Configuration Guide for Integration with OCS for Spending Limit Reporting (Sy)

Ericsson Service-Aware Policy Controller

USER GUIDE

21/1553-CSH 109 215/7 Uen B

Abstract

This document is a guideline to configure the SAPC node for interworking with Online Charging Systems (OCS).

Contents

1 Introduction

1.1 Document Purpose and Scope

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

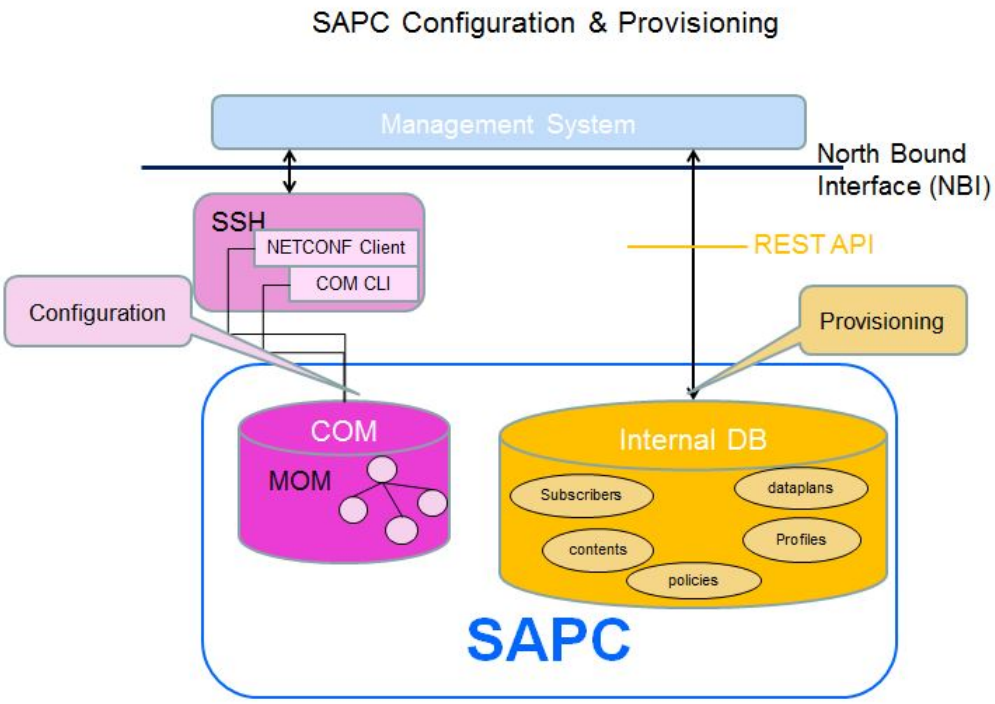

Figure 1 Configuration and Provisioning Overview

The purpose of this document is to provide a guideline to configure Online Charging System (OCS) related data in the SAPC node by providing some configuration examples.

This configuration is to be applied in addition to Gx configurations, so it is an extension of Configuration Guide for Access and Charging Control (Gx).

This document is not intended as an exhaustive guide to configure the SAPC for every possible scenario.

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.

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 Typographic Conventions

This document uses the following typographic conventions:

Table 1 Typographic Conventions

Convention	Description	Example
IMM	Diameter configuration	<imm:IMM-contents xmlns:imm="http://www.saforum.org/IMMSchema" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-in xsi:schemaLocation="http://www.saforum.org/IMM <object class="OtpdiaTransportTcp"> <dn>otpdiaTransportTcp=OcsWestMadrid-Sy,otpdiaService=PcrfSy <attr> <name>address</name> <value>10.41.30.22</value> </attr> <attr> <name>watchdogTimer</name> <value>6000</value> </attr> <attr> <name>reconnectTimer</name> <value>6000</value> </attr> <attr> <name>port</name> <value>0</value> </attr> <attr> <name>host</name> <value>:all</value> </attr> <attr> <name>connectTo</name> <value>otpdiaHost=OcsWestMadrid,otpdiaService=PcrfSy</valu </attr> </object> </imm:IMM-contents>
REST	SAPC Provisioning	**PUT /subscribers/34620000001 { "subscriberId" : "34620000001" }**

1.3 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 (dataplans), 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 /dataplans/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

2.1 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 Sy Network Data

3.1 Configure Sy Diameter Data

In addition to configure the SAPC as a Gx server, the SAPC can act as an Sy client. To do so, configure the data stated in this chapter.

To execute the Diameter configuration steps:

1. Log as sapcadmin into a System Controller (SC):

   ```
   ssh -X sapcadmin@<OAM VIP>
   ```

2. Create a .xml file with the content of the corresponding objects and attribute values (taking as templates the examples provided in this chapter)

3. Execute:

   ```
   immcfg -f <filename>.xml
   ```

Note: Origin-Host, Origin-Realm, IP address and diameter port values are set during the SAPC installation procedure. Diameter data related to capabilities exchange (application and vendor identifiers) are provided at installation time, so that no manual procedure is needed.

