ERICSSON

IoT Accelerator Developer Portal

Working with STOMP Notifications

Instruction

Contents

1 Purpose and scope

This document is in the scope of IoT Accelerator Service.

Purpose of the document is to describe the Push STOMP Notification Interface details, the STOMP procedures and the use cases to be considered for a STOMP client that could consume Service Portal STOMP notifications.

2 Overview of the push STOMP Notification Interface

Ericsson IoT Accelerator is using Simple Text Orientated Messaging Protocol (STOMP) to push notifications when certain events occur. The Service Portal supports several categories of events that triggers STOMP notifications. These are described in the Use Cases section of this document.

Application clients can be developed to subscribe to the Service Portal message broker, to analyze and to take actions, if necessary, when STOMP notifications are pushed.

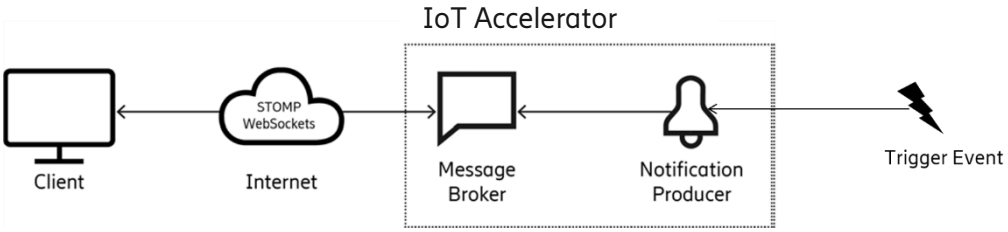


Figure 1: The Push STOMP Notifications Interface Overview

The connection to the message broker it's done over secure Web Sockets using TLS and the standard port 443.

It is assumed the reader is familiar with STOMP protocol:

https://stomp.github.io/stomp-specification-1.2.html.

To use the STOMP notifications generated by IoT Accelerator Service a STOMP client that implements the standard STOMP procedures and APIs must be developed. A sample Client is provided in the Developer Portal.

To access the Push Notification Interface, a Service Portal user account and corresponding permissions are needed, which are provisioned during the enterprise onboarding process.

2.1 Security Tokens:

The following security tokens are associated with the portal account:

— username

— password

— queueName

The following convention is used for queueName:

dcp.<organizationId>.<serviceName>.<version>.<service specific queue name>.

The organizationId (also companyId), which identifies the organizational unit of the queue and the version values are configured by the specific service.

The system needs these tokens to authenticate and authorize the requests over the Push Notification Interface.

The tokens are given to the enterprise user as part of the onboarding process.

Operator Service Desk for Service Portal administration should be contacted to support for the needed account and permissions.

2.2 Authentication:

The STOMP client is authenticated by the message broker when it initiates the STOMP connection by sending a CONNECT frame containing the portal account username and password.

The authentication is based on the following headers in the CONNECT frame:

— login:<the user identifier used to authenticate against a secured STOMP server, that is the portal account username>

— passcode:<the password used to authenticate against a secured STOMP server, that is the portal account password>

2.3 Authorization:

When the STOMP client sends a SUBSCRIBE frame to subscribe to a specific message queue (indicated by the queue name set in the destination header), the message broker checks if the client is authorized to consume messages from the given queue or not.

If the client is not authorized to consume messages from the queue, the message broker responds to the SUBSCRIBE frame with an ERROR frame.

2.4 Message Expiry:

To prevent old non-consumed messages from filling up the queues, messages are posted with a time-to-live set.

The time-to-live configuration value is specified within the service-specific documentation.

STOMP is text-based and uses UTF-8 encoding. The body of the STOMP MESSAGE that carries the payload is JSON text encoded and the MIME media type is set to Content_Type: application/json.

Note: A special Content_Type header is used instead of the standard Content-Type header.

3 STOMP Procedures

Using the credentials of the user granted to consume the STOMP notifications, the STOMP client should implement the following generic procedures:

3.1 Connect to the Message Broker

Establish the connection to the STOMP message broker using secure websockets port 443 and the following URL:

- **Server URL**: wss://<OPERATOR_BRAND_HOST>/dcpapi/push

Initiate the connection to the server by sending a CONNECT frame with the headers:

- **accept-version**: 1.2
- **login**:<the user identifier used to authenticate against the secured STOMP server, that is, the portal account username>
- **passcode**:<the password used to authenticate against the secured STOMP server, that is, the portal account password>
- **heart-beat**:<outgoing heart-beat interval>, <incoming heart-beat interval>

If the server accepts the connection attempt, it responds with a CONNECTED frame. If not, it responds with an ERROR frame.

Heart-beating can optionally be used to test the quality of the underlying TCP connection and to make sure that the remote end is available. A recommended default value for the heart-beat intervals is 10 seconds in each direction: heartbeat: 10000,10000. The heart-beat interval can be adjusted according to the expected use of the connection. If the incoming push notification rate is high, the heart-beat interval can be increased, as incoming STOMP messages and the corresponding ACKs sent indicate the health of the connection. Setting very low heart-beat intervals is not recommended, as it can create unnecessary load.

