

# Configuration Guide for End User Notifications

Ericsson Service-Aware Policy Controller

USER GUIDE

**Copyright**

© Ericsson España, S.A. 2017. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

**Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

**Trademark List**

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.

**Abstract**

This document is a guideline to configure the SAPC for End User Notifications.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Document Purpose and Scope	1
1.2	Revision Information	2
1.3	Typographic Conventions	2
1.4	Other Conventions	3
<b>2</b>	<b>Configuration Prerequisites</b>	<b>5</b>
<b>3</b>	<b>Configure Notification Network Data</b>	<b>7</b>
3.1	Configure Notifications Delivery	9
3.2	Configure Notification Servers	10
3.2.1	Configure SMS Centers	10
3.2.2	Configure Web Service End Points	11
3.3	Configure Notification Destinations	13
3.3.1	Notifications Sent to End Users	13
3.3.2	Notifications Sent to External Systems	14
<b>4</b>	<b>Provisioning Notification Policies</b>	<b>15</b>
4.1	Notifications Sent to End Users	16
4.2	Notifications Sent to External Systems	16
4.3	Notification Sent to End Users and to External Systems	23
4.4	Dynamic Notification Message Content	24
4.5	Encoding Schemes in Notification Messages	25
<b>5</b>	<b>Configuration Examples for Use Cases</b>	<b>27</b>
5.1	Notifications Based on Fair Usage	27
5.1.1	Notifications to be sent to Subscribers	27
5.1.2	Notifications and Shared Dataplan	28
5.2	Notifications Based on Subscriber Group Activation or Deactivation	33
<b>6</b>	<b>Appendix A. End User Notifications Policy Types</b>	<b>37</b>
	<b>Glossary</b>	<b>39</b>





# 1 Introduction

## 1.1 Document Purpose and Scope

Next figure, shows the main parts related to configuration and provisioning in the the SAPC.

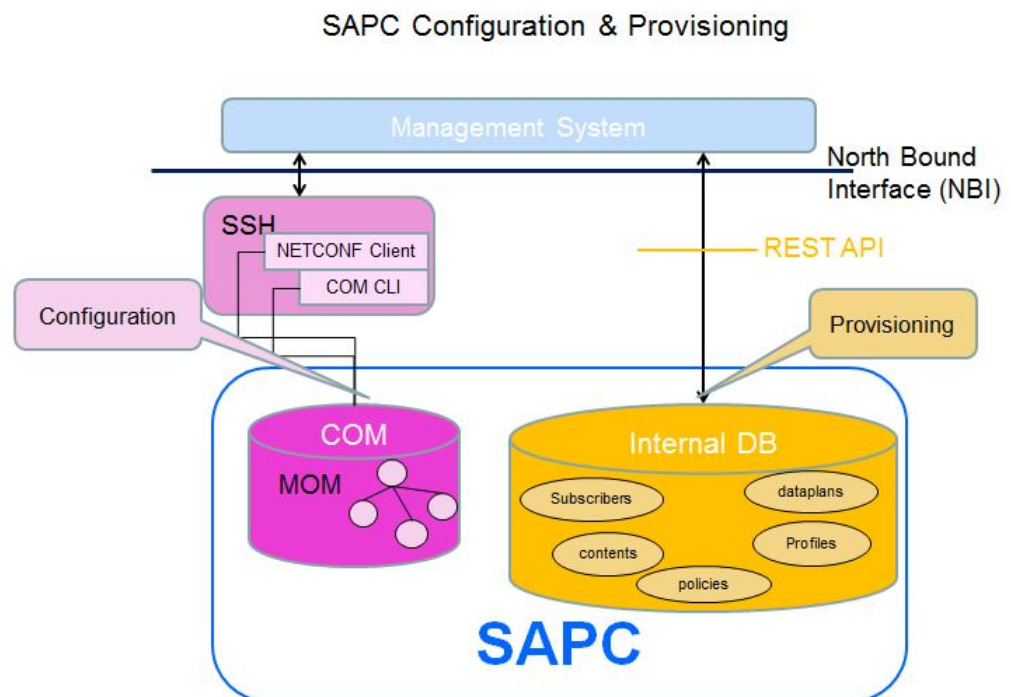


Figure 1 Configuration and Provisioning Overview

The purpose of this document is to provide a guideline for configuring End User Notifications in the SAPC by providing some configuration examples.

This document does not intend to be an exhaustive guide for configuring all possibilities related to End User Notifications in the SAPC.

The complete parameter list and details of all configuration options of the SAPC are included in separate documents, refer to *Managed Object Model (MOM)* and *Provisioning REST API*.

General concepts on how to provision Subscribers, Subscriber Groups and how to configure policies are covered in *Configuration Guide for Subscription and Policies*.

Examples on this document cover the case of data configured in the SAPC internal repository. In case an external repository is used, refer to *Database Access*.



## 1.2 Revision Information

**Rev. A** This is the first release of this document.

## 1.3 Typographic Conventions

The following typographic and document conventions are used:

Table 1 Typographic Conventions

Convention	Description	Example
<b>Representational State Transfer (REST)</b>	SAPC REST provisioning.  Exact REST resources, methods, attributes, or their corresponding values.	<pre>PUT /dataplan/Silver  {   "dataplanName" : "Silver",   "notification" : "sms" }</pre>



Table 1 Typographic Conventions

Convention	Description	Example
<b>Managed Object Class (MOC)</b> or <b>Attributes value</b>	Exact COM model object, classes names, attributes, or their corresponding values.	SmsCenter  enableDelivery=true
<b>NETCONF</b>	SAPC COM configuration	<pre> &lt;edit-config&gt;   &lt;target&gt;     &lt;running /&gt;   &lt;/target&gt;   &lt;config&gt;     &lt;ManagedElement       xmlns="urn:com:ericsson:ecim:ComTop"&gt;       &lt;managedElementId&gt;         1       &lt;/managedElementId&gt;       &lt;PolicyControlFunction         xmlns="urn:com:ericsson:ecim:sapcmom"&gt;         &lt;policyControlFunctionId&gt;           1         &lt;/policyControlFunctionId&gt;         &lt;NotificationConfig           xmlns="urn:com:ericsson:ecim:notificationconfigmom"&gt;           &lt;notificationConfigId&gt;             1           &lt;/notificationConfigId&gt;           &lt;enableDelivery&gt;             true           &lt;/enableDelivery&gt;         &lt;/NotificationConfig&gt;       &lt;/PolicyControlFunction&gt;     &lt;/ManagedElement&gt;   &lt;/config&gt; &lt;/edit-config&gt; </pre>

## 1.4 Other Conventions

This document refers to some configuration and provisioning data.

To clarify which detailed data is managed by COM or by the REST API, this document uses the following conventions:

- Configuration: whenever referring to Managed Object Class (MOC).



The detailed description for the object and attributes can be found in [Managed Object Model \(MOM\)](#).

Example: set `enableReauthsOnSubsChange` attribute in class `AppConfig`.

The tools or interfaces to manage these data in the SAPC are:

- a NETCONF interface, refer to [Ericsson NETCONF Interface](#).

The configuration examples show the NETCONF file contents, using the following syntax:

```
<edit-config>
...
<config>
  <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
    <managedElementId>1</managedElementId>
    ...
  </ManagedElement>
</config>
</edit-config>
```

- b Or COM CLI, refer to [Ericsson Command-Line Interface](#).

- Provisioning: mainly subscribers, subscriber groups (dataplan), services (contents), profiles, and policy-related data. The SAPC provides a REST API for them, see [Provisioning REST API](#).

This document uses the following terminology for them: `<resource-name>` URI in the provisioning REST API.

Example: To provision subscriber groups, use the `dataplan` URI in the provisioning REST API.

And provisioning examples show HTTP operations on REST resources with the following syntax:

HTTP-Operation /resource-URI  
 {json content} where /resource-URI is the relative URI from the SAPC provisioning base URI detailed in [Provisioning REST API](#).

Example:

```
PUT /dataplan/Gold
{ "dataplanName" : "Gold",
  "subscribedContents" : [{"contentName" : "HTTP_Streaming",
                           "redirect" : false}]
}
```

**Note:** To ease provisioning operations, the SAPC provides an HTTPS CLI client named `resty`, refer to [Provisioning Tools](#).