3.2 Configure Connections to OCS with One Realm

Direct Connection between the SAPC and the OCS

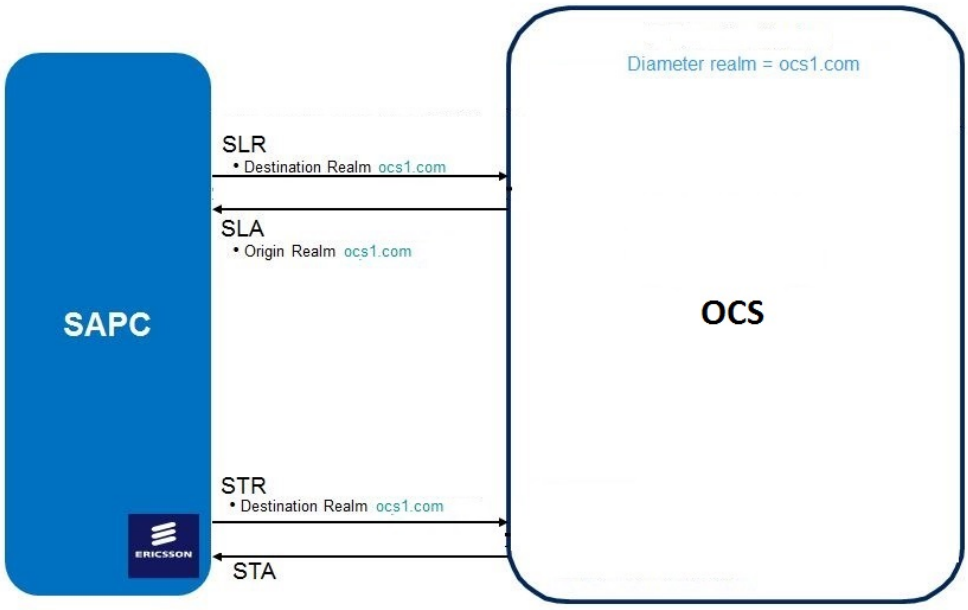

Figure 2 Direct connection to an OCS having only a realm

For each OCS:

1. Create an `OtpdiaHost` object under RDN `optdiaService=PcrfSy,otpdiaProduct=SAPC`

2. Enable the SAPC to act as Sy client creating either the `OtpdiaTransportTcp` or `OtpdiaTransportSctpE` object for all PLs using `:all` value in `host` attribute.

   Note: When the `:all` value in `host` attribute is used, that means that all the PLs will act as diameter clients for Sy, therefore, SAPC will open as many diameter connections (CER/CEA) as there are PLs in the SAPC cluster.

3. Proceed as Section 3.1 on page 7

```
<imm:IMM-contents xmlns:imm="http://www.saforum.org/IMMSchema"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.saforum.org/IMMSchema SAI-A
  <object class="OtpdiaHost">
    <dn>otpdiaHost=OCS_1,otpdiaService=PcrfSy,otpdiaProduct=SAPC</dn>
    <attr>
      <name>address</name>
      <value>192.168.14.42</value>
    </attr>
    <attr>
      <name>port</name>
      <value>13868</value>
   </attr>
  </object>
  <object class="OtpdiaTransportTcp">
    <dn>otpdiaTransportTcp=OCS_1_transport,otpdiaService=PcrfSy,otpdiaProduc
    <attr>
      <name>address</name>
      <value>192.168.12.40</value>
    </attr>
    <attr>
      <name>port</name>
      <value>0</value>
    </attr>
    <attr>
      <name>host</name>
      <value>:all</value>
    </attr>
    <attr>
     <name>connectTo</name>
      <value>otpdiaHost=OCS_1,otpdiaService=PcrfSy</value>
    </attr>
  </object>
</imm:IMM-contents>
```

Example 1 Configuration of the SAPC as Sy Diameter Client to direct OCS Peers

Example 1 configures the Sy traffic for an Online Charging System called "OCS_1" connected to the SAPC. In the `otpdiaTransportTcp` object, `address` and `port` attributes contain the local traffic IP and the port of the SAPC node (in this case, the IP is 192.168.12.40 and the local port `0`) whereas in the `otpdiaHost` object `address` and `port` attributes contain the IP and the port traffic for the OCS. The `connectTo` attribute in `otpdiaTransportTcp` object references the "OCS_1" OCS configured in `otpdiaHost`.

Note: In this case, the "OCS_1" realm is configured in `/profiles/online-charging-system/<profileId>` URI in the provisioning REST API. See Section 3.4 on page 18

Note: To get the IP of the SAPC execute `immlist -a address otpdiaTransportTcp=:all,otpdiaService=Pcrf,otpdiaProduct=SAPC` from the SC.