Example of CONNECT frame send by the client:

```
Opening Web Socket...
Web Socked Opened...
>>> CONNECT
accept-version:1.2
Heart-beat: 10000,10000
login: demo.user9@ericsson.com
passcode: ************
<<< CONNECTED
server: ActiveMQ/5.11.0.redhat-630283
heart-beat: 10000,10000
session: ID: broker-amq-4-h5nkz-46706-1548860591533-2:603
version: 1.2
connected to server ActiveMQ/5.11.0.redhat-630283
send PING every 10000ms
check PONG every 10000ms
connected to STOMP
```

3.2 Subscribe to the Message Queues

To subscribe to message queues:

1. Subscribe to push notifications by sending a SUBSCRIBE frame.
2. Set the following header to indicate which message queue to subscribe to: **destination**:<the name of the queue>.
3. Since STOMP allows a single connection to have multiple open subscriptions with a server, include an ID header in the frame to uniquely identify the subscription:
   id:<a unique subscription-id for the used connection>.
4. Set the **ack** header to set the message acknowledgment mode.

Note: If the server cannot successfully create the subscription, it responds with an ERROR frame and closes the connection.

The recommended ack mode is client individual: ack:client-individual.

This acknowledgment mode requires the client to ACK or NACK each message sent from the message broker. The message broker does not consider a message delivered until an ACK or NACK STOMP frame acknowledging the message has been received. No messages are removed from the broker message queue until successfully delivered. A positive acknowledgement using an ACK STOMP frame indicates that the message was successfully delivered and processed. A negative acknowledgement using a NACK STOMP frame indicates that the message was delivered but was not processed successfully. In both cases, the broker considers the message delivered and removes it from the message queue.
An un-acknowledged message is redelivered the next time a consumer connects to the broker and subscribes to the queue.

SUBSCRIBE Frame Format:

SUBSCRIBE
id:<subscription-id>
destination:<queue_name>
ack:client-individual

Examples of SUBSCRIBE frames are provided in the use cases described below in the document.

3.3 Unsubscribe from Message Queues

To unsubscribe from a message queue:

1. Unsubscribe from the push notifications of a given destination by sending an UNSUBSCRIBE frame.
2. Set the following header to indicate which message queue to unsubscribe from: **id**:<the subscription-id used in the SUBSCRIBE frame>.

UNSUBSCRIBE Frame Format:

UNSUBSCRIBE
id:<subscription-id>

3.4 Disconnect from the Message Broker

A client can disconnect from the server at any time by closing the socket.
Note: By using this method, it is possible that the previously sent frames are not received by the server.

To receive acknowledgement that all previous frames are received by the server, send a DISCONNECT frame with a receipt header set. Wait for the RECEIPT frame response to the DISCONNECT before closing the socket.

DISCONNECT Frame Format:

DISCONNECT
receipt:<arbitrary value>

4 Use Cases

Hereby is described how to trigger three categories of events and how to subscribe to Service Portal specific STOMP notifications:

- Trigger events notifications
- SMS push messaging notifications
- Subscription request notifications

4.1 Trigger Event Notifications

The Service Portal supports sending notifications for all trigger types. Notifications are sent with any other actions or as the only action of a trigger. A notification can contain a list of subscriptions. Sending these notifications must be configured for each trigger group separately.

4.1.1 How to trigger STOMP notifications:

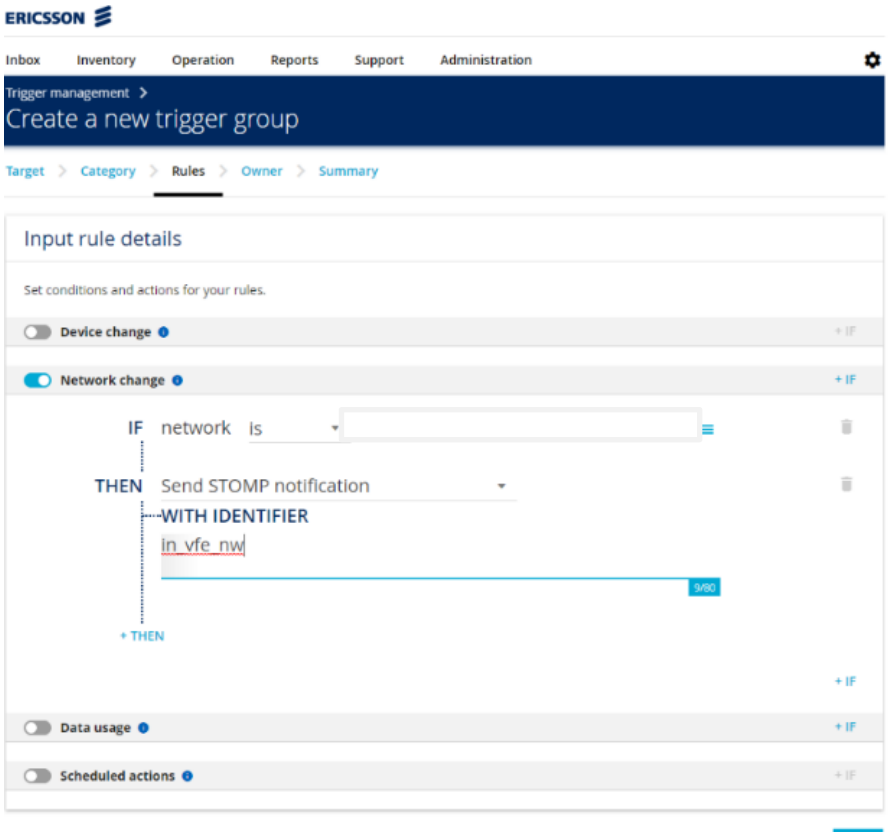
Pre-requisites:

"**STOMP notification for trigger action**" permission needed for the user onboarded in Service Portal. If you don´t have this permission, please ask the activation to your Connectivity Service Provider.