## 2 Configuration Prerequisites

Before configuring the SAPC in an operational network, assure that:

- CBA Components are installed.
- The SAPC product software is installed.
- To have a detailed understanding of the function.





## 3 Configure Notification Network Data

In End User Notifications, it is possible to differentiate the following types of notifications:

- Notifications sent to end users whose final destination are subscribers. The notification mechanism for this type is SMS using SMPP protocol.
- Notifications sent to External Systems whose final destination is an external node. The notification mechanisms for this type are SMS and SOAP.

To make the SAPC possible to send notifications, it is needed to:

- 1 Enable notifications to be delivered, see Section 3.1 on page 9
- 2 Configure the notification servers where the SAPC sends the notifications. The notification server is responsible for sending the notifications to the end user or to the external system. See Section 3.2 on page 10
- 3 Configure the subscriber notification destination that includes mechanism or address. The specific configuration depends on the type of notification: end user or External System, see Section 3.3 on page 13
- 4 Specify the business condition that triggers the notification, by provision notification policies. Specify the content of the notification message, and associated mechanism and destination. Section 4 on page 15

**Note:** Some specific details of the notification policies are configured in different ways depending on the notification type: end user or External Systems.

Next figure shows the main elements to configure to send notifications to End Users:

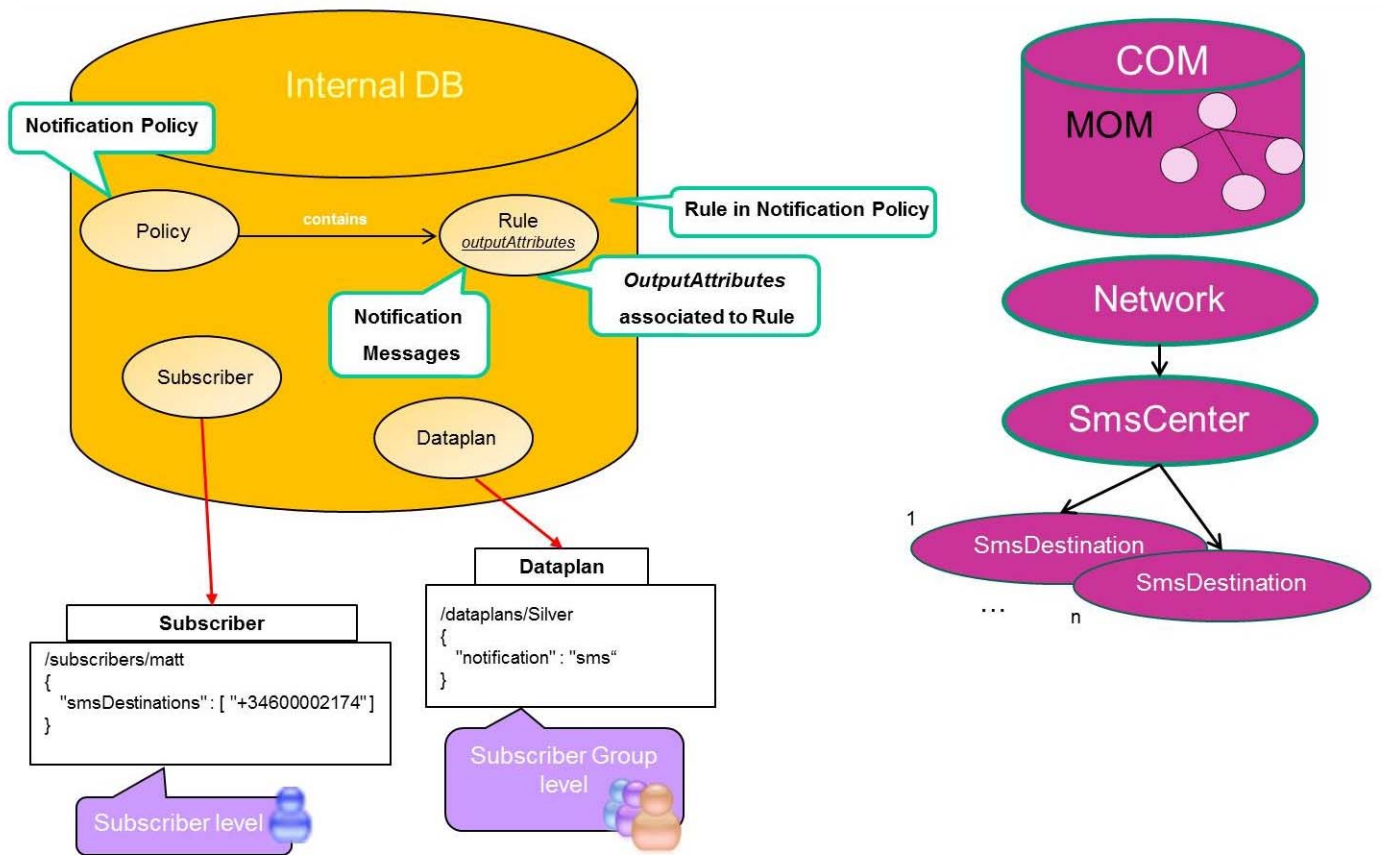


Figure 2 Notifications sent to End Users

Next figure shows the main elements to configure to send Notifications to External Systems:

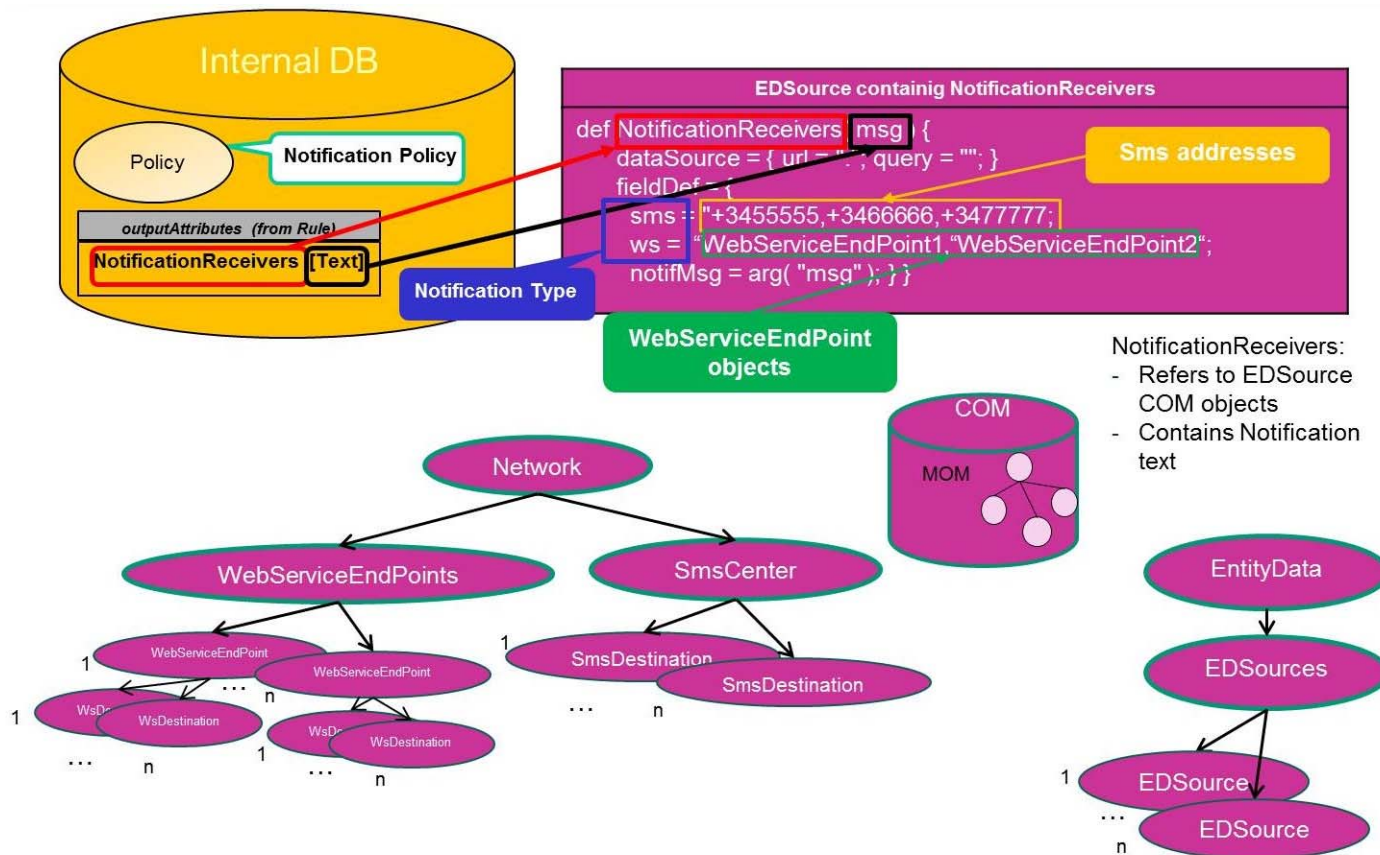


Figure 3 Notifications sent to External Systems

Following sections contain the configuration details in a progressive way.