No Direct Connection between the SAPC and the OCS

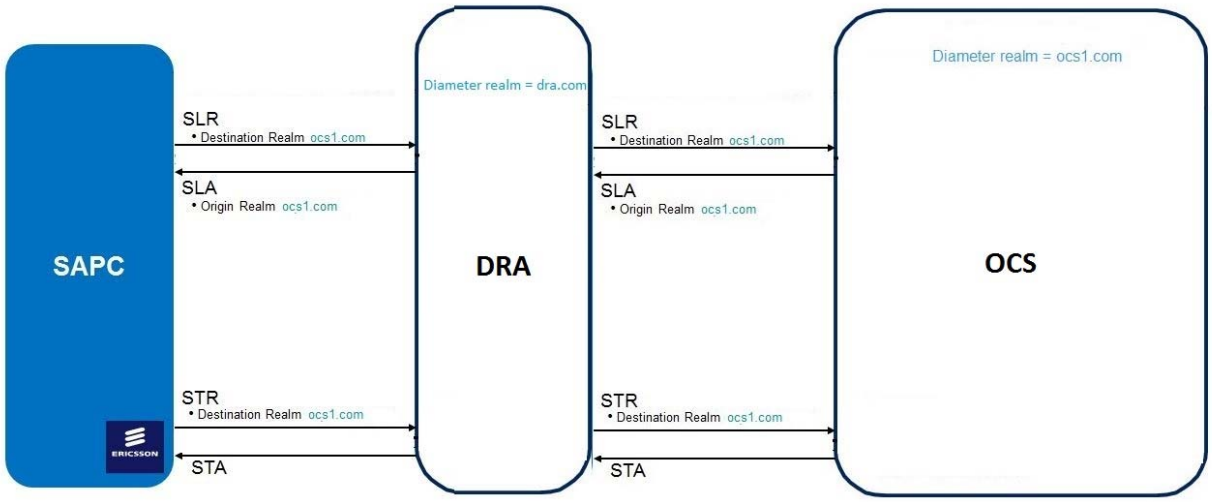

Figure 3 The SAPC not directly connected to the OCS.

To configure the Routing Table in the SAPC for not directly connected peers (see Example 2):

1. Configure the peer that is directly connected to the SAPC using `OtpdiaHost` and `OtpdiaTransportTcp` objects (or `OtpdiaTransportSctpE`) as described in Page 8

2. Define an `OtpdiaDomain` object with an identifier (in our example, `OCS`) to the destination OCS and realm value matching the one configured in the `serverRealm` attribute in the `/profiles/online-charging-system/<profileId>` URI in the provisioning REST API (See Section 3.4 on page 18)

3. Define the peer directly connected to the SAPC in an `OtpdiaDomain` object with an identifier (in our example, `DRA`). `Host` attribute must match with the value given in `otpdiaHost` configured in Step 1

4. Define a connection in an `OtpdiaCons` object for the `OtpdiaDomain` object that is directly connected to the SAPC (in our example, DRA). The distinguish name for the DRA in the `OtpdiaDomain` definition and the value for `head` attribute in the connection definition must match

5. To set a route (mapping of Destination-Host/Realm for an outgoing Request to a next-hop Origin-Host/Realm), create an `OtpdiaSelector` object.

6. Proceed as Section 3.1 on page 7

```
<imm:IMM-contents xmlns:imm="http://www.saforum.org/IMMSchema"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.saforum.org/IMMSchema SAI-A
<object class="OtpdiaHost">
  <dn>otpdiaHost=DRA,otpdiaService=PcrfSy,otpdiaProduct=SAPC</dn>
  <attr>
    <name>address</name>
    <value>192.168.14.42</value>
  </attr>
  <attr>
    <name>port</name>
    <value>13868</value>
 </attr>
</object>
<object class="OtpdiaTransportTcp">
  <dn>otpdiaTransportTcp=DRA_transport,otpdiaService=PcrfSy,otpdiaProduct=
  <attr>
    <name>address</name>
    <value>192.168.12.40</value>
  </attr>
  <attr>
    <name>port</name>
    <value>0</value>
  </attr>
  <attr>
    <name>host</name>
    <value>:all</value>
  </attr>
  <attr>
   <name>connectTo</name>
    <value>otpdiaHost=DRA,otpdiaService=PcrfSy</value>
  </attr>
</object>
<object class="OtpdiaDomain">
  <dn>otpdiaDomain=OCS,otpdiaProduct=SAPC</dn>
  <attr>
    <name>realm</name>
    <value>ocs1.com</value>
  </attr>
</object>
<object class="OtpdiaDomain">
  <dn>otpdiaDomain=DRA,otpdiaProduct=SAPC</dn>
  <attr>
    <name>host</name>
    <value>dra123.dra.com</value>
  </attr>
  <attr>
    <name>realm</name>
    <value>dra.com</value>
  </attr>
</object>
<object class="OtpdiaCons">
```

```
   <dn>otpdiaCons=Cons1,otpdiaProduct=SAPC</dn>
   <attr>
     <name>head</name>
     <value>otpdiaDomain=DRA,otpdiaProduct=SAPC</value>
   </attr>
  </object>
  <object class="OtpdiaSelector">
    <dn>otpdiaCons=Selector1,otpdiaProduct=SAPC</dn>
    <attr>
      <name>service</name>
      <value>otpdiaService=PcrfSy,otpdiaProduct=SAPC</value>
    </attr>
    <attr>
      <name>destination</name>
      <value>otpdiaDomain=OCS,otpdiaProduct=SAPC</value>
    </attr>
    <attr>
      <name>peer</name>
      <value>otpdiaCons=Cons1,otpdiaProduct=SAPC</value>
    </attr>
  </object>
</imm:IMM-contents>
```

Example 2 Sy configuration for not direct connection to OCS

Example 2 Configures a route to an OCS in realm ocs1.com through realm dra.com.