A new trigger group should be created through the menu *Operation* > *Automation* > *Trigger management*, by selecting a target enterprise or a subscription package, defining a category with a rule that has as condition to send a STOMP notification.

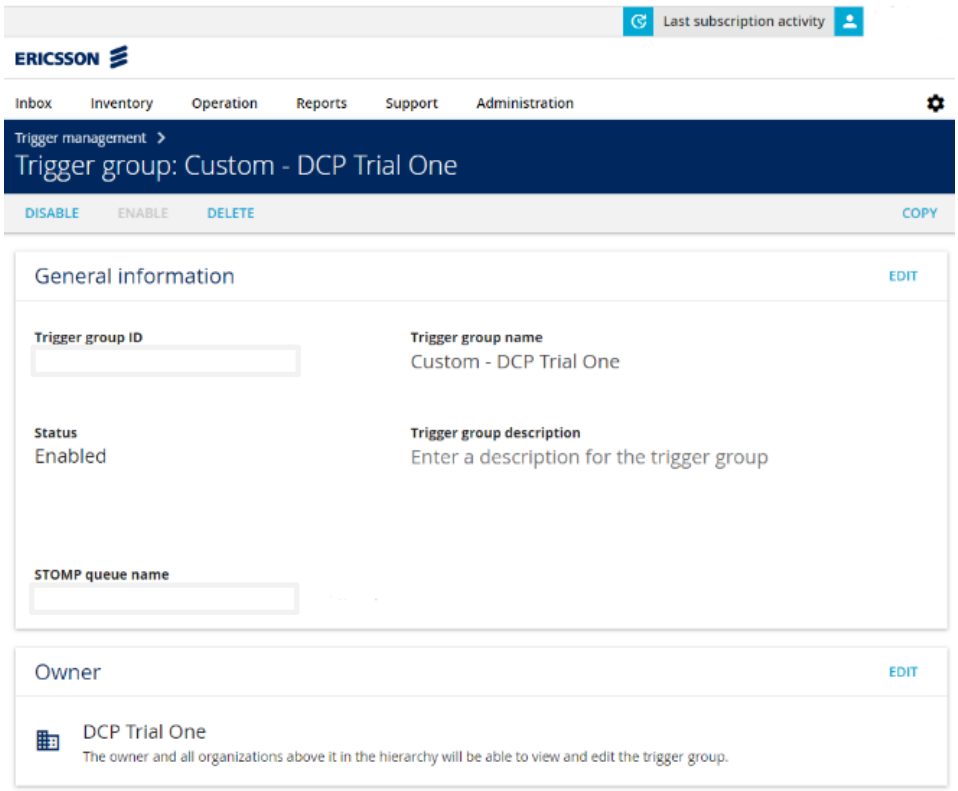
More details about how to create triggers can be found at Trigger Event Tutorial in Developer Portal.

Example of Creation Trigger group in Service Portal:



A STOMP queue name is provided at the end of the process of creating the Trigger Group. This queue can be used by an authorized STOMP client to subscribe to it.
To view the STOMP queue name, select the created trigger group from the list available in the menu *Operation* > *Automation* > *Trigger management* > *General Information section*.

Example of General Information for Trigger group in Service Portal:



4.1.2 How to subscribe to STOMP notifications:

Pre-requisites:

- "**Push API consumer for trigger event service**" permission granted for the onboarded user in Service Portal.
- The STOMP client is successfully connected to the STOMP server as described in the section "Connect to the Message Broker".

The client should subscribe to the STOMP queue for trigger notifications by sending the SUBSCRIBE frame specifying the headers:

- id: <unique identifier of the subscription>
- destination: dcp.<organizationId>.trigger.v1.0.0
  (The queue name for trigger notifications as specified at the trigger management summary page for trigger groups that have the Send STOMP notification action configured.)
- ack: client-individual

Example of SUBSCRIBE frame send by the client:

```
>>> SUBSCRIBE
id: sub-0
destination: dcp.0.6.XXX.XXX.trigger.v1.0.0
ack:client-individual
<<< PONG
>>> PING
<<< PONG
>>> PING
```

4.1.3 Receive Trigger Event Notifications

When the condition of the rule defined in the Trigger group is fulfilled, a STOMP notification is pushed in a MESSAGE frame.

MESSAGE Frame Format:

MESSAGE
destination:<queue_name>
message-id:<message-id>
subscription:<subscription-id>
Content_Type:application/json
{
<payload>
}

The STOMP notifications are created on a best-effort basis. If a STOMP notification fails to send, it cannot be resent. To prevent old non-consumed messages from filling up the queues, messages are posted with a time-to-live set.

For trigger notifications, the value is one hour. If the message is not consumed within one hour, it is permanently deleted.