### 3.1 Configure Notifications Delivery

Notifications can be enabled or disabled using **NotificationConfig** COM object.

- To enable notifications to be sent, set `enableDelivery` attribute to true.
- To disable notifications, set `enableDelivery` attribute to false.

Next, the example to enable notifications:



```
<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <NotificationConfig xmlns="urn:com:ericsson:ecim:notificationconfigmom">
          <notificationConfigId>1</notificationConfigId>
          <enableDelivery>true</enableDelivery>
        </NotificationConfig>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>
```

Example 1 Configuration of delivery notifications

## 3.2 Configure Notification Servers

---

---

### Attention!

To adapt the SAPC depending on the behavior of the notification servers (for example number of connections, notification response time including network delay), contact Ericsson personnel.

---

---

In this section, it is explained how to configure the notification servers towards the SAPC send notifications.

### 3.2.1 Configure SMS Centers

In the SAPC, it is possible to configure only one SmsCenter **COM object** and several children SmsDestination **COM object**. Then, the SAPC sends each SMS message, towards a different SMS destination, selecting the SMS destination using a round robin algorithm.

Next, an example on how an SmsCenter with one SmsDestination is configured:



```

<edit-config>
  <target>
    <running/>
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <Network xmlns="urn:com:ericsson:ecim:networkmom">
          <networkId>1</networkId>
          <SmsCenter xmlns="urn:com:ericsson:ecim:smscentermom">
            <smsCenterId>1</smsCenterId>
            <SmsDestination xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
              nc:operation="merge">
              <smsDestinationId>159.107.22.33</smsDestinationId>
              <serverPort>2775</serverPort>
              <serverUser>SAPC</serverUser>
              <serverPwd>
                <cleartext></cleartext>
                <password>Password</password>
              </serverPwd>
              <serverTonDestination>3</serverTonDestination>
              <serverTonOrigin>3</serverTonOrigin>
              <serverNpiOrigin>14</serverNpiOrigin>
              <serverNpiDestination>14</serverNpiDestination>
            </SmsDestination>
          </SmsCenter>
        </Network>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>

```

### Example 2 Configuration of one SmsDestination

The previous example configures a single `SmsDestination` child of the `SmsCenter` “1”, whose IP address is “159.107.22.33”, listening to SMPP protocol on 2775 port. The connection to this server is done using user “SAPC” and password “Password”.

## 3.2.2 Configure Web Service End Points

It is possible to configure several Web Service End-Point servers. Each of one of these servers represented by the corresponding `WebServiceEndPoint` **COM object**, has common properties shared among their children represented by `WsDestination` **COM object**. In case a `WebServiceEndPoint` has several `WsDestination` children configured under it, the notification is sent to one of them following a round robin algorithm.

Next, an example on how to configure several `WebServiceEndPoint` and `WsDestination` **COM objects**:

```

<edit-config>
  <target>
    <running/>
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <Network xmlns="urn:com:ericsson:ecim:networkmom">
          <networkId>1</networkId>

```



```

<WebServiceEndpoints xmlns="urn:com:ericsson:ecim:webserviceendpointsmom">
<WebServiceEndpointsId>1</WebServiceEndpointsId>
  <WebServiceEndPoint xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    nc:operation="merge">
    <WebServiceEndPointId>BBSC</WebServiceEndPointId>
    <connectionTimeout>3000</connectionTimeout>
    <maxNumberRetries>1</maxNumberRetries>
    <soapAction>http://operator1.org/abc#MySoapNotif</soapAction>
    <WsDestination xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
      nc:operation="merge">
      <wsDestinationId>http://10.1.22.337:11222/webService/service1
      </wsDestinationId>
      <httpProxy></httpProxy>
    </WsDestination>
    <WsDestination xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
      nc:operation="merge">
      <wsDestinationId>http://10.1.22.338:11222/webService/service2
      </wsDestinationId>
      <httpProxy>http://134.15.20.229:11000</httpProxy>
    </WsDestination>
  </WebServiceEndPoint>
  <WebServiceEndPoint xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    nc:operation="merge">
    <WebServiceEndPointId>WsUsageLimitSurpassed</WebServiceEndPointId>
    <connectionTimeout>3000</connectionTimeout>
    <maxNumberRetries>1</maxNumberRetries>
    <soapAction>post</soapAction>
    <WsDestination xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
      nc:operation="merge">
      <wsDestinationId>http://10.1.22.339:11222/webService/service3
      </wsDestinationId>
      <httpProxy>http://134.15.20.229:11000</httpProxy>
    </WsDestination>
  </WebServiceEndPoint>
  <WebServiceEndPoint xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    nc:operation="merge">
    <WebServiceEndPointId>WebServiceEndPoint1</WebServiceEndPointId>
    <connectionTimeout>3000</connectionTimeout>
    <maxNumberRetries>1</maxNumberRetries>
    <soapAction>post</soapAction>
    <WsDestination xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
      nc:operation="merge">
      <wsDestinationId>http://192.168.14.42:11222</wsDestinationId>
      <httpProxy></httpProxy>
    </WsDestination>
  </WebServiceEndPoint>
  <WebServiceEndPoint xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    nc:operation="merge">
    <WebServiceEndPointId>WebServiceEndPoint2</WebServiceEndPointId>
    <connectionTimeout>3000</connectionTimeout>
    <maxNumberRetries>1</maxNumberRetries>
    <soapAction>post</soapAction>
    <WsDestination xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
      nc:operation="merge">
      <wsDestinationId>http://192.168.14.42:11222</wsDestinationId>
      <httpProxy></httpProxy>
    </WsDestination>
  </WebServiceEndPoint>
</WebServiceEndpoints>
</Network>
</PolicyControlFunction>
</ManagedElement>
</config>
</edit-config>

```

### Example 3 Configuration of several WebServiceEndPoint and WsDestination objects

The Example 3 configures four WebServiceEndPoint objects (BBSC, WsUsageLimitSurpassed, WebServiceEndPoint1 and WebServiceEndPoint2) under the WebServiceEndpoints object “1”. The WebServiceEndPoint





object “BBSC” has two WsDestination. First one with URL `http://10.1.22.337:11222` and no `httpProxy` configured and second one with URL `http://10.1.22.338:11222` and `httpProxy http://134.15.20.229:11000`. The `WebServiceEndPoint` object “WsUsageLimitSurpassed” has only one WsDestination configured with URL `http://10.1.22.339:11222` and `httpProxy http://134.15.20.229:11000`. The latter `WebServiceEndPoint` objects, “WebServiceEndPoint1” and “WebServiceEndPoint2”, share the same WsDestination (`http://192.168.14.42:11222`) with no `httpProxy` attached.

## 3.3 Configure Notification Destinations

Next step is to configure the destination of the notifications.

### 3.3.1 Notifications Sent to End Users

The address for notifications sent to end users can be provisioned at subscriber or at subscriber group level.

- Subscriber group level. This option makes easier the provisioning, avoiding to set a destination address in all subscribers belonging to that group.