3.3 Configure Connections to an OCS with Several Diameter Realms

This is a network deployment in which a logical Charging System exposes different Diameter realms to the SAPC.

Figure 4 shows an example where the SAPC sends an SLR to ocs1.com realm, and the OCS internally decides that such Sy charging session has to be routed to sdp2.ocs1.com. Afterwards, the STR sent from the SAPC to the Charging System, has to reach the same sdp2.ocs1.com realm.

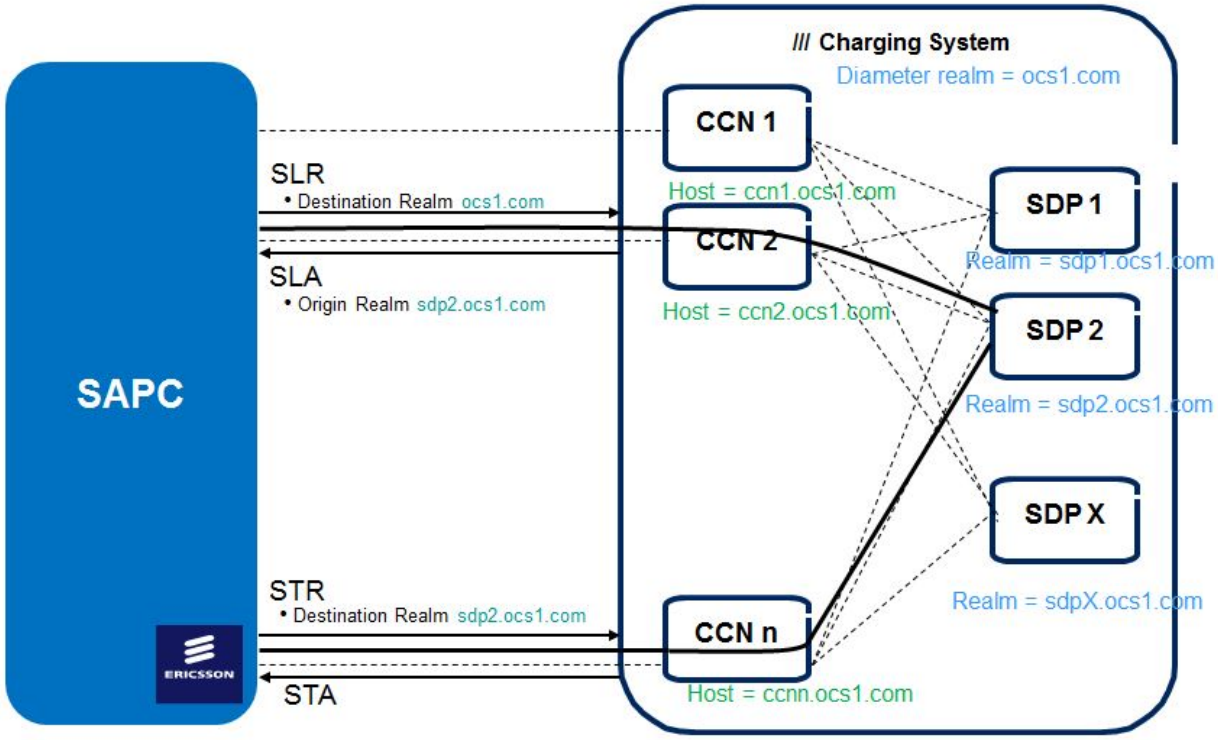

Figure 4 Several Sy Diameter Realms

1. Configure the peers that are connected directly to the SAPC using `OtpdiaHost` and `OtpdiaTransportTcp` objects (or `OtpdiaTransportSctpE`) as described in Page 8

2. Define one `OtpdiaDomain` for each node (peers and hosts) present in the topology that can receive/send a Sy message from/to the SAPC.

   Example 3. Defines an internal identifier (in our example,`OCS`) whose realm is the one declared in the SAPC internal database in `serverRealm` attribute in the `/profiles/online-charging-system/<profileId>` URI in the provisioning REST API (See Section 3.4 on page 18) and is not connected directly to the SAPC.

   Example 4. Defines an internal identifier (in our example, `CCN`) for each peer that is directly connected to the SAPC. `Host` attribute must match with the value given in `otpdiaHost` configured in Step 1

   Example 5. Defines an internal identifier (in our example, `SDP`) for each host that can send messages to the SAPC and are not connected directly.