Example of MESSAGE frame received by the client:

```
<<< MESSAGE
expires: 1549016005420
destination: /queue/ dcp.0.6.216.270.trigger.v1.0.0
subscription: sub-0
priority:4
Content_Type: application/json
message-id: ID: cen0-exs01-39520-15482
persistent: true
timestamp: 1549016005420
{
   "time_stamp":"2020-09-15T13:09:53.867Z",
   "trigger_group_id":"17491dfbcfae7",
   "trigger_group_name":"CUSTOM- DCP Trial One",
   "custom_identifier":"in_vfe_nw",
   "subscriptions":[
   {
        "imsi":"238xxxxxxxxx",
```

```
            "msisdn":"453xxxxxxxxx",
            "iccid":"8xxxxxxxxxxxxxxxxxx",
            "current_imei":"",
            "previous_imei":"xxxxxxxxxxxxxxxxx",
            "subscription_package_id":"xxxxxxxxxxxxxxxxxxxx",
            "subscription_package_description":"xxxxxxxxxxxxx",
            "enterprise id":"06xxxxx",
            "organization_id": "*.**.*",
            "mcc": "***",
            "mnc": "***"
         }
      ],
      "actions":[
         {
            "action_type":"STOMP_NOTIFICATION"
         }
      ],
      "condition":{
         "condition_type":"networkAgreement",
         "networks":[
            {
               "mcc":"262",
               "mncs":"02"
            }
         ],
         "type":"IS"
      }
   }
```


4.2 SMS Messaging Push Notifications

The Service Portal is enabling SMS messaging between enterprise web applications and cellular devices using Short Message Peer to Peer Protocol (SMPP), so that for every outgoing and incoming SMS message of an enterprise application, a correspondent STOMP notification is pushed towards the STOMP client.

The SMS Messaging Push Notification API supports real-time delivery of inbound Application Terminated (AT) SMS messages, that are Mobile Originated (MO) SMS messages sent to an enterprise application. It also supports real-time delivery of delivery receipts sent back notifying the sender of the final delivery status of a Mobile Terminated (MT) SMS.

Pre-requisites:
To access the SMS Messaging Push Notification Interface, an SMS Messaging Service account and associated Service Portal account are needed. The same portal user must be used to send SMS REST API requests and to connect to the SMS Messaging Push Notification Interface by using STOMP.
This portal user must have been given the "**Push API consumer for SMS service"** permission in the Service Portal.

Operator Service Desk for Service Portal administration should be contacted to support for the needed accounts and permissions.

As described in the Service Portal Operator Guide, the SMS Messaging Service account can be provisioned during the enterprise onboarding process by filling in the Resource Setup Template section "SMS Messaging API Setup".

The SMS Messaging Service account has the following security tokens associated:

- username (from the Service Portal user account)
- password (from the Service Portal user account)
- applicationTerminatedSmsQueueName
- smsDeliveryReceiptsQueueName

The applicationTerminatedSmsQueueName token uses the following convention:

dcp.<organizationId>.sms.<version>.inbound.<registrationId>.

The smsDeliveryReceiptsQueueName token uses the following convention:

dcp.<organizationId>.sms.<version>.receipts.<registrationId>.

The organizationId (also companyId) identifies the organizational unit of the queue.

The <version> and <registrationId> values are the same values as configured for the SMS Messaging Service account.
The tokens are given to the enterprise user as part of the onboarding process.
The system needs these tokens to authenticate and authorize the requests over the SMS Messaging Push Notification Interface.

The SMS Messaging Service account used in this tutorial is related to an enterprise application and it has the following parameters values configured:

```
- smsAccountId: tel:31013
- dcpUserName: demo.user9@ericsson.com
- companyId: 0000001
- smsRetrieveType: sms_push
- senderAddress: tel:31013
- registrationId: 31013
- pushMessageTimeToLive: 60
- pushApiVersion: v1
```

4.2.1 Push Notifications for Delivery Receipts:

The Service Portal provides SMS Messaging APIs that allows enterprise web applications to send and receive SMS messages. The SMS Messaging API interface is using HTTS protocol to simplify the communication between the enterprise and the DCP SMSC.

The enterprise application is sending an outgoing SMS to a cellular device (eq. mobile telephone) by using the corresponding SMS rest API. When the message is delivered to the destination, a STOMP notification is pushed to the client that subscribed to receive delivery receipts for the sent SMSs .

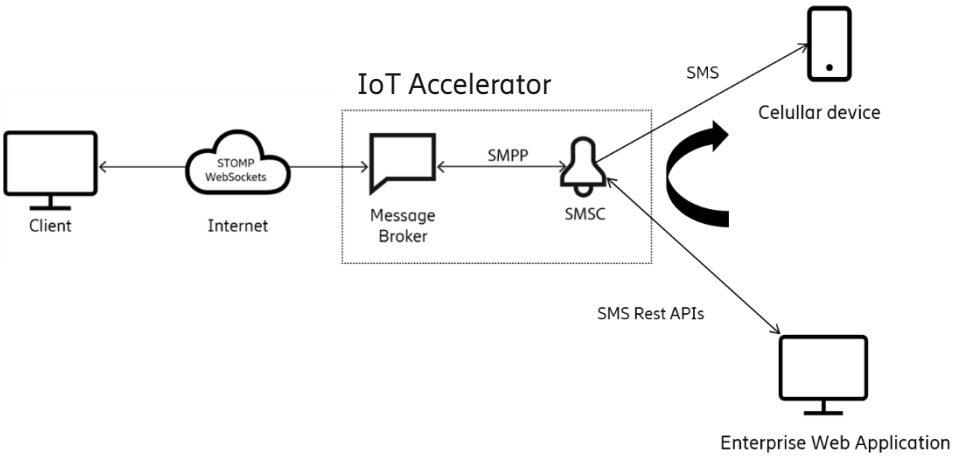


Figure 2 Logical Architecture of the Messaging Service for Delivery Receipts

4.2.1.1 How to subscribe to STOMP notifications

After the STOMP client is successfully connected to the STOMP server as described in the section "Connect to the Message Broker", the client should subscribe to the STOMP queue for delivery receipts by sending the SUBSCRIBE frame specifying the headers:

- id: <unique identifier of the subscription>
- destination:
  dcp.<organizationId>.sms.<version>.receipts.<registrationId>
  the queue name for SMS delivery receipts where:
  - <organizationId> is the companyId
  - <version>
  - <registrationId>

  have the values as configured for SMS Messaging Service account.

