

Integration in Software Defined Network

Ericsson Service-Aware Policy Controller

Description

Copyright

© Ericsson AB 2018. 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](#).



Contents

1	Integration in Software Defined Network Introduction	1
2	Integration in Software Defined Network Solution Description	2
2.1	Solution Overview	2
2.2	Service Chaining	3
2.2.1	Use Case: Content Filtering	4
2.3	Restoration Procedures	5
3	Network Topology	6
4	Interface Description	7
5	Operation and Maintenance	9
5.1	Configuration Management	9
5.1.1	Configuring the subscriber data	9
5.1.2	Configuring BBSC Peers	9
5.1.3	Configuring the SOAP Notification Policies	11
5.1.3.1	Configuring SOAP Notification Policies for assignSubscriberProfile operation	11
5.1.3.2	Configuring SOAP Notification Policies for removeSubscriberProfile operation	13
5.1.3.3	Selecting Different BBSCs Depending on Network Topology	15
5.2	Performance Management	16
5.3	Fault Management	16
5.4	Logging	16
6	Capabilities	17
7	Appendix A. Policy Tags	18
8	Reference List	19





1 Integration in Software Defined Network Introduction

This document describes, functionally, the SAPC integration in the SDN solution, and provides guidelines and examples to configure the SAPC node to integrate in the SDN solution.

2 Integration in Software Defined Network Solution Description

2.1 Solution Overview

SDN is an emerging network architecture that decouples the network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services.

The SDN allows operator to do the following:

- Support subscriber-specific service bundles as a new business model.
- Steer traffic to dedicated service functions determined by the traffic type, ensuring efficient use of the computing resources.
- Identify traffic to dedicated destinations and forward it accordingly.

The Ericsson SDN service chaining solution uses the concept of SDN to steer subscriber traffic flows according to a defined service chain in the operator's service network. The operator's service network processes the subscriber traffic and adds value to the operator or to the subscriber. The services provided by the operator's service network can be firewalls, HTTP header enrichment servers, video optimization servers, web catching servers, and content filtering servers.

The [Figure 1](#) shows a high-level view of Ericsson SDN service chaining solution architecture, including the role of the SAPC in the SDN domain.

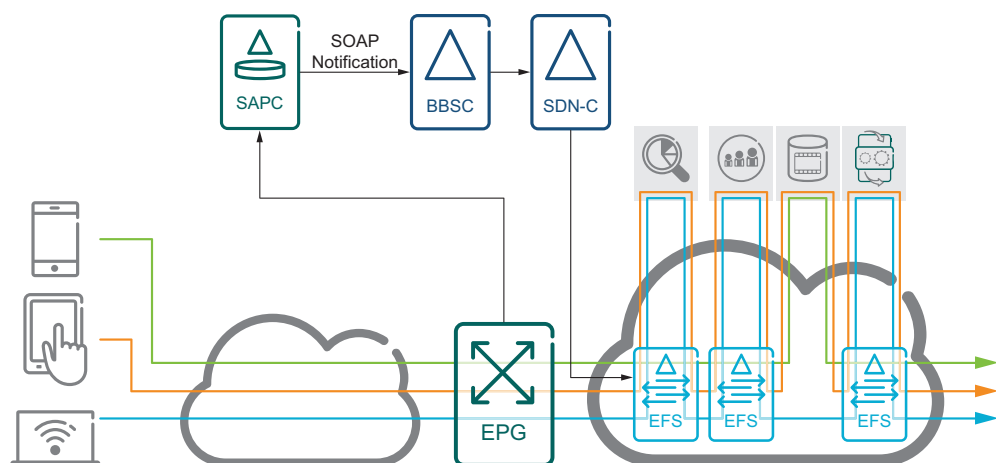


Figure 1 SDN Service Chaining Solution Architecture

The architecture consists of the following components:



EPG	An entry point that injects data flows into the SDN domain.
EFS	A switch that implements OpenFlow™ standards with Ericsson proprietary extensions.
SDN-C	A central controller that provides an abstraction layer of the forwarding plane to the Broadband Service Controller (BBSC) and programs the forwarding behavior of EFSs using OpenFlow™ standards.
BBSC	An intelligent control function that classifies the subscriber traffic according to filter rules and subscriber profile identifiers received from different sources, and instructs the SDN-C to program EFSs for service chaining.
SAPC	<p>A service chain supporting node that provides subscriber-related steering information to the BBSC.</p> <p>Upon detection of certain network events, for example, subscriber session creation or deletion, the SAPC can trigger a SOAP notification to the BBSC through the Dynamic Filter Control (DFC) interface to request the creation or deletion of dynamic filter rules in the BBSC. The notification includes typically subscriber IP addresses and their associated subscriber profile identifier used for the BBSC to classify subscriber's packets to service chains. The subscriber IP address is used to identify traffic flows of a specific subscriber. Multiple subscriber IP addresses (IPv4 and IPv6 prefix) can be associated with the same subscriber profile.</p> <p>SOAP Notifications function is used by the SAPC to notify the BBSC about subscriber information. The availability of this function in the SAPC is under license control. For more information about the SOAP notifications, refer to notification sending procedure in User Notifications.</p>

2.2 Service Chaining

The service chaining is a concept programed at EFSs for specific subscribers by the SDN-C. The service chain consists of a chain of service functions (for example, Content Filtering) for traffic processing. The service function receives packets and sends them back to the SDN domain after processing. Processing means analyzing and altering or deleting packets. The service chain forwards the processed packets further along its data path.