3. Define a connection in an `OtpdiaCons` object just for each `OtpdiaDomain` object that is directly connected to the SAPC. In our example, the ones called CCNs.

This connection is identified by its distinguish name. The distinguish name for the CCN definition and the value for `head` attribute in the connection definition must match. SeeExample 6

4. To set a route, create an `OtpdiaSelector` object with a distinguish name for each host not directly connected. This object includes a `destination` attribute with the `otpdiaDomain` of the destination host defined inStep 2 , and a `peer` attribute with the `OtpdiaCons` connection to the peer defined inStep 3. See Example 7. The route states that to get OCS and SDP1, Sy message is sent to Cons1 (CCN1) as first step

```
<imm:IMM-contents xmlns:imm="http://www.saforum.org/IMMSchema"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.saforum.org/IMMSchema SAI-AIS-
<object class="OtpdiaDomain">
    <dn>otpdiaDomain=OCS,otpdiaProduct=SAPC</dn>
    <attr>
      <name>realm</name>
      <value>ocs1.com</value>
    </attr>
  </object>
</imm:IMM-contents>
```

Example 3 Definition of OCS realm

```
<imm:IMM-contents xmlns:imm="http://www.saforum.org/IMMSchema"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.saforum.org/IMMSchema SAI-AIS-
 <object class="OtpdiaDomain">
    <dn>otpdiaDomain=CCN1,otpdiaProduct=SAPC</dn>
    <attr>
      <name>host</name>
      <value>ccn1.ocs1.com</value>
    </attr>
    <attr>
      <name>realm</name>
      <value>ocs1.com</value>
    </attr>
 </object>
</imm:IMM-contents>
```

Example 4 Definition of CCNs

```
<imm:IMM-contents xmlns:imm="http://www.saforum.org/IMMSchema"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.saforum.org/IMMSchema SAI-A
 <object class="OtpdiaDomain">
    <dn>otpdiaDomain=SDP1,otpdiaProduct=SAPC</dn>
    <attr>
      <name>host</name>
      <value>sdp1.ocs1.com</value>
    </attr>
    <attr>
      <name>realm</name>
      <value>ocs1.com</value>
    </attr>
 </object>
</imm:IMM-contents>
```

Example 5 Definition of SDPs

```
<imm:IMM-contents xmlns:imm="http://www.saforum.org/IMMSchema"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.saforum.org/IMMSchema SAI-A
<object class="OtpdiaCons">
 <dn>otpdiaCons=Cons1,otpdiaProduct=SAPC</dn>
 <attr>
       <name>head</name>
       <value>otpdiaDomain=CCN1,otpdiaProduct=SAPC</value>
 </attr>
 </object>
</imm:IMM-contents>
```

Example 6 Definition of Connections

```
<imm:IMM-contents xmlns:imm="http://www.saforum.org/IMMSchema"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.saforum.org/IMMSchema SAI-AIS-
<object class="OtpdiaSelector">
    <dn>otpdiaCons=Selector1,otpdiaProduct=SAPC</dn>
    <attr>
      <name>service</name>
      <value>otpdiaService=PcrfSy,otpdiaProduct=SAPC</value>
    </attr>
    <attr>
      <name>destination</name>
      <value>otpdiaDomain=OCS,otpdiaProduct=SAPC</value>
    </attr>
    <attr>
      <name>peer</name>
      <value>otpdiaCons=Cons1,otpdiaProduct=SAPC</value>
    </attr>
  </object>

  <object class="OtpdiaSelector">
    <dn>otpdiaCons=Selector2,otpdiaProduct=SAPC</dn>
    <attr>
      <name>service</name>
      <value>otpdiaService=PcrfSy,otpdiaProduct=SAPC</value>
    </attr>
    <attr>
      <name>destination</name>
      <value>otpdiaDomain=SDP1,otpdiaProduct=SAPC</value>
    </attr>
    <attr>
      <name>peer</name>
      <value>otpdiaCons=Cons1,otpdiaProduct=SAPC</value>
    </attr>
  </object>

</imm:IMM-contents>
```

Example 7 Definition of Topology

Script to Automate Configuration Values

Preparing the configuration for the Diameter Routing Table in a deployment asFigure 4 with n CCNs and x SDPs involves the declaration of many objects and can lead to errors. To ease complex configurations, the SAPC provides a script to generate the XML files. To run this script:

1. Log as sapcadmin into a System Controller (SC):

   ssh -X sapcadmin@<OAM VIP>

2. Generate an XML file named input_file.txt asExample 8 with the definition of attributes related to each `OCS`, `CCN`, and `SDP`