- ack: client-individual

Example of SUBSCRIBE frame send by the client:

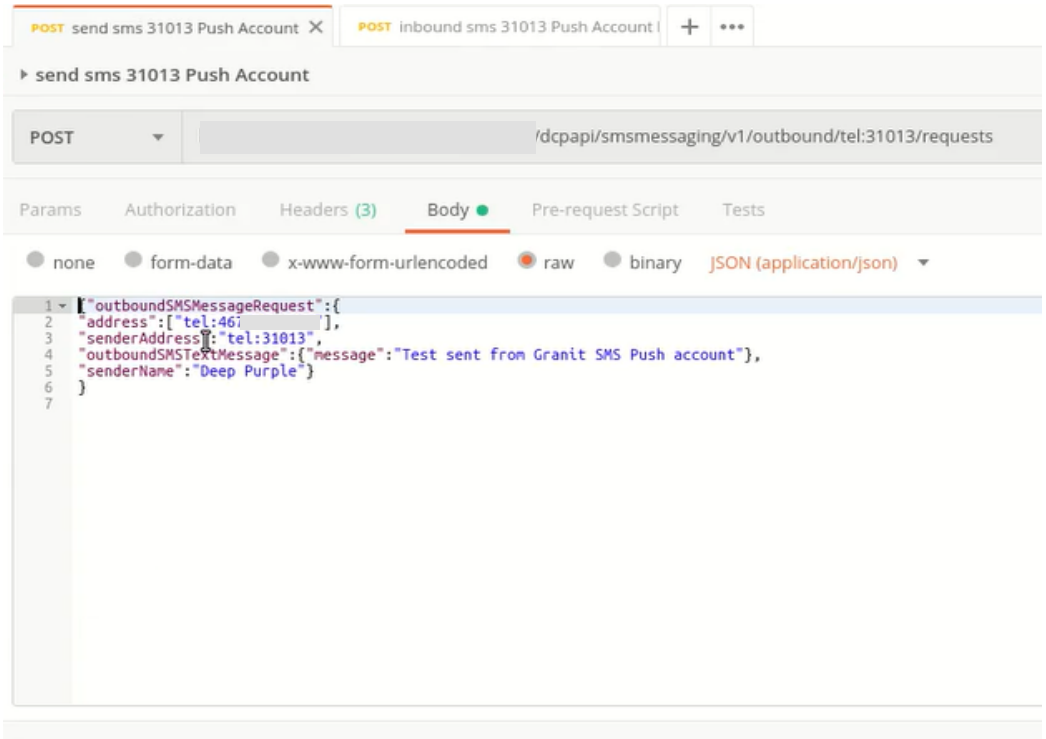
```
>>> SUBSCRIBE
id: sub-1
destination: dcp.00000001.sms.v1.receipts.31013
ack:client-individual

<<< PONG
>>> PING
<<< PONG
>>> PING
```

4.2.1.2 How to trigger STOMP notifications for outgoing SMSs

After the STOMP client successfully subscribed to the message queue, an outgoing SMS is sent by using the corresponding Service Portal SMS Messaging REST API specifying:

- the address to send the SMS
- the sender address
- the sender name
- the message text to be sent

Example of Service Portal API for sending SMS:



4.2.1.3 Receive Push Notifications for outgoing SMSs:

For each MT SMS sent using the SMS Messaging Service REST API, a notification is pushed to the client.

The notification is received by the client in a STOMP MESSAGE frame and it contains delivery information for the SMS that was sent:

- requestId identifying the request
- the address that the SMS was sent to
- the delivery status which is DeliveredToTerminal

STOMP MESSAGE frame format for SMS delivery:

```
MESSAGE
      destination: <queue_name>
      message-id: <message-id>
      subscription: <subscription-id>
      Content_Type: application/json
      {
        deliveryInfo: {
          requestId: <requestId>,
          address: <address>,
          deliveryStatus: <deliveryStatus>,
          detailedDeliveryStatus: {
            <detailedDeliveryStatus>
          }
        }
      }
```

The client should ACK or NACK the MESSAGE frame received including the message-id of the MESSAGE frame received.

Example of MESSAGE frame received by the client:

```
>>> PING

<<< MESSAGE
expires:1549019618420
destination: /queue/dcp.00000001.sms.v1.receipts.31013
subscription: sub-1
priority:4
Content_Type: application/json
message-id: ID: cen0-exs01-39520-15482
persistent: true
timestamp: 1549016018420

{“deliveryInfo”:{“requestId”:”5fa362f2-350f-4dba-8737-af94f18b655”,
 address”:”tel:46xxxxxxxxx,”deliveryStatus”:”DeliveredToTerminal”}}

 >>> PING
 <<<PONG


 >>> ACK
 message-id: ID: cen0-exs01-39520-15482
```

4.2.2 Push Notifications for Inbound AT SMS Message:

When an application receives an incoming SMS, a STOMP notification is pushed to the client that subscribed for the inbound AT SMS messages. The logical flow is similar with the one illustrated in the figure 2 for delivery receipts, but in this case the application is the receiver of the SMSs.