Use notification attribute through `dataplan` URI in the provisioning REST API.

- For SMS: set `sms` value. The SAPC sends the SMS to the Mobile Subscriber ISDN Number (MSISDN) of the subscriber for which the notification policies are evaluated. It is important to assure that the traffic request sent from the PCEF to the SAPC satisfies one of the following conditions:
  - Either its first `Subscription-Id` occurrence contains a valid MSISDN identifier.
  - Or it contains several `Subscription-Id` AVP occurrences, at least one being a valid MSISDN identifier.

Otherwise, it is not possible to guarantee that the SAPC sends the SMS to the proper destination.

- Subscriber level.

Use `smsDestinations` attribute through `subscriber` URI in the provisioning REST API.

- For SMS: fill `smsDestinations` attribute with one or more subscriber destination addresses.

If at least one destination address is provisioned for a subscriber, provisioning at subscriber group level is not used; that is, notification addresses set at subscriber level have precedence over the ones provisioned at subscriber group level.



Next, an example of provisioning of a notification address at subscriber level:

```
PUT /subscribers/34600010101
{
  "dataplanName" :
  [
    {
      "dataplanName" : "Silver"
    }
  ],
  "smsDestinations" : [ "+341111111" ],
  "subscriberId" : "34600010101"
}
```

#### Example 4 Configuration of SMS Notification Data at subscriber level

Next, an example of provisioning at subscriber group level:

```
PUT /dataplanName/Gold
{
  "dataplanName" : "Gold",
  "notification" : "sms"
}

PUT /subscribers/34600010201
{
  "dataplanName" :
  [
    {
      "dataplanName" : "Gold"
    }
  ],
  "subscriberId" : "34600010201"
}
```

#### Example 5 Configuration of SMS Notification Data at subscriber group level

### 3.3.2 Notifications Sent to External Systems

In the SAPC when the destination of a notification is not a subscriber but an external system it is needed to configure:

- One or several notification receiver identities.
- The notification message text format.

The notification mechanisms available for external systems are SMS and SOAP.



## 4 Provisioning Notification Policies

The final step for configuring notifications is to configure notification policies as it is shown in the following steps:

— For **Global policy locator**:

```
/locators/resources/any/contexts/notification
```

— For **Subscriber group locator**:

```
/dataplanes/<dataplanName>/locators/resources/any/contexts/no  
tification
```

— For **Subscriber locator**:

```
/subscribers/<subscriberId>/locators/resources/any/contexts/no  
tification
```

— Within the `outputAttributes` object in the rule, set:

- `attrName` attribute to `notification`
- `attrValue` attribute is the notification message content. Fill it depending on the type of notifications: End users or external systems. Read in next subsections the how to.
- `result` attribute is fixed to `permit`

For configuration examples containing notification policies, see Section 5 on page 27.

---



---

### Warning!

Regarding the **business condition** that causes the notification to be sent: The same notification message can be sent several times if policy tags whose value can be different for several concurrent active IP-CAN sessions of a Subscriber are used.

For example, this can happen with the following:

```
AccessData.subscriber.ueIpAddress,  
AccessData.subscriber.session.accumulatedUsage.reportingGroup[x].i  
sLimitSurpassed and  
AccessData.bearer.accessPoint.
```

Therefore, to avoid this notification repetition, Ericsson **does not** recommend using this kind of policy tags.

---



---



## 4.1 Notifications Sent to End Users

For end users notifications, fill the attribute `attrValue` of `outputAttributes`, using the following syntax:

"<notification message>"

Next, an example on how to configure the notification data message for end users notifications:

```
PUT /rules/rFirstBidirVolumeLimitSurpassed
{
  "condition" : "AccessData.subscriber.accumulatedUsage.reportingGroup[\"401\"].
                isLimitSurpassed[\"bidirVolume\"][0]",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "\"First bidir Volume Limit Surpassed for reportingGroup 401\"",
      "result" : "permit"
    }
  ],
  "ruleName" : "rFirstBidirVolumeLimitSurpassed"
}
```

Example 6 Configuration of Notification Data Message for End Users

## 4.2 Notifications Sent to External Systems

Next figure shows the detailed elements to configure for sending notifications to external systems:

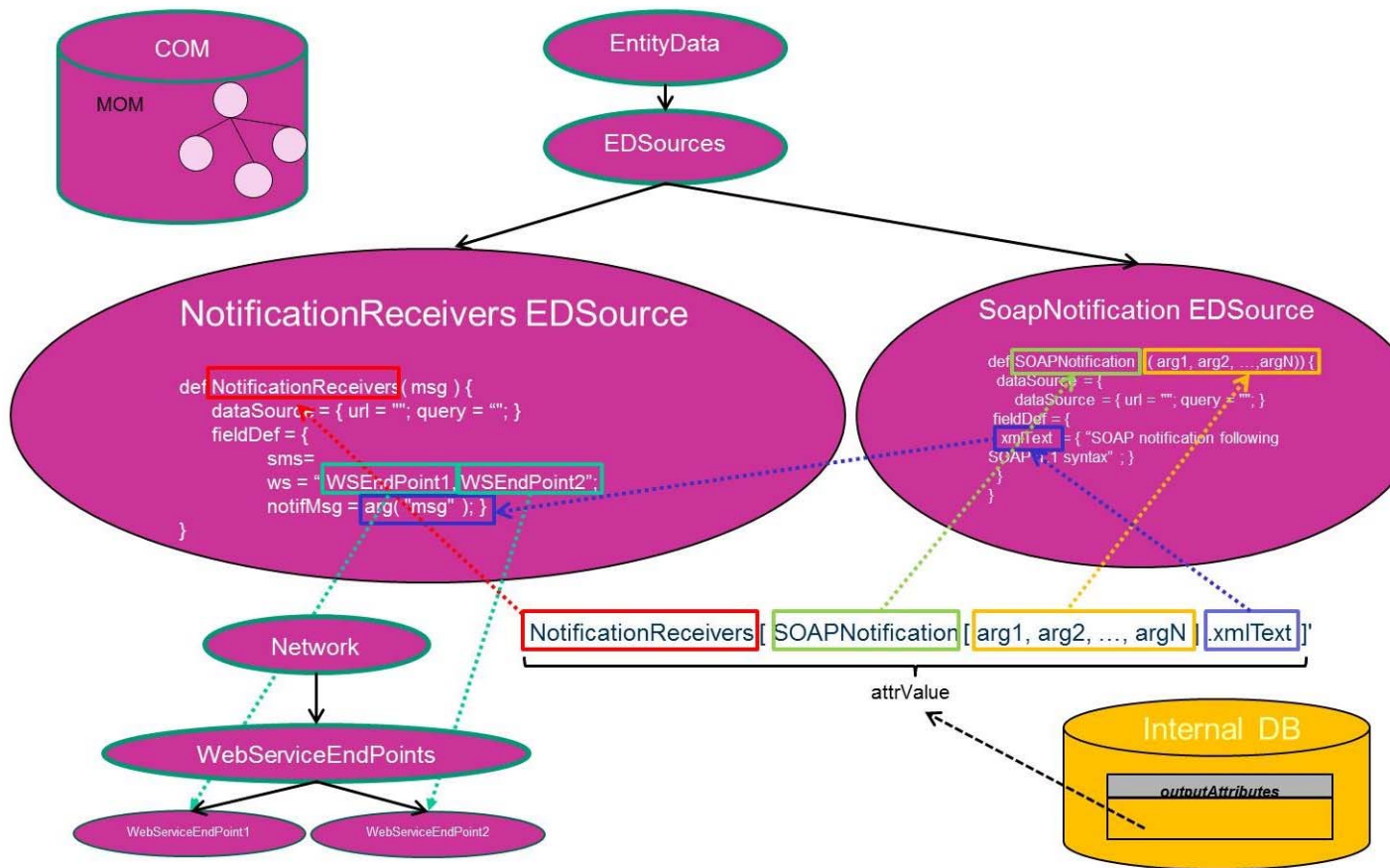


Figure 4 Elements to configure notifications to send to External Systems

In notifications sent to External Systems, the `attrValue` attribute contained in the `outputAttributes` object contains the notification receiver identities and the notification text, using the following syntax:

```
NotificationReceivers[ <notification msg> ]
```

**NotificationReceivers** is an instance name of an **EDSource** COM object. This **NotificationReceivers** includes both the list of the notification destinations and the notification message.