2.2.1

Use Case: Content Filtering

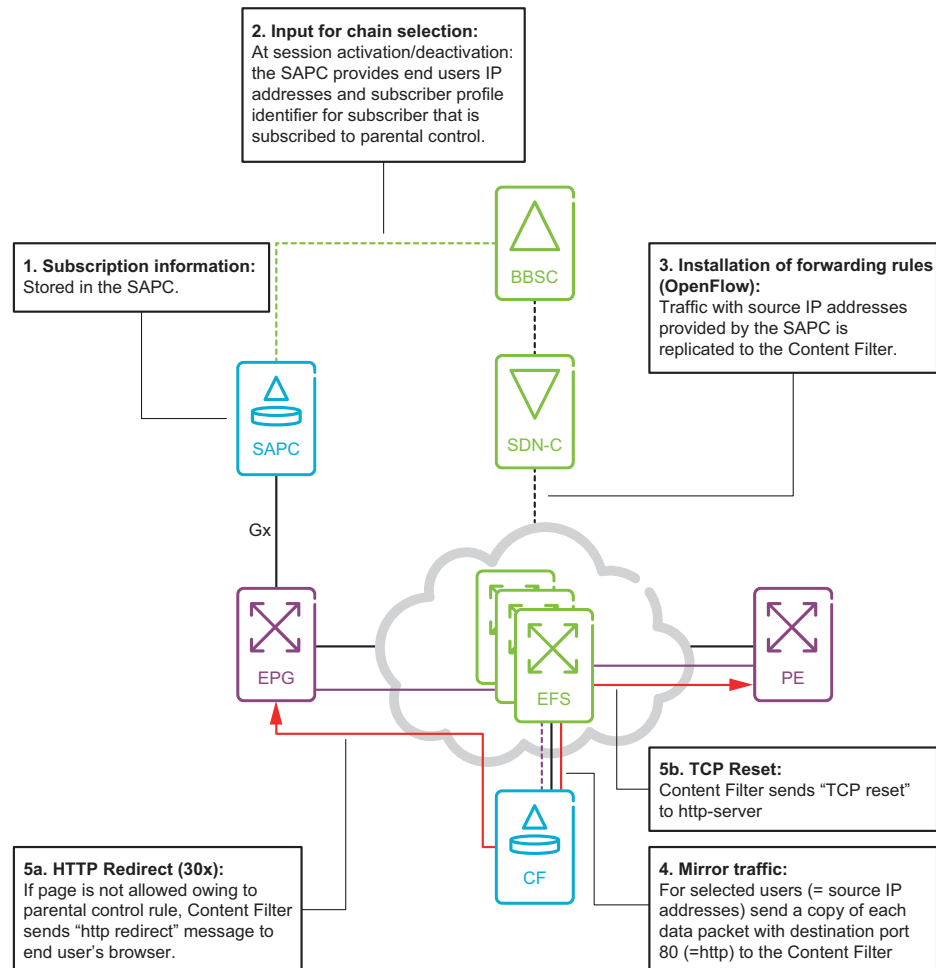


Figure 2 Use Case: Content Filtering

Content Filtering means the analysis of traffic by applying filter rules or policies on traffic with predetermined characteristics. Once subscribed, the subscriber can select different profiles. Based on the profile, certain web pages are made inaccessible. In this way children can, for example, be prevented from accessing adult-oriented web contents.

The subscription to the Content Filtering is stored with other subscriber-specific information as the subscriber profile in the SAPC. When a subscriber registers to the network activating an IP-CAN session, the EPG assigns an IP address to the subscriber and reports the IP address to the SAPC. The SAPC checks the request type (CCR-Initial) and the status of the subscription to the group "ParentalControl" of the subscriber in the notification condition. If the condition is



fulfilled, the SAPC forwards the IP address to the BBSC with the Content Filtering profile ID, requesting the BBSC to update the dynamic filter rules on the subscriber's packet traffic.

The BBSC then installs forwarding rules in the EFSs through the SDN-C. All uplink HTTP traffic for the subscriber is replicated to the Content Filtering service function (CF node in [Figure 2](#)). The Content Filtering service function analyzes the replicated traffic in near real time. If a page is barred for a subscriber, the Content Filtering service function sends the subscriber's browser an HTTP redirection to another page and resets the TCP connection to the web server through the Provider Edge (PE) Internet Gateway.

2.3 Restoration Procedures

When a SAPC restart occurs, the last database session information is not recovered, and therefore all the dynamic data related to sessions are neither recovered. Hence, the SAPC does not send any notification to the BBSC.

When the SAPC detects an EPG restart, a massive cleanup mechanism is performed to remove the obsolete sessions belonging to the restarted EPG. As part of that mechanism, the BBSC receives the indication to stop the corresponding subscriber-profile based service chain.

For more information about restoration procedures in the SAPC, refer to *Availability and Scalability*.

3 Network Topology

In the SDN solution, the SAPC can be deployed in a network where multiple SDN domains exist, with multiple BBSCs. The SAPC is able to select the appropriate BBSC per domain based on the information about the EPG applicable in that SDN domain.

The [Figure 3](#) shows a simple network topology where the SAPC supports the multiple SDN domains.

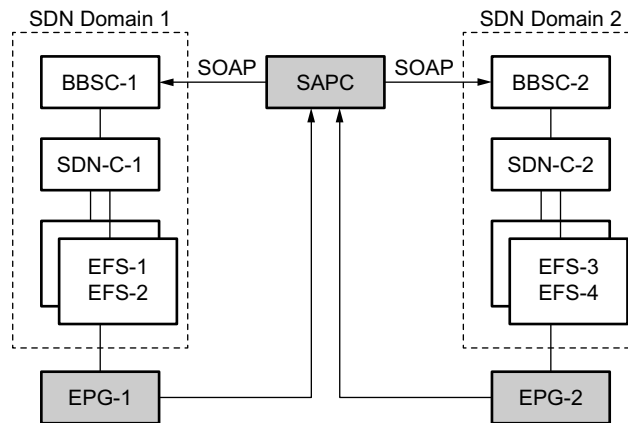


Figure 3 One SAPC for Multiple SDN Domains



4 Interface Description

The DFC interface is opened by the SAPC (client) to the BBSC (server). This interface uses the SOAP protocol carried on a non-encrypted HTTP connection to provide subscriber awareness to the BBSC. Given this information, the BBSC configures the mapping of data flows to service chains for a particular subscriber.

The [Figure 4](#) shows the traffic case over the DFC interface.