4.2.2.1 How to subscribe to STOMP notifications

After the STOMP client is successfully connected to the STOMP server as described in the section “Connect to the Message Broker”, the STOMP client should subscribe to the STOMP queue for incoming SMS by sending the SUBSCRIBE frame specifying the headers:

- id: <unique identifier of the subscription>

- destination:
  dcp.<organizationId>.sms.<version>.inbound.<registrationId>

  the queue name for Inbound AT SMS where:

  — <organizationId> is the companyId
  — <version>
  — <registrationId>

  have the values as configured for SMS Messaging Service account.

- ack: client-individual

Example of SUBSCRIBE frame send by the client:

```
>>>> SUBSCRIBE
id: sub-2
destination: dcp.00000001.sms.v1.inbound.31013
ack:client-individual

<<< PONG
>>> PING
<<< PONG
>>> PING
```

4.2.2.2 How to trigger STOMP notifications for incoming SMSs

After the STOMP client successfully subscribed to the message queue, a SMS is sent to the enterprise application by using the corresponding Service Portal SMS Messaging REST API specifying:

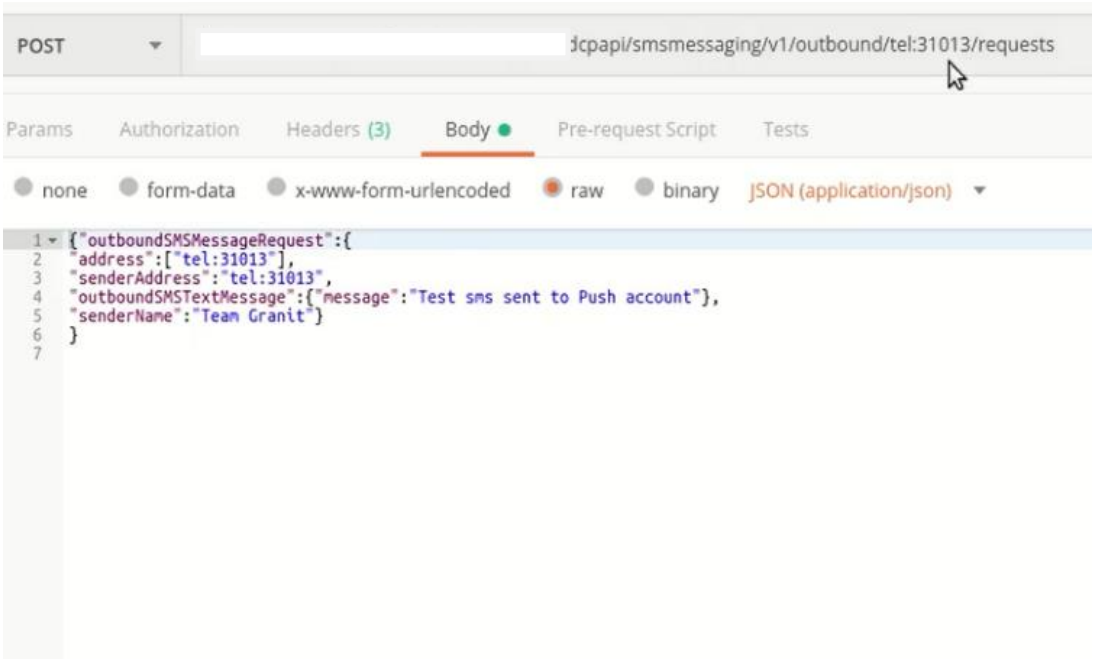
- the address to send the SMS
- the sender address
- the sender name
- the message text to be sent

which in this case is actually looped back to the same SMS sender address

or

by sending a SMS message to the SMS account (eq: 31013 from a device that uses a Service Portal SIM.

Example of Service Portal API for sending SMS:



4.2.2.3 Receive Push Notifications for incoming SMSs

For each inbound AT SMS received by the DCP SMSC, a STOMP MESSAGE frame containing the MT SMS delivery status information is pushed out.

MESSAGE frame format for application terminated SMS:

```
MESSAGE
    destination: <queue_name>
    message-id: <message-id>
    subscription: <subscription-id>
    Content_Type: application/json
    {
      dateTime: <dateTime>,
      destinationAddress: <destinationAddress>,
      messageId: <smsMessageId>,
      message: <message>,
      senderAddress: <senderAddress>,
      messageType: <messageType>
    }
```

The following two parameters carry message IDs:

- **message-id header:** A STOMP header containing a unique identifier for that STOMP MESSAGE set by the message broker.
- **messageId:** A JSON body parameter containing a unique identifier for the AT SMS.

Each MESSAGE frame successfully received by the client must be acknowledged by sending an ACK frame containing an ID header matching the message-id header of the MESSAGE frame.

Example of MESSAGE frame received by the client:

```
>>> PING
<<< MESSAGE
expires:1549019725467
destination: /queue/dcp.00000001.sms.v1.inbound.31013
subscription: sub-2
priority:4
Content_Type: application/json
message-id: ID: cen0-exs01-39520-15487
persistent: true
timestamp: 1549019725467


{“datetime”:190201101525”,,”destinationAddress”:”31013”,”messageId”:”dcpMsg111
31,”message”:”Test SMS sent to Push account”,”senderAddress”:”31013”}

>>> PING
<<< PONG


>>> ACK
message-id: ID: cen0-exs01-39520-15487
```

