

# Flexible Output Protocol

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 function description of flexible output protocol and guideline to configure the Ericsson Service-Aware Policy Controller (SAPC) for this function.



# Contents

<b>1</b>	<b>Flexible Output Protocol Function</b>	<b>1</b>
1.1	Flexible Output Protocol Overview	1
1.2	Flexible Output Policies	2
1.3	Message Transformation Orders	2
1.4	Supported Protocols	3
1.5	Flexible Output Dictionary	3
<b>2</b>	<b>Flexible Protocol Configuration</b>	<b>5</b>
2.1	Provision Flexible Output Policies	5
2.2	Configure Message Transformation Orders	7
2.3	Configure Flexible Output Dictionary Syntax	15
2.4	Configuration Examples for Flexible Output Use Cases	17
<b>3</b>	<b>Flexible Output Protocol Language Syntax</b>	<b>25</b>
3.1	Configure AVP Values of OctetString Type	27
3.2	Flexible Output Protocol Message Transformation Orders	28
<b>4</b>	<b>Transformation and Matching Criteria Specification</b>	<b>37</b>
<b>5</b>	<b>Appendix A. Flexible Output Policy Types</b>	<b>39</b>





# 1 Flexible Output Protocol Function

## 1.1 Flexible Output Protocol Overview

The SAPC supports several standard protocol interfaces. Functions and advantages offered by standards are well-known, allowing inter-operation among different vendor nodes. However, there are also non-standard Vendor Specific needs.

Flexible output protocol allows the use of non-standard data inside outgoing protocol messages from the SAPC.

The availability of this function in the SAPC is under license control.

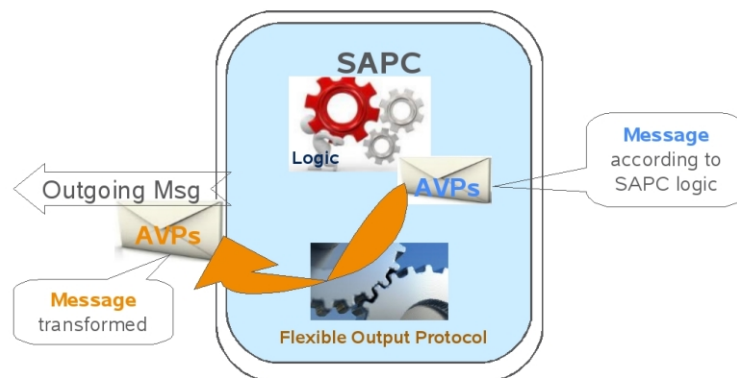


Figure 1 Flexible Output Protocol Functionality Overview

This functionality executes transformations of the outgoing protocol messages that do not affect the SAPC logic, but that are complementary to it. This function is achieved with configuration, and executed after the SAPC processes incoming messages and performs the corresponding business logic.

The configurable components in flexible output protocol are the following:

- Flexible Output policies (Section 1.2 on page 2).
- Message transformation orders (Section 1.3 on page 2).
- Flexible output dictionary (Section 1.5 on page 3).

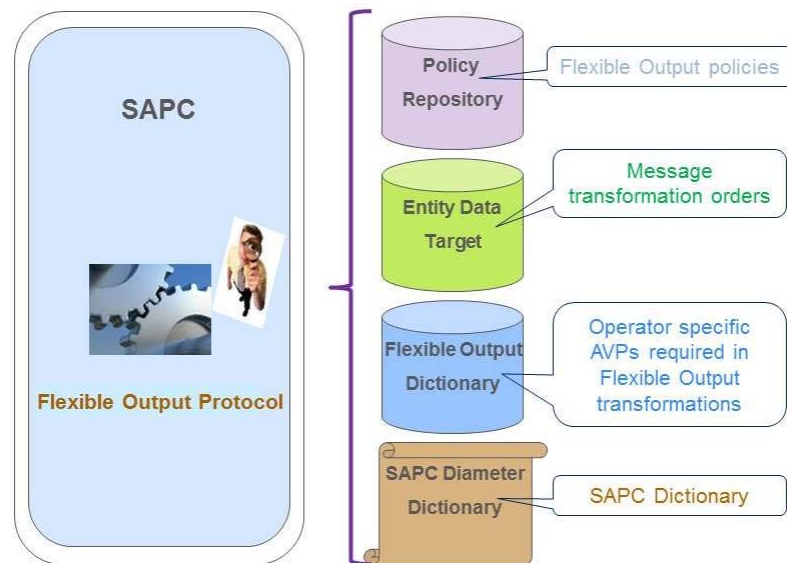


Figure 2 Configurable Elements In flexible output protocol

## 1.2 Flexible Output Policies

The SAPC applies flexible output protocol depending on the evaluation of dynamic conditions, configured with flexible output policies.

For example in Gx, these policies are evaluated just before sending CCA or RAR messages.