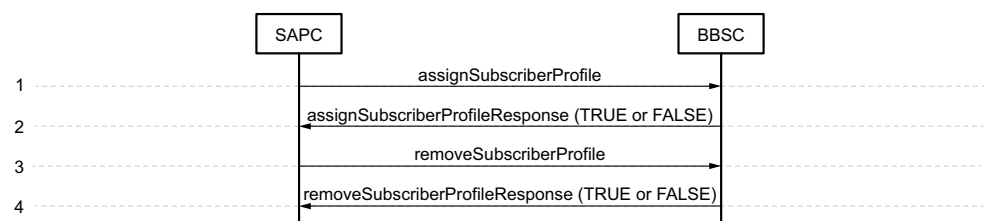


Figure 4 SOAP Operations in DFC Interface

The SAPC sends a SOAP request message `assignSubscriberProfile` when the IP-CAN session establishment is performed. This SOAP request message triggers the start of subscriber-profile based service chain for the indicated subscriber traffic. The BBSC generates the SOAP response message `assignSubscriberProfileResponse` that includes a boolean result for the transaction. The result can be TRUE or FALSE.

The possible parameters within the `assignSubscriberProfile` message are shown in [Table 1](#).

Table 1 Assign Subscriber Profile Message Parameters

Parameter Name	Description	Type	Presence
subscriberProfile	A subscriber profile identifier which corresponds to a BBSC subscriber profile identity configured in the BBSC. The profile identity is used in the service chain selection.	String	Mandatory
subscriberId	The subscriber's identity in the form of the International Mobile Subscriber Identity (IMSI) or Mobile Subscriber ISDN Number (MSISDN)	String	Mandatory
ueIPv4Address	An IPv4 address of the user device	String	Mandatory
ueIPv6Prefix	An IPv6 prefix of the user device	String	Optional
apn	Access Point Name	String	Optional



Parameter Name	Description	Type	Presence
rat	Radio Access Technology	String	Optional

Note: According on how the SOAP WebService is deployed, all fields above are mandatory, that is, they must be present in the request even if they are an empty value. The value Mandatory in the column Presence means that the field must contain a value.

The SAPC sends a SOAP request message `removeSubscriberProfile` when the IP-CAN session termination is performed. This SOAP request message triggers the stop of subscriber-profile based service chain for the indicated subscriber traffic. The BBSC generates the SOAP response message `removeSubscriberProfileResponse` that includes a boolean result for the transaction. The result can be TRUE or FALSE.

The possible parameters within the `removeSubscriberProfile` message are shown in [Table 2](#).

Table 2 Remove Subscriber Profile Message Parameters

Parameter Name	Description	Type	Presence
ueIPv4Address	An IPv4 address of the user device	String	Mandatory
ueIPv6Prefix	An IPv6 prefix of the user device	String	Optional
apn	Access Point Name	String	Optional

Note: According on how the SOAP WebService is deployed, all fields above are mandatory, that is, they must be present in the request even if they are an empty value. The value Mandatory in the column Presence means that the field must contain a value.

In addition, for all SOAP request messages, the SAPC IP address is included in the IP header of the HTTP payload. The IP address is checked against the valid SAPC peer addresses configured in the BBSC and also maps to a service chain forwarding identity in the BBSC.

A correctly processed SOAP message request is answered with a TRUE SOAP message response and HTTP status code 200 indicating success.

Note: For the latest details about the DFC interface, refer to *BBSC-PCRF Interface Description* in BBSC CPI Library.



5 Operation and Maintenance

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

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

To configure the SAPC to use SOAP Notifications, perform the following tasks according to [Configuration Guide for End User Notifications](#):

- Configure the subscribers data in the SAPC.
- Configure the BBSC as a `WsDestination` for a `WebServiceEndPoint` and configure the SOAP notification receiver.
- Configure the notification policies and the SOAP notifications text.

5.1 Configuration Management

5.1.1 Configuring the subscriber data

Configure the `attributeName` "serviceChainingProfileId" and its `attributeValue` in the `operatorSpecificInfos` attribute of the `subscribers` URI in the Provisioning REST API. The `attributeValue` is the same as the `contentFiltering` value configured in the `staticQualification` attribute of the `subscribers` URI in the Provisioning REST API

For more information see the [Configuration Guide for Access and Charging Control \(Gx\)](#) .