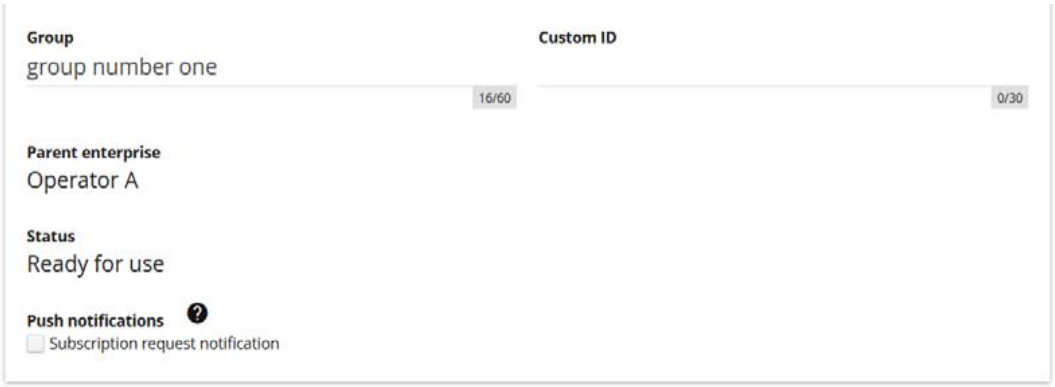
4.3 Subscription Request Notifications

As a third group of events that may generate STOMP notifications, Service Portal supports sending of STOMP notifications about the completion of the following requests:

— eUICC Localization
— eUICC Batch Localization
— eUICC Delocalization
— eUICC Batch Delocalization
— eUICC Add Locale
— eUICC Batch Add Locale
— eUICC Remove Locale
— Change Subscription State (CSS)
— Change Subscription Package (CSP)

STOMP messages are used to notify the user about activities related to eUICCs. Sending these notifications must be configured for each enterprise separately. Users gain access to the notifications by subscribing to their company queue in Service Portal ActiveMQ broker through STOMP.

To enable the queue, select the **Subscription request notification** checkbox in the **Enterprise Details** page of Service Portal:



4.3.1 How to subscribe to Subscription Request STOMP notifications

Pre-requisites:

- "**Push API consumer for Service requests**" permission is granted for the onboarded user in Service Portal.
- The STOMP client is successfully connected to the STOMP server as described in the section "Connect to the Message Broker".

The STOMP client should subscribe to the STOMP queue for Subscription Request notifications by sending the SUBSCRIBE frame with the headers:

- id: <unique identifier of the subscription>
- destination: dcp.<companyId>.service_requests_2.v1
  (The queue name for Subscription Request Notifications.)
- ack: client-individual

Example of SUBSCRIBE frame send by the client:

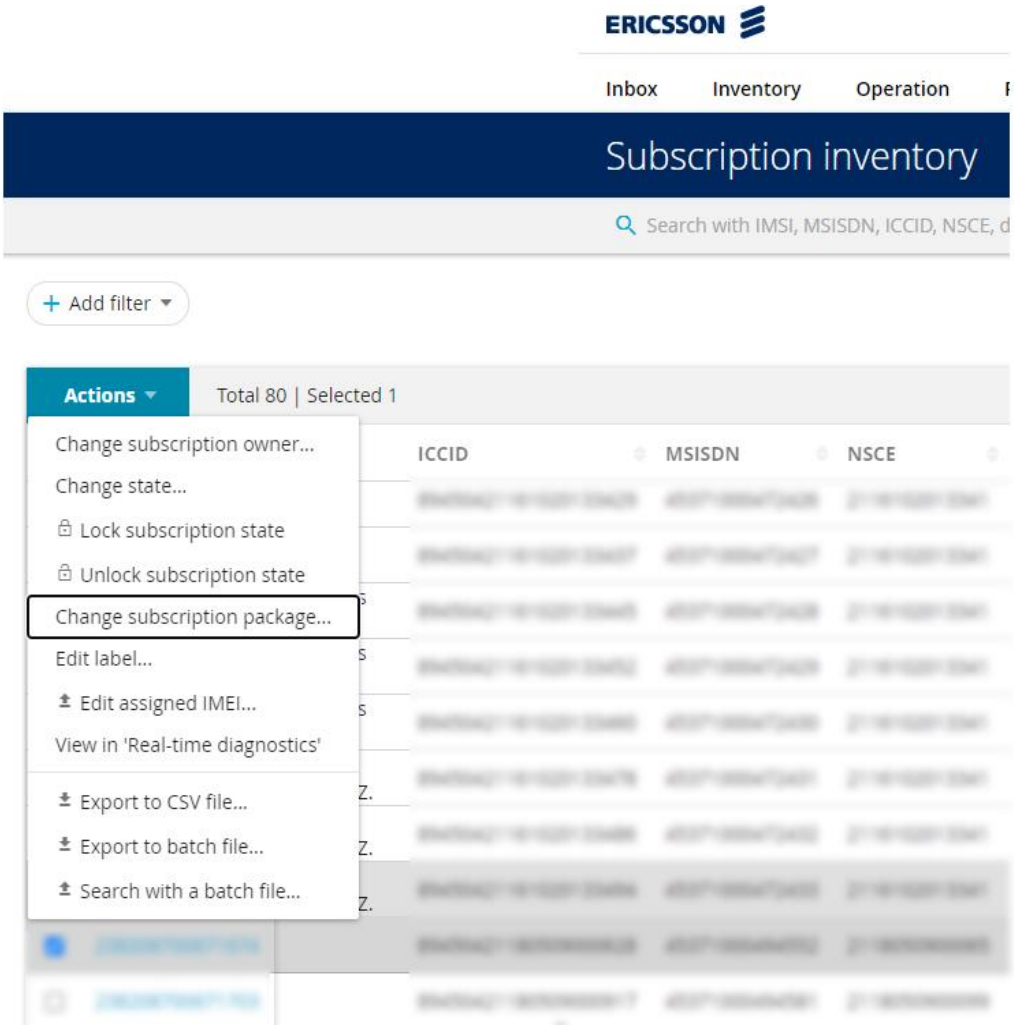
```
>>> SUBSCRIBE
id: sub-3
destination: dcp.00000001.service_requests_2.v1
ack:client-individual
<<< PONG
>>> PING
<<< PONG
>>> PING
```