Thus, to configure in the SAPC a **NotificationReceivers**, create an **EDSource** COM object and fill:

- `eSourceId` attribute with the name of the notification receivers (in the example of Page 16, **NotificationReceivers**).
- definition attribute with the following format:
  - The tag element `def` and the name of the notification receivers. Ensure that is equal both to the `eSourceId` attribute of the **EDSource** object and the name filled in the `attrValue` attribute. See Page 16.



- Inside `fieldDef` element, it is mandatory to fill depending on SMS or SOAP mechanism, with the fixed tag `sms` or `ws` followed respectively by the list of SMS addresses or the list of `WebServiceEndPoint` COM objects separated by “,” and bounded by “ ”.
- Inside `fieldDef` element, it is mandatory to add the fixed string `notifMsg` = `arg( "msg" )` to manage the notification text.

Next the syntax of the main elements of the definition attribute of a `NotificationReceivers` **EDSource**:

```
def NotificationReceivers( msg ){  
    dataSource = { url = ""; query = ""; }  
    fieldDef = {  
        sms = "MobileNumber1,MobileNumber2,...,MobileNumberN";  
        ws = "WebServiceEndPoint1,...,WebServiceEndPointN";  
        notifMsg = arg( "msg" ); }  
}
```

---

---

### Warning!

Do not include in the same notification receiver a SOAP destination with SMS destination because the message format is not compatible.

---

---

Following with **NotificationReceivers**, it contains the notification text and depending on the type of mechanism the format is different:

#### Configure SMS Notification Message

For SMS mechanism, fill it with the notification message text following this syntax:

```
NotificationReceivers[" < Notification message text > "]
```

Next, an example of both `outputAttributes` object and `NotificationReceivers` **EDSource** object to configure `NotificationReceivers` for SMS mechanism:



```
PUT /rules/rFirstBidirVolumeLimitSurpassed
{
  "condition" : "AccessData.subscriber.accumulatedUsage.reportingGroup[\"401\"]
               isLimitSurpassed[\"bidirVolume\"][0]\",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "\"First bidir Volume Limit Surpassed for reportingGroup 401\"",
      "result" : "permit"
    },
    {
      "attrName" : "notification",
      "attrValue" : "SmsNotifReceivers[\"First bidir Volume Limit Surpassed for reportingGroup 401\"]",
      "result" : "permit"
    }
  ],
  "ruleName" : "rFirstBidirVolumeLimitSurpassed"
}
```

### Example 7 Configuration of outputAttributes object with NotificationReceivers for SMS Notifications for SMS Notifications

```
<edit-config>
  <target>
    <running/>
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
            <eDSourcesId>1</eDSourcesId>
            <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom"
                      xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
                      nc:operation="merge">
              <eDSourcesId>SmsNotifReceivers</eDSourcesId>
              <definition>
                def SmsNotifReceivers ( msg )
                {
                  dataSource = {
                    url = "";
                    query = "";
                  }
                  fieldDef = {
                    sms = "+34609102030,+34609908070";
                    notifMsg = arg( "msg" );
                  }
                }
              </definition>
            </EDSource>
          </EDSources>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>
```

### Example 8 Configuration of NotificationReceivers EDSource object for SMS notifications

The previous examples represent the **SmsNotifReceivers** **eDSource** that contains two destinations: the first one corresponds to the mobile number +34609102030 and the second one corresponds to the mobile number +34609908070. The



notification text “First bidir Volume Limit Surpassed for reportingGroup Total” is passed to SmsNotifReceivers.

### Configure SOAP Notification Message

For SOAP mechanism, fill it with the notification message text following this syntax:

```
NotificationReceivers[<New eDSourceId>[<arg1,...,argN>].xmlText]
```

In SOAP mechanism, instead of specifying the notification message itself, it can be formatted in an XML message. The XML message content depends on what the external Web Service End Point service needs.

To configure in the SAPC the XML message content, create another **EDSource** COM object with a `xmlText` fieldDef that contains the XML notification text. This new **EDSource** COM object is the argument of the notification receiver **EDSource** mentioned before and it can receive several input arguments. The input arguments serve to pass variable values inside the content of the XML message. This can be done using policy tags (see Section 4.4 on page 23) as input arguments.

Next an example on how to configure a NotificationReceivers **EDSource** for SOAP Notifications:

```
PUT /rules/rSecondBidirVolumeLimitSurpassed
{
  "condition" : "AccessData.subscriber.accumulatedUsage.reportingGroup[\"401\"].
                isLimitSurpassed[\"bidirVolume\"][1]",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "SOAPNotificationReceivers[SOAPNotification[AccessData.
                    subscriber.msisdn, AccessData.subscriber.imsi].xmlText]",
      "result" : "permit"
    }
  ],
  "ruleName" : "rSecondBidirVolumeLimitSurpassed"
}
```

#### Example 9 Configuration of the outputAttributes object for SOAP notifications





```

<edit-config>
  <target>
    <running/>
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
            <eDSourcesId>1</eDSourcesId>
            <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom"
              xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
              nc:operation="merge">
              <eDSourcesId>SOAPNotificationReceivers</eDSourcesId>
              <definition>
                def SOAPNotificationReceivers ( msg )
                {
                  dataSource = {
                    url = "...";
                    query = "...";
                  }
                  fieldDef = {
                    notifMsg = arg( "msg" );
                    ws = "WebServiceEndPoint1,WebServiceEndPoint2";
                  }
                }
              </definition>
            </EDSource>
          </EDSources>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>

```

#### Example 10 Configuration of NotificationReceivers EDSource object for SOAP notifications

The preceding examples represent the SOAPNotificationReceivers **eDSources** object that contains two destinations: WebServiceEndPoint1 and WebServiceEndPoint2.

See next table to understand how input arguments are used:

Table 2 Example of Policy tags to use as Input Parameters in SOAP Notifications

XML Argument	Argument in SOAPNotification	SOAP Input Argument
MSISDN	arg1	AccessData.subscriber.msisdn
IMSI	arg2	AccessData.subscriber.imsi

Next an example on how to create the **EDSource** object for the SOAP message, with the arguments of the example above:



```

<edit-config>
  <target>
    <running/>
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
            <eDSourcesId>1</eDSourcesId>
            <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom"
              xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
              nc:operation="merge">
              <eDSourcesId>SOAPNotification</eDSourcesId>
              <definition>
                def SOAPNotification( arg1, arg2 )
                {
                  dataSource = {
                    url = "";
                    query = "";
                  }
                  fieldDef = {
                    xmlText = <![CDATA["<soapenv:Envelope xmlns:soapenv='http://schemas.
                                                                xmlns:xsd='http://www.w3.org/2001/XML
                                                                xmlns:xsi='http://www.w3.org/2001/XMLS
                                                                xmlns:tns='http://ericsson.com/sapc/so
                                                                <soapenv:Body>
                                                                <tns:SAPCNotification xmlns:soapNotifTypes='http:
                                                                <Message>
                                                                <EventType>notification</EventType>
                                                                <MSISDN>" + arg(arg1) + "</MSISDN>
                                                                <IMSI>" + arg(arg2) + "</IMSI>
                                                                </Message>
                                                                </tns:SAPCNotification>
                                                                </soapenv:Body>
                                                                </soapenv:Envelope>";
                                                                ]]>
                                                                }
                                                                }
                                                                </definition>
                                                                </EDSource>
                                                                </EDSources>
                                                                </EntityData>
                                                                </PolicyControlFunction>
                                                                </ManagedElement>
                                                                </config>
                                                                </edit-config>

```

Example 11 Configuration by means of an EDSource for the SOAP notification text

The SOAPNotification **eDSources** of the example contains an XML format text with next predefined sections:

- The SOAP Envelope including all namespaces definitions. There are two mandatory namespaces with fixed values, but with configured names: soapenv and xsd:
  - xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'
  - xmlns:xsd='http://www.w3.org/2001/XMLSchema'
- The SOAP message body containing the formatted notification message:



- Each XML tag of the SOAP notification message is configured by the operator in this section.
- The information is in this case dynamic by using MSISDN and IMSI as XML arguments and EventType XML element is fixed to value notification.