5.1.2 Configuring BBSC Peers

The valid BBSC peer that needs to communicate with the SAPC must be configured in the SAPC as a Web Service End Point by using the corresponding `WebServiceEndPoint` and `WsDestination` COM objects.

The following example shows the configuration of a Web Service End Point that corresponds to the BBSC.

Example 1 Configuration of `WebServiceEndPoint` and `WsDestination` COM objects for BBSC

```
<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_Notifications</webServiceEndPointId>
            <connectionTimeout>3000</connectionTimeout>
            <maxNumberRetries>1</maxNumberRetries>
            <soapAction>http://ericsson.com/scsf/webservice/SubsReportingService/</soapAction>
            <WsDestination xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="merge">
              <wsDestinationId>http://10.87.163.178:8090/subsReporting/v1.0/</wsDestinationId>
              <httpProxy></httpProxy>
            </WsDestination>
          </WebServiceEndPoint>
        </WebServiceEndPoints>
      </Network>
    </PolicyControlFunction>
  </ManagedElement>
</config>
</edit-config>

```

The example above configures one Web Service End Point identified as "BBSC_Notifications". The SOAP notifications sent through the Web Service End Point set the "soapAction" HTTP header to the configured value `http://ericsson.com/scsf/webservice/SubsReportingService/`.

The example above configures the BBSC URL "`http://10.87.163.178:8090/subsReporting/v1.0/`" as the unique Web Service Destination for that Web Service End Point.

Example 2 Configuration of SOAP Notification Receiver for BBSC

To specify the BBSC as the SOAP notification receiver, do the following:

- Create an **EDSource** COM object.
- Configure the SOAP notification receiver through the "ws" attribute of the COM object.

```

<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 = "BBSC_Notifications";
                  }
                }
              </definition>
            </EDSource>
          </EDSources>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>

```



```

        </definition>
      </EDSource>
    </EDSources>
  </EntityData>
</PolicyControlFunction>
</ManagedElement>
</config>
</edit-config>

```

The example above represents an **EDSource** COM object identified as "SOAPNotificationReceivers". The receiver of the SOAP notification is the Web Service Destination configured in the Web Service End Point identified as "BBSC_Notifications."

5.1.3 Configuring the SOAP Notification Policies

The SOAP notification condition includes the `AccessData.bearer.requestType` policy tag and some subscriber profile policy tags as dynamic conditions at least. For example, the `Subscription.group["groupName"].isActive` or any other policy tag related with the subscriber.

The `Subscriber.serviceChainingProfileId` policy tag provides the value for the "subscriberProfile" SOAP operation parameter.

In order to use the `Subscriber.serviceChainingProfileId` policy tag as subscriber extra data in policies, as in [Example 4](#), configure the "serviceChainingProfileId" name in the Subscriber **EDSource** COM object.

For more information about policies configuration and how to use subscriber extra data in policies see the [Configuration Guide for Subscription and Policies](#).

To configure the SOAP notification policies, configure the XML notification text for the SOAP operations `assignSubscriberProfile` and `removeSubscriberProfile`.

To set the values needed by the BBSC in the SOAP operations, the SAPC uses the input arguments of the **EDSource** instances containing the notifications text.

5.1.3.1 Configuring SOAP Notification Policies for assignSubscriberProfile operation

The [Example 3](#) permits to compose the content of the SOAP message corresponding to the `assignSubscriberProfile` request operation.

