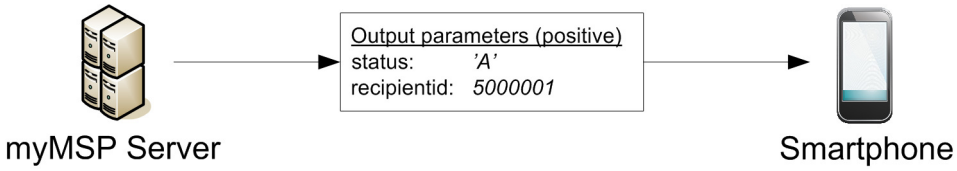
The app can notify the external server that it wants to remove a subscription by sending a "unsubscribe"-request to myMSP. The recipient is NOT deactivated as a result of this request.

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.

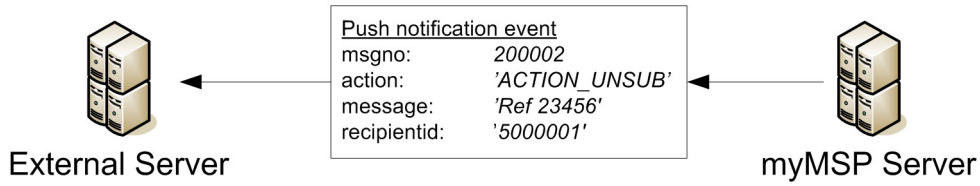


2. myMSP returns a response.

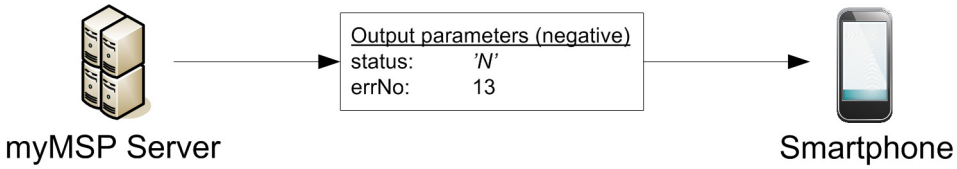
   a. If myMSP has accepted the request then the recipient ID is returned to the app.



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



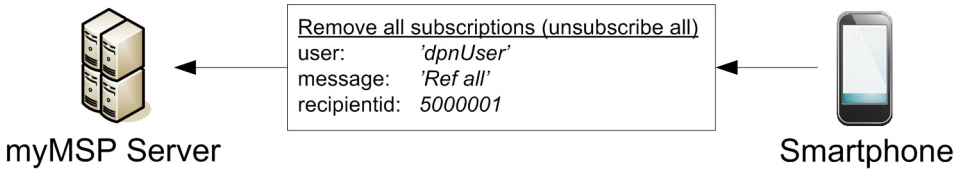
   b. If myMSP has rejected the request a specific error code is returned. A notification event message is not created.



3.5 Remove all subscriptions from myMSP

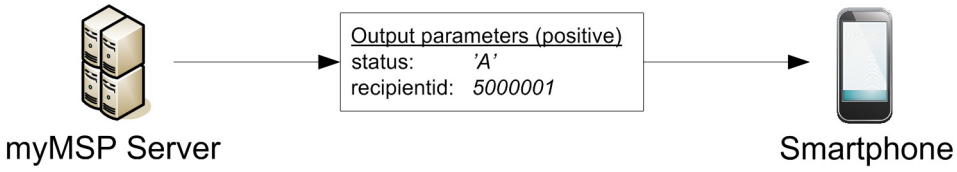
The app can notify the external server that it wants to remove all of its' subscriptions by sending a "unsubscribe all"-request to myMSP. The recipient is deactivated as a result of this request.

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.

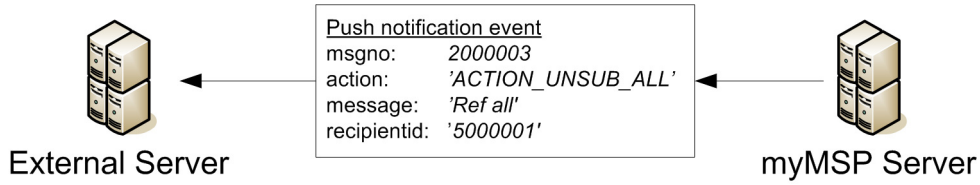


2. myMSP returns a response.

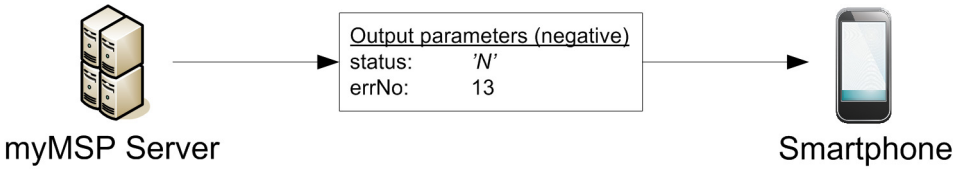
   a. If myMSP has accepted the request then the recipient ID is returned to the app.



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.