3. Run the following command:

   ```
   generateRoutingTableOCS  --config=input_file.txt --populate
   ```

   Note: The option `--populate` applies the output XML files to the SAPC directly

```
<config>
  <ocs>
    <otpdiaDomain>OCS</otpdiaDomain>
    <realm>ocsrealm.com</realm>
  </ocs>
  <ccns>
     <ccn>
         <otpdiaDomain>CCN1</otpdiaDomain>
         <host>ccn1.realm0</host>
         <realm>realm0</realm>
         <port>13222</port>
         <address>172.16.2.21</address>
     </ccn>
     <ccn>
         <otpdiaDomain>CCN2</otpdiaDomain>
         <host>ccn2.realm0</host>
         <realm>realm0</realm>
         <port>13223</port>
         <address>172.16.2.21</address>
     </ccn>
  </ccns>
  <sdps>
     <sdp>
         <otpdiaDomain>SDP1</otpdiaDomain>
         <host>sdp1.realm1</host>
         <realm>realm1</realm>
     </sdp>
     <sdp>
         <otpdiaDomain>SDP2</otpdiaDomain>
         <host>sdp2.realm2</host>
         <realm>realm2</realm>
     </sdp>
     <sdp>
         <otpdiaDomain>SDP3</otpdiaDomain>
         <host>sdp3.realm3</host>
         <realm>realm3</realm>
     </sdp>
     <sdp>
         <otpdiaDomain>SDP4</otpdiaDomain>
         <host>sdp4.realm4</host>
         <realm>realm4</realm>
```

```
    </sdp>
    <sdp>
        <otpdiaDomain>SDP5</otpdiaDomain>
        <host>sdp5.realm5</host>
        <realm>realm5</realm>
    </sdp>
    </sdps>
</config>
```

Example 8 XML example for script input

3.4 Configure Online Charging Systems

For each Online Charging System the SAPC needs to communicate with, provision the charging profile in the `/profiles/online-charging-system/<profileId>` URI in the provisioning REST API with the following attributes:

— Set `profileId` attribute with a charging profile name

— Set `serverRealm` attribute to the realm of the Online Charging System

— Set `syVersion` attribute to specify the Sy interface version to use:

- Value `0`: 3GPP standard Sy interface.
- Value 1: Ericsson Sy interface.

An example to configure an Online Charging System is the following:

```
PUT /profiles/online-charging-system/OcsWestMadrid

{
 "profileId" : "OcsWestMadrid",
 "serverRealm" : "ocs3gpprealm.com",
 "syVersion" : 0
}
```

Example 9 Configuration of Charging Systems

This example configures an Online Charging System with values “OcsWestMadrid” as `profileId`, “ocs3gpprealm.com” as `serverRealm` and "0" for `syVersion` (3GPP standard Sy interface).

4 Provision of Subscribers

4.1 Subscribers Provisioned Only in Charging Systems

Subscribers can be provisioned only in Charging Systems without the need of provisioning corresponding resources in the SAPC database. In this case, the following configurations (for the SAPC to initiate communication towards a Charging System) apply:

a Policies for dynamic Charging System selection as described in Section 5.3 on page 22

b Or the default configuration as described in Section 5.4 on page 25.

Note: If the SAPC obtains the subscription from the Online Charging System, it makes no sense to use Unknown subscriber. If it is not obtained any Policy Group or Policy Counter for the Subscriber from the OCS, the SAPC can apply Unknown subscriber (if provisioned).

Note: This scenario only applies to deployments with Ericsson Sy.

4.2 Subscribers Provisioned in the SAPC and in the Charging System

In this case, the resulting Subscriber profile is a combination of the Subscriber data provisioned in the SAPC internal database (`subscribers` URI in the provisioning REST API) and the subscription information (Policy Groups and activation/deactivation time) obtained from Charging System.

The mapping between the Policy Groups obtained from Ericsson Sy interface and the SAPC Subscriber Group is needed in case it is needed to associate some static qualification data that characterize such group.

In this case, the following configurations (for the SAPC to initiate communication towards a Charging System) apply:

a Provisioning Charging System at Subscriber profile described in Section 5.1 on page 21

Note: This option does not apply in case Autoprovisioning (refer to Configuration Guide for Subscription and Policies) is enabled in the SAPC.

b Policies for dynamic Charging System selection as described in Section 5.3 on page 22

c Or the default configuration as described in Section 5.4 on page 25

5 Configure Selection of Charging System

This chapter explains the different ways to configure the selection of Charging System in the SAPC.

5.1 Provisioning at Subscriber Profile

To assign statically a Charging System to a subscriber do:

— Add a value that corresponds to the `profileId` attribute in the`/profiles/online-charging-system/<profileId>` URI in the provisioning REST API to the `onlineChargingSystemProfileId` JSON attribute inside the `staticQualification` attribute in the `subscriber` URI in the provisioning REST API.

The following example shows how to associate a particular Charging System for a single subscriber:

```
PUT /subscribers/34600320101

{
 "dataplans" :
 [

  {
   "dataplanName" : "MobileBroadband"
  }
 ],
 "staticQualification" :
 {
  "onlineChargingSystemProfileId" : "OcsWestMadrid"
 },
 "subscriberId" : "34600320101"
}


PUT /dataplans/MobileBroadband

{
        "dataplanName" : "MobileBroadband"
}
```

Example 10 Provisioning of Subscriber

The example above associates “OcsWestMadrid” profile to the subscriber “34600320101”, which belongs to “MobileBroadband “ group.