Table 3 The parameters of the assignSubscriberProfile SOAP Operation

SOAP Operation Parameter Name	Argument Name in SOAPNotification_activate	SOAP Notification Input Parameter
subscriberProfile	arg4	Subscriber.serviceChainingProfileId
subscriberId	arg3	AccessData.subscriber.imsi



SOAP Operation Parameter Name	Argument Name in SOAPNotification_activate	SOAP Notification Input Parameter
ueIPv4Address	arg1	AccessData.subscriber.ueIpAddress
ueIPv6Prefix	arg2	-
apn	arg5	AccessData.bearer.accessPoint
rat	arg6	AccessData.bearer.accessType

In the name space declaration, sap is declared as xmlns:sap=http://ericsson.com/sccf/webService/SubsReportingService

Example 3 Configuration for the SOAP assignSubscriberProfile operation

```

<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_activate</eDSourcesId>
              <definition>
                def SOAPNotification_activate( arg1, arg2, arg3, arg4, arg5, arg6)
                {
                  dataSource = {
                    url = "";
                    query = "";
                  }
                  fieldDef = {
                    xmlText = "<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org:
/soap/envelope/'
hema'
ema-Instance'
ervice/SubsReportingService'>
                  <soapenv:Body>
                    <sap:assignSubscriberProfile>
                      <Message>
                        <ueIPv4Address>" + arg(arg1) + "</ueIPv4Address>
                        <ueIPv6Prefix>" + arg(arg2) + "</ueIPv6Prefix>
                        <subscriberId>" + arg(arg3) + "</subscriberId>
                        <subscriberProfile>" + arg(arg4) + "</subscriberProfile>
                        <rat>" + arg(arg5) + "</rat>
                        <apn>" + arg(arg6) + "</apn>
                      </Message>
                    </sap:assignSubscriberProfile>
                  </soapenv:Body>
                </soapenv:Envelope>";
              }
            }
          </EDSource>
        </EDSources>
      </EntityData>
    </PolicyControlFunction>
  </ManagedElement>
</config>
</target>
</edit-config>

```




```

        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>

```

Once the SOAP operation with the notification message format is configured by means of the "SOAPNotification_activate" instance of the **EDSource**, configure the notification rule.

The attrValue of the outputAttributes in the rules URI in the Provisioning REST API contains the notification receiver identities and the notification text using the following syntax:

```
NotificationReceivers[<New eDSourceId>[<arg1, arg2, ..., argN>].xmlText]
```

Example 4 Configuration of Notification Rule Using SOAP EDSource for assignSubscriberProfile operation

[Example 4](#) shows how to configure the notification condition and to associate to it the BBSC assignSubscriberProfile operation (using the SOAPNotification_activate **EDSource** instance).

```

PUT /rules/rSOAP_IpcanActivated
{
  "condition" : "(AccessData.bearer.requestType ==1) && (Subscription.group[\"ParentalControl\
  \"].isActive)",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "SOAPNotificationReceivers[SOAPNotification_activate[AccessDat
      a.subscriber.ueIpAddress, \"\", AccessData.subscriber.imsi, Subscriber.serviceChainingProfileId, Acc
      essData.bearer.accessPoint, AccessData.bearer.accessType].xmlText]",
      "result" : "permit"
    }
  ],
  "ruleName" : "rSOAP_IpcanActivated"
}

```

This example sends a SOAP notification to notify for the IP-CAN session establishment when the subscriber has the "ParentalControl" group active. The SAPC sends the SOAP notification to the web service identified by "BBSC_Notifications". The notification message is configured in the xmlText field definition of the "SOAPNotification_activate"**EDSource** object. The input arguments of the "SOAPNotification_activate" instance receive the needed values to fill the SOAP notification text properly.

5.1.3.2 Configuring SOAP Notification Policies for removeSubscriberProfile operation

The [Example 5](#) permits to compose the content of the SOAP message corresponding to the removeSubscriberProfile request operation.



Table 4 The parameters of the removeSubscriberProfile SOAP Operation

SOAP Operation Parameter Name	Argument Name in SOAPNotification_activate	SOAP Notification Input Parameter
ueIPv4Address	arg1	AccessData.subscriber.ueIpAddress
ueIPv6Prefix	arg2	-

In the name space declaration, sap is declared as xmlns:sap=http://ericsson.com/sccf/webService/SubsReportingService