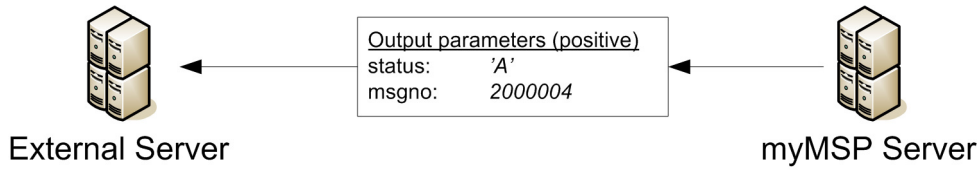
3.6 Send notification

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.

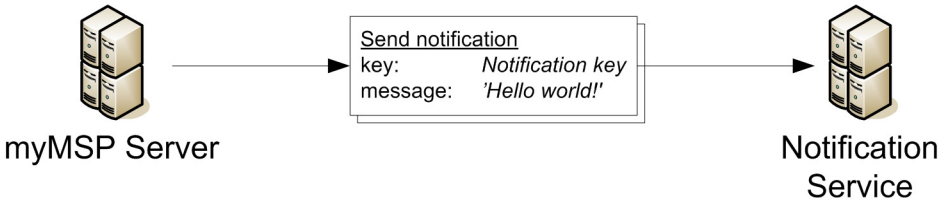
1. The external server sends a notification to myMSP and includes a list of recipients and a message.

2. myMSP returns a response.

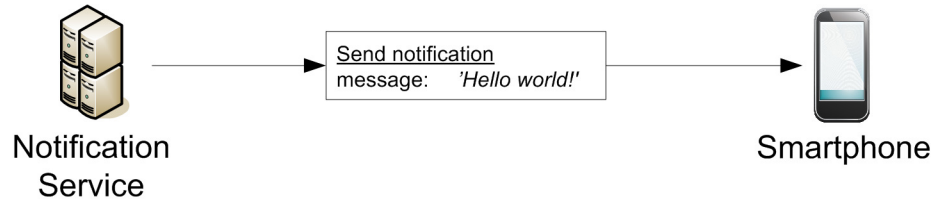
   a. If the request has been accepted then the *message number* that myMSP has assigned the notification is returned.



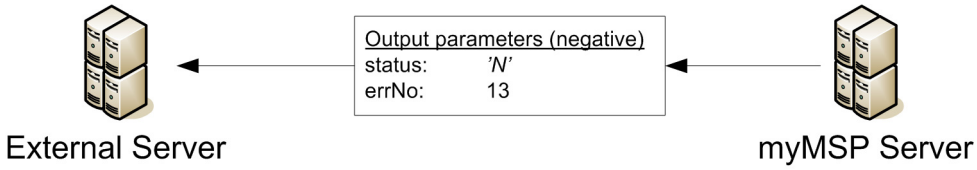
myMSP send the notification to each recipient through their respective notification services. The keys that the apps sent to myMSP (see 3.2) are included in the requests to the notification services.



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.



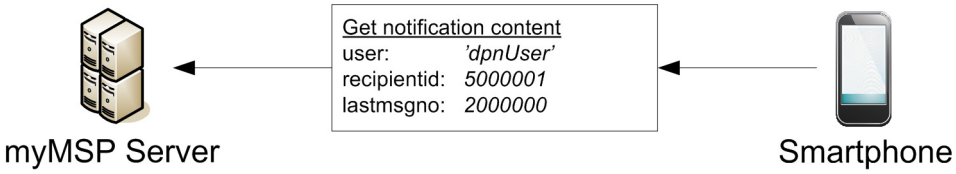
b. If myMSP has rejected the request a specific error code is returned to the external server.



3.7 Get notification content

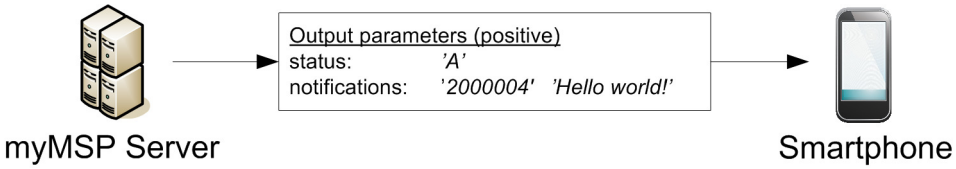
Since the maximum size of a notification varies for each notification service myMSP might truncate the text when the notification is sent. Therefore the app must fetch the content from myMSP after it has received a notification.

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.

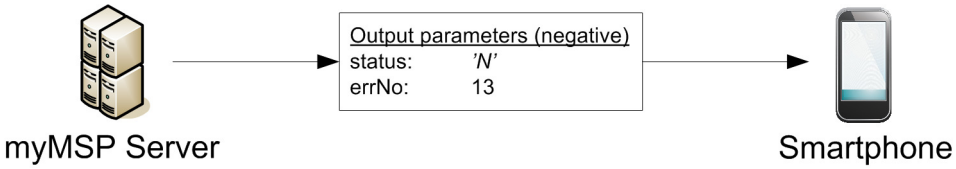


2. myMSP returns a response.

   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.