5.2 Avoiding Sy Interface Communication for Subscribers Provisioned in the SAPC

To avoid communication with any Charging System for a subscriber provisioned in the SAPC, use a <charging system name identifier> in the `onlineChargingSystemProfileId` JSON attribute inside the `staticQualification` attribute in the `subscribers` URI in the provisioning REST API that does not correspond to any existing entry of the `/profiles/online-charging-system/<profileId>` URI in the provisioning REST API.

```
PUT /subscribers/34600320102

{
 "staticQualification" :
 {
  "onlineChargingSystemProfileId" : "NoOcs"
 },
 "subscriberId" : "34600320102"
}
```

Example 11 Provisioning of Subscriber

The example above assigns “NoOcs” to the subscriber “34600320102”. As “NoOcs” does not exist in `/profiles/online-charging-system/<profileId>` URI in the provisioning REST API, the SAPC does not establish communication with any Charging System for this subscriber.

5.3 Configure Policies for Dynamic Charging System Selection

It is possible to use flexible conditions to assign a Charging System to Subscribers in the SAPC, by using Charging System policies.

To configure policies for Charging System selection, configure the following objects:

Policy Type	Policy Locator			Output Attributes
	Context	Resource	Subject	
Charging System	charging-system 	any	-	permit **charging-system** <chargingSystemName>

Figure 5 Charging System Policy Type

— For Global policy locator, use the following REST URI: `/locators/resources/any/contexts/charging-system`

— Within the `outputAttributes` attribute in the `/rules/<ruleId>` URI in the provisioning REST API set:

  - `attrName` attribute to `charging-system`.
  - `attrValue` with the alias used in `/profiles/online-charging-system` f or this OCS, for example "OCS_1"

Figure 6 shows an example of selection of OCS using policies.

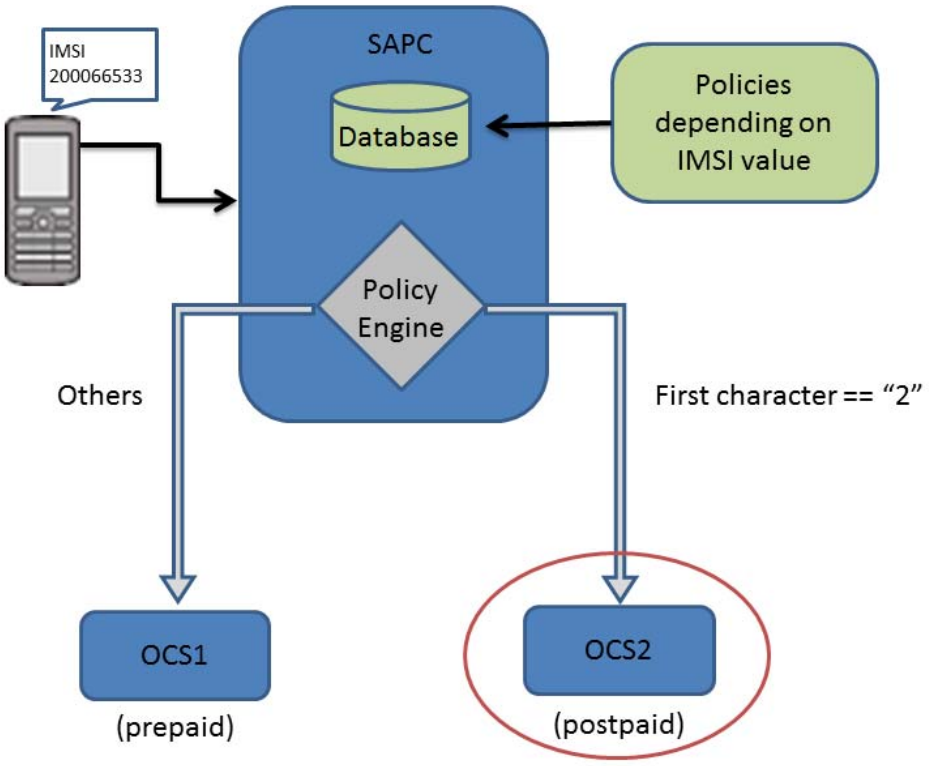

Figure 6 Configuration for Charging System selection based on IMSI

Figure 6 shows an example for the SAPC configuration for the following case: a postpaid Charging System has to be selected for subscribers whose first IMSI character matches certain number; and other Charging System applies for the rest of subscribers.