Example 5 Configuration for the SOAP removeSubscriberProfile operation

```

<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">
              <eDSourceId>SOAPNotification_deactivate</eDSourceId>
              <definition>
                def SOAPNotification_deactivate(arg1, arg2, arg3)
                {
                  dataSource = {
                    url = "";
                    query = "";
                  }
                  fieldDef = {
                    xmlText = "<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org:
/soap/envelope/'
                                xmlns:xsd='http://www.w3.org/2001/XMLSchema
                                xmlns:xsi='http://www.w3.org/2001/XMLSchema
                                xmlns:sap='http://ericsson.com/sccf/webs
hema'
ema-Instance'
ervice/SubsReportingService'>
                                <soapenv:Body>
                                  <sap:removeSubscriberProfile>
                                    <Message>
                                      <ueIPv4Address>" + arg(arg1) + "</ueIPv4Addre
                                      <ueIPv6Prefix>" + arg(arg2) + "</ueIPv6Prefix
                                      <apn>" + arg(arg3) + "</apn>
                                    </Message>
                                  </sap:removeSubscriberProfile>
                                </soapenv:Body>
                                </soapenv:Envelope>";
                                  }
                                </definition>
                              </EDSource>
                            </EDSources>
                          </EntityData>
                        </PolicyControlFunction>
                      </ManagedElement>
                    </config>
                  </edit-config>

```



Example 6 Configuration of Notification Rule Using SOAP EDSOURCE for removeSubscriberProfile operation

Once the SOAP operation with the notification message format is configured by means of the "SOAPNotification_deactivate" instance of the **EDSource**, configure the notification rule.

The attrValue of the outputAttributes in the rules URI in the Provisioning REST API contains the notification receiver identities and the notification text using the following syntax:

```
NotificationReceivers[<New eDSourceId>[<arg1, arg2, ..., argN>].xmlText]
```

[Example 6](#) shows how to configure the notification condition and to associate to it the BBSC removeSubscriberProfile operation (using SOAPNotification_deactivate **EDSource** instance).

```
PUT /rules/rSOAP_IpcanDeactivate
{
  "condition" : "(AccessData.bearer.requestType ==3) && (Subscription.group[\"ParentalControl\"].isActive)",
  "outputAttributes" :
  [
    {
      "attrName" : "notification",
      "attrValue" : "SOAPNotificationReceivers[SOAPNotification_deactivate[AccessData.subscriber.ueIpAddress, \"\"];xmlText]",
      "result" : "permit"
    }
  ],
  "ruleName" : "rSOAP_IpcanDeactivate"
}
```

This example sends a SOAP notification to notify for the IP-CAN session termination when the subscriber has the "ParentalControl" group active. The SAPC sends the SOAP notification to the web service identified by "BBSC_Notifications". The notification message is configured in the xmlText field definition of the "SOAPNotification_deactivate" **EDSource** object. The input arguments of the "SOAPNotification_deactivate" instance receive the needed values in order to fill the SOAP notification text properly.

5.1.3.3 Selecting Different BBSCs Depending on Network Topology

To select different BBSCs, add dynamic conditions such as EPG Origin-Host (AccessData.host.name) in the condition attribute for the rules URI in the Provisioning REST API. Configure a Web Service End Point for each BBSC and associate it with the suitable instance of the **EDSource** as a notification receiver. Afterwards, use these instances in the attrValue value of the outputAttributes attribute in the rules URI in the Provisioning REST API. The **EDSource** instances for the notifications text can be reused for the different destinations.



5.2 Performance Management

Refer to User Notifications.

5.3 Fault Management

Refer to User Notifications.

5.4 Logging

Refer to User Notifications.



6 Capabilities

Refer to User Notifications.



7 Appendix A. Policy Tags

The policy tags that can be used to trigger SOAP notifications from the SAPC to the BBSC, are the ones described in Configuration Guide for Access and Charging Control (Gx), and Configuration Guide for Subscription and Policies.



8 Reference List

Ericsson Documents

1. BBSC-PCRF Interface Description, 2/15519-AXB 250 18/5

Standards

1. Hypertext Transfer Protocol -- HTTP/1.1, 1999-06, RFC 2616
2. Simple Object Access Protocol (SOAP) 1.1, 2000-08, W3C SOAP/1.1 Note