---

### Warning!

Only SOAP 1.1 protocol is supported, so the SOAP message format and namespaces values has to be configured accordingly.

---

## 4.3 Notification Sent to End Users and to External Systems

When the same business condition is wanted to be triggered towards different nature destinations, the SAPC can combine it to send to end users and external systems, by adding several elements into the outputAttributes object.

Next an example on how to combine both notifications types:

```
PUT /rules/rFirstBidirVolumeLimitSurpassed_1
{
  "condition" : "AccessData.subscriber.accumulatedUsage.reportingGroup[\"total\"].
                isLimitSurpassed[\"bidirVolume\"]{0}",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "\"Usage limit has been surpassed, please visit ou
                    r portal to extend your subscribed limit\"",
      "result" : "permit"
    },
    {
      "attrName" : "notification",
      "attrValue" : "SmsNotifReceivers[\"First bidir Volume Limit Surpassed\"]",
      "result" : "permit"
    },
    {
      "attrName" : "notification",
      "attrValue" : "SOAPNotificationReceivers[SOAPNotification[AccessData.
                    subscriber.msisdn].xmlText]",
      "result" : "permit"
    }
  ],
  "ruleName" : "rFirstBidirVolumeLimitSurpassed_1"
}
```

### Example 12 Combination of Notification Data Message for End Users and External Systems

In the example above, the first notification is a SMS notification sent to the subscriber configured (see Example 4 or Example 5), the second one is sent to the list of destinations specified within “SmsNotifReceivers” that sends the notification using SMS mechanism, and the last one is sent to the list of destinations specified within “SOAPNotificationReceivers” that sends the notification using SOAP mechanism.

## 4.4 Dynamic Notification Message Content

It is possible to include **dynamic** information within the notification message, by using any of the provided policy tags in Configuration Guide for Access and Charging Control (Gx) , Configuration Guide for Fair Usage and policy functions in Configuration Guide for Subscription and Policies.

For example, consider that several Subscription-Ids are received in the Gx traffic request, that the MSISDN is used to locate the subscriber profile (and in receipt of destination for the SMS), but consider that the IMSI is wanted to be included in the notification message. An outputAttributes like the following can be used:

```
"outputAttributes" :
[
  {
    "attrName" : "notification",
    "attrValue" : "strcat(\"Subscriber is roaming,
                    IMSI=\",AccessData.subscriber.imsi)",
    "result" : "permit"
  }
]
```

Another example: Imagine it is wanted to include in the notification message the current Fair Usage accumulator and expiration date at the end of each session (business condition AccessData.bearer.requestType==3). A dynamic string composition for the notification message is needed, to produce a message such as the following:

**You have consumed: <XX> KBytes. Your subscription is valid till: <dd-mm-yyyyThh:mm:ss>**

The configuration to be done in the rule resource is the following:

```
PUT /rules/r_notifDynamicTextVolume
{
  "condition" : "AccessData.bearer.requestType == 3",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "strcat(\"You have consumed: \",
                          strcat( AccessData.subscriber.accumulatedUsage.reportingGroup[\"total\
                          current[\"bidirVolume\"],
                          strcat(\" KBytes.\",
                          strcat(\" Your subscription is valid till: \",
                          AccessData.subscriber.accumulatedUsage.reportingGroup[\"total
                          expiryDate[\"volume\"] ]))\"),
      "result" : "permit"
    }
  ],
  "ruleName" : "r_notifDynamicTextVolume"
}
```

**Example 13** Configuration to include dynamic information within the notification message



## 4.5 Encoding Schemes in Notification Messages

The SAPC supports the following encoding schemes:

- ASCII and UTF-8 (Unicode) for SOAP notification messages.
- ASCII and UTF-16 (Unicode) for SMS notification messages.

When the notification message contains non-ASCII characters (for example for notification messages to be sent in chinese, arabic or cyrillic alphabet), use Base64 format for the value of the `attrValue` attribute within the `outputAttributes` object.





## 5 Configuration Examples for Use Cases

### 5.1 Notifications Based on Fair Usage

#### 5.1.1 Notifications to be sent to Subscribers

Let us suppose a Subscriber Fair Usage with limits for total traffic (IP-CAN session):

```
PUT /subscribers/34600000200
{
  "subscriberId" : "34600000200",
  "smsDestinations" : [ "+341111111" ],
  "usageLimits" :
  [
    {
      "absoluteLimits" :
      {
        "bidirVolume" : [ 1572864, 2097152 ],
        "resetPeriod" :
        {
          "volume" : "monthly day 12 19:30"
        }
      },
      "description" : "Total traffic"
    }
  ]
}
```

#### Example 14 Configuration of a Subscriber Fair Usage Profile

And, next an example of configuration of a notification policy based on limits surpassed:



```

PUT /rules/notifRuleVolumeLimit
{
  "condition" : "AccessData.subscriber.accumulatedUsage.reportingGroup[\"total\\\"].
                isLimitSurpassed[\"bidirVolume\\\"] [1] ",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "\"Usage limit has been surpassed, please visit our portal to ex
                    tend your subscribed limit\\\"",
      "result" : "permit"
    }
  ],
  "ruleName" : "notifRuleVolumeLimit"
}

PUT /policies/notifPolicyVolumeLimit
{
  "policyName" : "notifPolicyVolumeLimit",
  "ruleCombiningAlgorithm" : "all-permit",
  "rules" : [ "notifRuleVolumeLimit" ]
}

PUT /locators/resources/any/contexts/notification
{
  "policies" : [ "notifPolicyVolumeLimit" ]
}

```

#### Example 15 Configuration of Notification Policy based on Fair Usage

With the configuration above, a notification is sent for all subscribers when their respective provisioned usage limit, for total traffic (IP-CAN session), is reached. The notification text is the one specified in the `attrValue` attribute.

For the particular case of Subscriber 3460000200 (see Example 14) the limit considered for sending the notification is 2 Gbytes; the notification is sent using SMS mechanism (see Example 4 `smsDestinations` attribute) and the SAPC uses the data configured in to send the notification through the SMS center.

## 5.1.2 Notifications and Shared Dataplans

The following examples show how to set the notification policies when the subscribers receiving the notifications are members of a shared dataplan.

### Notifications to each member of the Shared Dataplan

Consider a shared dataplan. Individual limits per member are defined and each subscriber wants to be notified when the corresponding individual limit is surpassed. To model this, it is necessary to define a notification policy per individual limit.

Next an example of this use case that extends the configuration of a Fair Usage use case. See the Fair Usage related configuration in [Configuration Guide for Fair Usage](#):

```

PUT /rules/rSubscBLimitSurpassed
{
  "condition" : "(AccessData.subscriber.id==\"34600702041\") && (AccessData.subscriber.
                accumulatedUsage.reportingGroup[\"1111\\\"].counter[\"Subsc2\\\"].
                isLimitSurpassed[\"bidirVolume\\\"])",

```





```

        "outputAttributes" :
        [
            {
                "attrName" : "notification",
                "attrValue" : "\"Sublimit established for subscriber 2 has been surpassed\"",
                "result" : "permit"
            }
        ],
        "ruleName" : "rSubscBLimitSurpassed"
    }
}

PUT /rules/rSubscCLimitSurpassed
{
    "condition" : "(AccessData.subscriber.id==\"34600702042\") && (AccessData.subscriber.
        accumulatedUsage.reportingGroup[\"1111\"].counter[\"Subsc3\"].
        isLimitSurpassed[\"bidirVolume\"])",
    "outputAttributes" :
    [
        {
            "attrName" : "notification",
            "attrValue" : "\"Sublimit established for subscriber 3 has been surpassed\"",
            "result" : "permit"
        }
    ],
    "ruleName" : "rSubscCLimitSurpassed"
}

PUT /policies/pIndividualLimitsSurpassed
{
    "policyName" : "pIndividualLimitsSurpassed",
    "ruleCombiningAlgorithm" : "all-permit",
    "rules" : [ "rSubscBLimitSurpassed", "rSubscCLimitSurpassed" ]
}

PUT /dataplan/StreamingSharedPlan/locators/resources/any/contexts/notification
{
    "policies" : [ "pIndividualLimitsSurpassed" ]
}

PUT /dataplan/StreamingSharedPlan
{
    "dataplanName" : "StreamingSharedPlan",
    "notification" : "sms",
    "usageLimits" :
    [
        {
            "absoluteLimits" :
            {
                "bidirVolume" : 1228800,
                "conditionalLimits" :
                [
                    {
                        "bidirVolume" : 307200,
                        "name" : "Subsc2"
                    },
                    {
                        "bidirVolume" : 409600,
                        "name" : "Subsc3"
                    }
                ],
                "resetPeriod" :
                {
                    "volume" : "monthly"
                }
            },
            "name" : "1111"
        }
    ]
}