Next example completes the configuration for Figure 6

```
PUT /rules/r_0

{
```

```
 "condition" : "(firstCharsOf(AccessData.subscriber.imsi, 1) == \"2\")",
 "outputAttributes" :
 [

  {
   "attrName" : "charging-system",
   "attrValue" : "\"OCS_postPaid\"",
   "result" : "permit"
  }
 ],
 "ruleName" : "r_0"
}


PUT /rules/r_1

{
 "condition" : "(firstCharsOf(AccessData.subscriber.imsi, 1) != \"2\")",
 "outputAttributes" :
 [

  {
   "attrName" : "charging-system",
   "attrValue" : "\"OCS_prePaid\"",
   "result" : "permit"
  }
 ],
 "ruleName" : "r_1"
}


PUT /policies/pGlobal

{
 "policyName" : "pGlobal",
 "ruleCombiningAlgorithm" : "permit-overrides",
 "rules" : [ "r_0", "r_1" ]
}


PUT /locators/resources/any/contexts/charging-system

{
 "policies" : [ "pGlobal" ]
}

PUT /profiles/online-charging-system/OCS_postPaid

{
 "profileId" : "OCS_postPaid",
 "serverRealm" : "ocsrealm.com",
```

```
 "syVersion" : 1
}


PUT /profiles/online-charging-system/OCS_prePaid

{
 "profileId" : "OCS_prePaid",
 "serverRealm" : "newocsrealm.com",
 "syVersion" : 1
}
```

Example 12 Configuration of default Charging System

In addition, to select "default" chargingSystem as the prepaid OCS, seeExample 13

5.4 Configure Default Charging System

To configure a default entry that applies to all subscribers without any static Online Charging System qualification, configure the `profileId` attribute with the “default” value in the `/profiles/online-charging-system` URI in the provisioning REST API as follows:

```
PUT /profiles/online-charging-system/default

{
 "profileId" : "default",
 "serverRealm" : "newocsrealm.com",
 "syVersion" : 1
}
```

Example 13 Configuration of default Charging System

This example provisions a charging profile in the `/profiles/online-cha rging-system` URI in the provisioning REST API with “default” profileId, `serverRealm` attribute set to “newocsrealm.com” value and `syVersion` attribute set to 1 (Ericsson Sy interface).

Default Sy Interface Communication for Subscribers Provisioned in the SAPC

If there is a "default" profile defined in `/profiles/online-charging-system` URI in the provisioning REST API, a subscriber provisioned in the SAPC without any profile in the `onlineChargingSystemProfileId` JSON attribute inside the `staticQualification` attribute in the `subscribers` URI in the provisioning REST API is assigned to that "default" OCS.

6 Configure Policy Control Based on Spending Limits Reporting

The SAPC can deny the access to a service, change the bandwidth, apply different ratings, and so on, depending on the subscriber monetary balance state, by using operator configured policies.

To configure policies for Spending Limits Reporting as any other policy in the SAPC, refer to Configuration Guide for Subscription and Policies and Provisioning REST API.

Note: In this scenario, **accumulation** policy type does not apply.

A typical case for a condition based on Spending Limits Reporting counter state to authorize or not a Service is shown in the following example:

```
PUT /rules/Auth_FB

{
 "condition" : "(SubsCharging.state[\"volume\"] != \"80ThresholdReached\")",
 "ruleName" : "Auth_FB"
}


PUT /policies/Auth_FB

{
 "policyName" : "Auth_FB",
 "ruleCombiningAlgorithm" : "permit-overrides",
 "rules" : [ "Auth_FB" ]
}


PUT /locators/resources/Facebook/contexts/access

{
 "policies" : [ "Auth_FB" ]
}
```

Example 14 Configuration for Service Authorization based on Spending Limits

The previous example shows a policy for Authorization of Facebook Service. The Facebook Service is authorized when the state for ‘‘volume’’ counter received from the Online Charging System is different from‘‘80ThresholdReached’’.

7 Appendix A. Policy Tags

Table 2 includes the policy tags that can be used to configure policies based on Spending Limit Reporting. These tags only apply to **Gx**.

Table 2 Integration with Online Charging System for Spending Limit Reporting Related Tag

Tag	Return Type	Possible Values	Comments
`SubsCharging.state["identifier"]`	String	any	The policy counter state obtained from the Charging System for the indicated Policy Counter. • For 3GPP standard Sy interface: obtained from `Policy-Counter-Status` AVP. • For Ericsson Sy interface(1): obtained from `Ericsson-Policy-Counter-Status`
`SubsCharging.group["groupName"].state["identifier"]`(2)	String	any	The policy counter state obtained from an Ericsson Charging System for the indicated Policy Counter and Policy Group.

(1) Only use this tag if the counter identifier is unique among different Policy Groups. Otherwise, use SubsCharging.group["groupName"].state["identifier"].
(2) Only applicable for Ericsson Sy interface.

Glossary

CBA
Component Based Architecture

IMSI
International Mobile Subscriber Identity

MOC
Managed Object Class

OCS
Online Charging System

PL
Payload Processor

RRT
Realm Routing Table

SAPC
Ericsson Service-Aware Policy Controller

Reference List

Ericsson Documents

[1] Configuration Guide for Access and Charging Control (Gx)

[2] Configuration Guide for Subscription and Policies

[3] Provisioning REST API