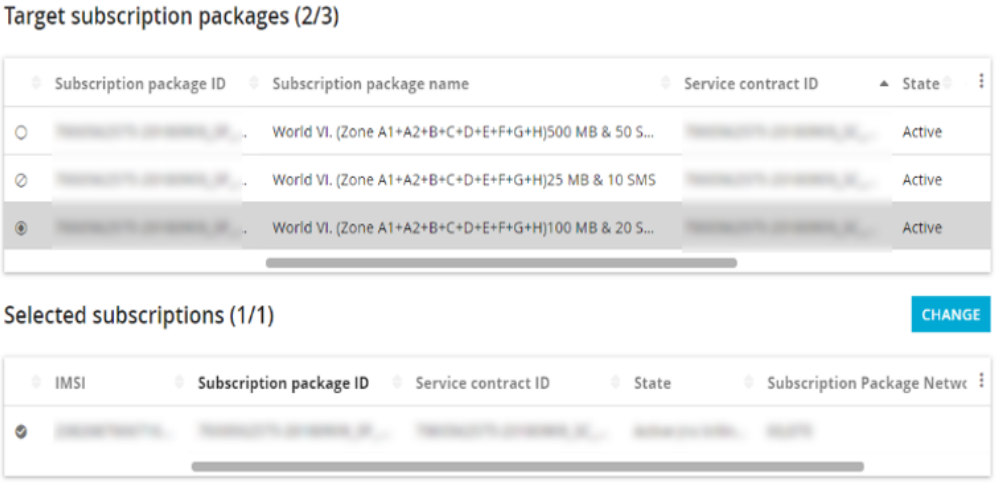
4.3.2 How to trigger STOMP notifications for Subscription Requests:

Here it is taken as example the Change Subscription Package Request.

(For the other above-mentioned subscriptions requests, push notifications can be triggered by using Service Portal GUI or using the corresponding APIs, in a similar manner.)
Request the Change of the subscription package by using the Service Portal GUI:

1. Go to Inventory > Subscriptions > Subscription Inventory
2. Select the subscription checkbox
3. From Actions menu select Change subscription package
4. Select the Target subscription package
5. Press button CHANGE





4.3.3 Receive Push Notifications for Subscription Requests:

When the subscription package is changed, a STOMP notification is pushed in a MESSAGE frame.

MESSAGE Frame Format:

```
MESSAGE

  destination:<queue_name>
  message-id:<message-id>
  subscription:<subscription-id>
  Content_Type:application/json
  {
    <payload>"
  }
```

The payload content of the notification is an extension of the content in a GET /serviceRequests/{id} query through the Device Localization API of the Service Portal. For more information on /serviceRequests/{id} queries, refer to the document Device Localization API.

Example of MESSAGE frame received by the client for subscription package changed:

```
<<< MESSAGE
expires: 1549018065310
destination: /queue/dcp.00000001.service_requests_2.v1
subscription: sub-3
priority:4
Content_Type: application/json
message-id: ID: cen0-exs01-39520-15482
persistent: true
timestamp: 1549018065310

{
   "subscriptionRequestId":"REQxxxxxxxxxxx",
   "subscriptionRequestType":"CHANGE_SUBSCRIPTIONPACKAGE",
   "subscriptionRequestState":"COMPLETED",
   "createdBy":"tester@mycompany.com",
   "companyId":"12345678",
   "companyName":"My_company",
   "timeCreated":{
      "epochSecond":1558761529,
      "nano":400000000
   },
   "lastUpdated":{
      "epochSecond":1558761530,
      "nano":161000000
   },
   "size":1,
   "subscriptionRequestResourceCounters":{
      "inProgress":0,
      "pending":0,
      "canceled":0,
      "rejected":0,
      "completed":1,
      "failed":0
   },
   "subscriptionRequestResources":[
      {
         "resourceProperties":[
            {
               "type":"IMSI",
               "value":"xxxxxxxxxxx8",
```

```
                "operation":"SUBSCRIPTION PACKAGE CHANGE"
             },
             {
                "type":"MSISDN",
                "value":"xxxxxxxxxxxxxxx8",
                "operation":"SUBSCRIPTION_PACKAGE_CHANGE"
             },
             {
                "type":"ICCID",
                "value":"xxxxxxxxxxxxxxxxxxx8",
                "operation":"SUBSCRIPTION_PACKAGE_CHANGE"
             },
             {
                "type":"TARGET_SUBSCRIPTION_PACKAGE",
                "value":"xxxxxxxxxxxxx_24",
                "operation":"SUBSCRIPTION_PACKAGE_CHANGE"
             }
          ],
          "resourceState":"COMPLETED",
          "resourceId":"INT201004416492"
       }
    ]
}
```


For more details about the STOMP procedures and headers see STOMP Protocol Specification, Version 1.2.