```

#### Example 16 Configuration of Notifications For Members of a Shared Subscriber Plan



In this example, notification is set to “sms” within dataplan, so the notification is sent to the MSISDN of the subscriber for which the notification policies are evaluated. The dataplanName is set to the value of the dataplanName associated to the shared dataplan.

Define one notification per individual limit. To assure that the notification is only sent to the subscriber whose limit is surpassed, it is necessary to include in the condition the particular subscriber Id (AccessData.subscriber.id==\"<...>\") each individual limit applies to (AccessData.subscriber.accumulatedUsage.reportingGroup[\"1111\"]).counter[\"<...>\"].isLimitSurpassed[\"bidirVolume\"]).

### Notification to the Head Member of the Shared Dataplan

Consider a shared dataplan. Category limits for shared dataplan members are defined and only the subscriber within “head” category (the head subscriber could typically be the one paying the shared dataplan) is notified when any of the shared dataplan limits is surpassed. To model this, it is necessary to define a notification condition per limit (one for the general shared dataplan limit and one for each category limit).

Here is an example that extends the configuration of a Fair Usage use case. See the Fair Usage related configuration in [Configuration Guide for Fair Usage](#)

```
PUT /rules/rHeadLimitSurpassed
{
  "condition" : "(Subscriber.category==\"head\") && (AccessData.subscriber.accumulatedUsage.
    reportingGroup[\"Internet\"].counter[\"head\"].
    isLimitSurpassed[\"bidirVolume\"])",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "\"Limit Surpassed for your category\"",
      "result" : "permit"
    }
  ],
  "ruleName" : "rHeadLimitSurpassed"
}

PUT /rules/rRegularLimitSurpassed
{
  "condition" : "(Subscriber.category==\"regular\") && (AccessData.subscriber.accumulatedUsage.
    reportingGroup[\"Internet\"].counter[\"regular\"].
    isLimitSurpassed[\"bidirVolume\"])",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "\"Limit Surpassed for your category\"",
      "result" : "permit"
    }
  ],
  "ruleName" : "rRegularLimitSurpassed"
}

PUT /rules/rTotalLimitSurpassed
{
  "condition" : "AccessData.subscriber.accumulatedUsage.reportingGroup[\"Internet\"].
    isLimitSurpassed[\"bidirVolume\"]",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "\"Shared Internet limit surpassed\"",

```



```

        "result" : "permit"
      }
    ],
    "ruleName" : "rTotalLimitSurpassed"
  }
}

PUT /policies/pInternetNotifications
{
  "policyName" : "pInternetNotifications",
  "ruleCombiningAlgorithm" : "all-permit",
  "rules" :
  [
    "rRegularLimitSurpassed",
    "rHeadLimitSurpassed",
    "rTotalLimitSurpassed"
  ]
}

PUT /dataplan/InternetSharedPlan/locators/resources/any/contexts/notification
{
  "policies" : [ "pInternetNotifications" ]
}

PUT /subscribers/34600702320
{
  "operatorSpecificInfos" :
  [
    {
      "attributeName" : "category",
      "attributeValue" : "head"
    }
  ],
  "sharedDataplan" : "DoeFamily",
  "smsDestinations" : [ "34600702320" ],
  "subscriberId" : "34600702320"
}

PUT /subscribers/34600702321
{
  "operatorSpecificInfos" :
  [
    {
      "attributeName" : "category",
      "attributeValue" : "regular"
    }
  ],
  "sharedDataplan" : "DoeFamily",
  "smsDestinations" : [ "34600702320" ],
  "subscriberId" : "34600702321"
}

PUT /subscribers/34600702322
{
  "operatorSpecificInfos" :
  [
    {
      "attributeName" : "category",
      "attributeValue" : "regular"
    }
  ],
  "sharedDataplan" : "DoeFamily",
  "smsDestinations" : [ "34600702320" ],
  "subscriberId" : "34600702322"
}

```

### Example 17 Configuration of Notifications to Head of Shared Subscriber Plan

In this example, the `smsDestinations` attribute is defined for each subscriber of the shared dataplan, and it contains the addresses of the “head” member (SMS to “34600702320”). The `dataplanName` is set to the same value as the `dataplanName` associated to the shared dataplan.



One notification per limit is defined:

- For the category limits, include in the condition the particular subscriber category (`Subscriber.category=="<...>"`), and the limit (`AccessData.subscriber.accumulatedUsage.reportingGroup["Internet\"].counter["<...>"].isLimitSurpassed["bidirVolume"]`).
- For the shared limit, include in the condition the limit (`AccessData.subscriber.accumulatedUsage.reportingGroup["Internet\"].isLimitSurpassed["bidirVolume"]`).

**Note:** Head subscriber “34600702320” can receive the same notification, one for each different subscriber reaching the category limit or the shared limit.

To avoid that the Head subscriber receives the same notification several times, extend the condition for each limit with the following expression:

```
&& ( (<...>.current["type\"][n] * 1024) -
      AccessData.subscriber.receivedUsage.reportingGroup
        ["total\"]/["reportingGroupName\"].usageType["type\"])
      < ( <corresponding limit expressed in bytes> ) )
```

In the previous example:

- For the shared limit (1.2 Gbytes):

```
condition:
(AccessData.subscriber.accumulatedUsage.reportingGroup
 ["Internet\"]).isLimitSurpassed["bidirVolume"])&&
(((AccessData.subscriber.accumulatedUsage.reportingGroup
 ["Internet\"]).current["bidirVolume"] * 1024) -
 AccessData.subscriber.receivedUsage.reportingGroup
 ["Internet\"].usageType["bidirVolume"])
 < ( 1228800 * 1024 ) )
```

- For the regular limit (600 Mbytes):

```
condition:
(Subscriber.category=="regular") &&
(AccessData.subscriber.accumulatedUsage.reportingGroup
 ["Internet\"]).counter["regular"].
isLimitSurpassed["bidirVolume"])&&
(((AccessData.subscriber.accumulatedUsage.reportingGroup
 ["Internet\"]).counter["regular"].
current["bidirVolume"] * 1024) -
 AccessData.subscriber.receivedUsage.reportingGroup
 ["Internet\"].usageType["bidirVolume"])
 < ( 614400 * 1024 ) )
```



In case the subscriber surpassing the limit needs to be also notified, consider to add another item to the smsDestinations attribute, reflecting the corresponding subscriber MSISDN. Here is a figure that represents how it should be configured:



## NOTIFICATIONS TO MEMBERS OF A CATEGORY FOR A SHARED SUBSCRIBER PLAN

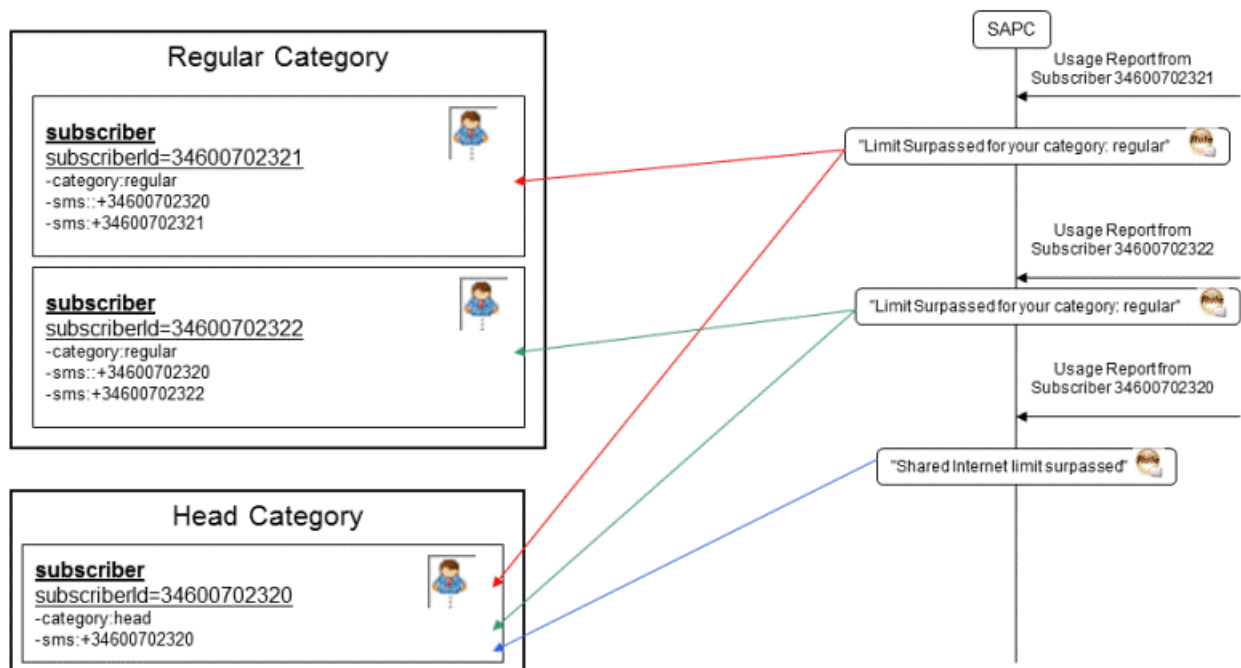


Figure 5 Notifications to Head Member and the corresponding Subscriber

## 5.2 Notifications Based on Subscriber Group Activation or Deactivation

It may be interesting to send a notification to a subscriber when any of the Service Offerings hired (applicable Subscriber Group) is active or inactive. This section covers how to achieve such Notifications.

**Note:** Apart from Notification Policies, this Use Case has sense together with Handling of Multiple Service Offerings or Dynamic Group Selection (for details on how to configure that, refer to Configuration Guide for Subscription and Policies).



To send notifications about activation or deactivation of groups for a Subscriber, use Notification Policies (as explained in Section 4 on page 15) but with the following particularities:

- Associate a Notification Policy at **Global Policy Locator** level using the following URI: `/locators/resources/any/contexts/notification`.

**Note:** Owing to the nature of the group activation/deactivation, to guarantee that such events are notified, do not configure Notification Policies at Subscriber Group level (that is under associated to a specific `dataplanName`), as for groups not being active, such policies are not evaluated.

- Use `Subscription.group["groupName"].isActive` tag within the condition formula of the needed rule.

**Note:** It is **not** recommended to use these notifications when Dynamic Group Selection policies depend on the APN and there are several concurrent IP-CAN sessions for the Subscriber, see Page 15.

Next, an example is provided:



```

PUT /rules/rActivatedCityGroup
{
  "condition" : "(Subscription.group[\"City\"]').isActive)",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "\"INFO about your Subscription Plan\"",
      "result" : "permit"
    }
  ],
  "ruleName" : "rActivatedCityGroup"
}

PUT /rules/rDeactivatedCityGroup
{
  "condition" : "contains(Subscriber.groups, \"City\") &&
    not(Subscription.group[\"City\"]').isActive)",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "\"INFO about your Subscription Plan\"",
      "result" : "permit"
    }
  ],
  "ruleName" : "rDeactivatedCityGroup"
}

PUT /policies/pGroupActivation
{
  "policyName" : "pGroupActivation",
  "ruleCombiningAlgorithm" : "all-permit",
  "rules" : [ "rActivatedCityGroup", "rDeactivatedCityGroup" ]
}

PUT /locators/resources/any/contexts/notification
{
  "policies" : [ "notifPolicyVolumeLimit", "pGroupActivation" ]
}

PUT /subscribers/34600500400
{
  "dataplan" :
  [
    {
      "dataplanName" : "Rustic"
    },
    {
      "dataplanName" : "City"
    }
  ],
  "smsDestinations" : [ "+34600500400" ],
  "subscriberId" : "34600500400"
}

```

#### Example 18 Configuration of Notification based on Subscriber Group activation/deactivation

In the Example 18, notifications by SMS are applied to Subscriber 34600500400. The notification message contains different text depending on whether the City group is active or inactive (assuming that Dynamic Group Selection policies are based on cell location for the Subscriber, refer to [Configuration Guide for Subscription and Policies](#)).

**Note:** contains function is used within the condition formula of the deactivation rule, to assure that such deactivation notification is only sent for the groups actually subscribed to the Subscriber.

**Note:** For a configuration where the SAPC is handling tens or hundreds of different Subscriber Groups (different `dataplanName` entries, that can be applicable for different functions such as Service Access Control, Bearer QoS Control or so on), take care in configuring `rules` just the ones to the groups for which actually it is relevant to send activation/deactivation notifications.

Otherwise, for example if a `rule` (as the Notification message is different for each Subscriber Group id) is done for each of the hundreds of provisioned Subscriber Groups in the system, there may be a significant impact in the SAPC performance owing to the evaluation of useless conditions.

Here, there is another example where Multiple Service Offering is applied for the Subscriber, but there is no Dynamic Group Selection:

```
PUT /rules/notifActiveTurbo
{
  "condition" : "(Subscription.group[\"Turbo\"]').isActive)",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "\"Your Subscription has been updated\"",
      "result" : "permit"
    }
  ],
  "ruleName" : "notifActiveTurbo"
}

PUT /policies/notifActiveTurbo
{
  "policyName" : "notifActiveTurbo",
  "ruleCombiningAlgorithm" : "all-permit",
  "rules" : [ "notifActiveTurbo" ]
}

PUT /locators/resources/any/contexts/notification
{
  "policies" : [ "notifPolicyVolumeLimit", "pGroupActivation", "notifActiveTurbo" ]
}

PUT /subscribers/34600500500
{
  "dataplan" :
  [
    {
      "dataplanName" : "Basic",
      "priority" : 2
    },
    {
      "dataplanName" : "Turbo",
      "priority" : 1,
      "startDate" : "15-12-2020T20",
      "stopDate" : "15-12-2020T22"
    }
  ],
  "smsDestinations" : [ "34600500500" ],
  "subscriberId" : "34600500500"
}
```

### Example 19 Configuration of Temporary Group activation Notification

In the example above, an SMS is sent to “34600500500” when Turbo group becomes active for Subscriber “34600500500”.





# 6      Appendix A. End User Notifications Policy Types

Next figure shows the policy type related to End User Notifications that can be configured in the SAPC.



Notification related Policies					
Policy Type	Policy Locator			Output Attributes	Comments
	Context	Resource	Subject		
Notifications	notification 	any	<subscriberId> <dataplanId> 	permit notification "<Notification message>"  permit notification NotifReceiver["<Notificati on message>"]	Type III = All Permit  Used to send Subscriber notifications Algorithms: all permit

Figure 6    End User Notifications policies in the SAPC





# Glossary

**APN**

Access Point Name

**HTTP**

Hypertext Transfer Protocol

**IP**

Internet Protocol

**NPI**

Numeric Plan Indicator

**PCEF**

Policy Charging Enforcement Function

**REST**

Representational State Transfer

**SAPC**

Ericsson Service-Aware Policy Controller

**SMPP**

Short Message Peer-to-Peer Protocol

**SMS**

Short Message Service

**SMSC**

Short Message Service Center

**SOAP**

Simple Object Access Protocol

**ToN**

Type of Number

**URI**

Uniform Resource Identifier

**UTF**

Unicode Transformation Format

**XML**

Extensive Markup Language