Depending on the kind of the outgoing message to modify, there are the following levels of flexible output policies:

- Policies **applicable to command level**.
- Policies **applicable to modify AVPs at service level**.

Flexible output protocol with policies configured at **service level** is service qualification data. Therefore, any change in the set of transformation orders applied at **service level** for the same service causes its reauthorization.

The SAPC evaluates policies configured at **service level** whenever the corresponding service is authorized during the IP-CAN session reauthorization.

## 1.3 Message Transformation Orders

Transformation orders modify the contained AVPs in outgoing messages and are configured through **Entity Data Target**, using a particular syntax format (see Section 4 on page 37) which, includes different types of operations internally interpreted according to certain semantic rules.



The transformations that the SAPC can apply are:

- **Add** AVPs (indicating their values).
- **Delete** AVPs that result from internal the SAPC business logic.
- **Replace** AVPs value that result from internal SAPC business logic

To see the types of supported transformations, see Table 2.

## 1.4 Supported Protocols

Flexible output protocol in the SAPC can be applied to the following protocols:

- Gx, DIAMETER (CCA, RAR messages).

For specific details, see Page 8.

## 1.5 Flexible Output Dictionary

The SAPC provides at installation time a default dictionary with some Diameter data for the various diameter interfaces that supports. Flexible output protocol use flexible output dictionary to apply transformation orders in the AVPs that are not included in the default dictionary.

The AVPs defined in flexible output dictionary are used from **message transformation orders** to have the particular details of the AVPs content before sending them into the network. For each particular AVP stated in a EDTarget, and not present in the SAPC dictionary, there must be its corresponding entry in flexible output dictionary.

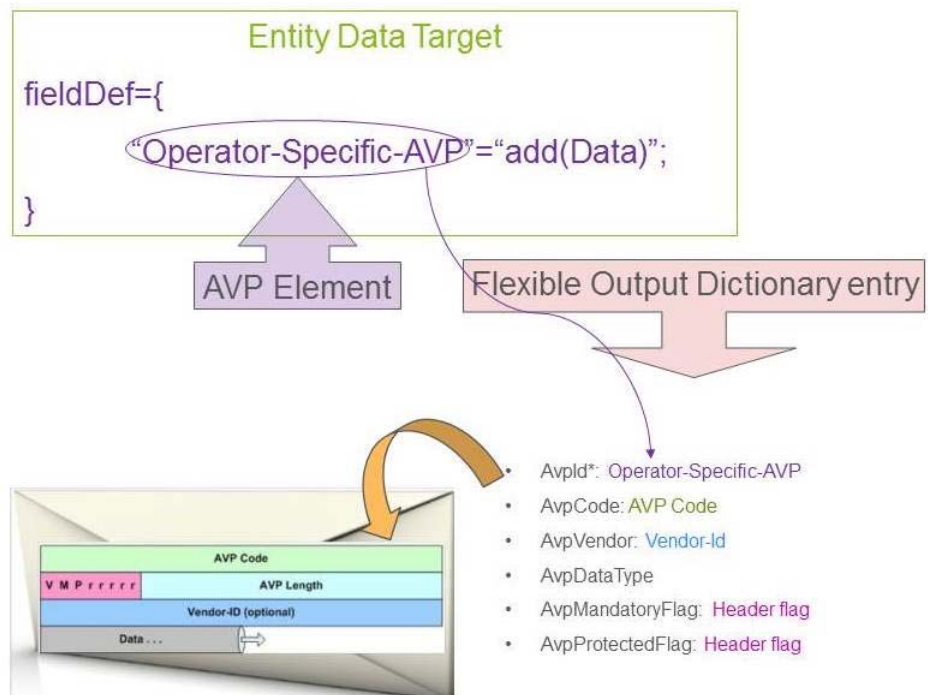


Figure 3 Relationship between Entity Data Target Attributes and Flexible Output Dictionary





## 2 Flexible Protocol Configuration

To configure flexible output protocol, follow the next steps:

1. Provision flexible output policies. See Section 2.1 on page 5.
2. Configure entity data targets. See Section 2.2 on page 7.
3. Configure diameter dictionary syntax. See Section 2.3 on page 14.

### 2.1 Provision Flexible Output Policies

To configure flexible output policies follow the next steps:

1. Provision at least one rule (condition) and policy associated to establish the scope where flexible output protocol is going to be applied.

— For **global policy locator**:

`/locators/resources/any/contexts/flexible-output`, to modify AVPs at command level.

`/locators/resources/<contentName>/contexts/flexible-output`, to modify AVPs at service level.

— For **Subscriber group locator**

`/dataplanes/<dataplanName>/locators/resources/any/contexts/flexible-output`, to modify AVPs at command level.

`/dataplanes/<dataplanName>/locators/resources/<contentName>/contexts/flexible-output`, to modify AVPs at service level.

— For **Subscriber locator**

`/subscribers/<subscriberId>/locators/resources/any/contexts/flexible-output`, to modify AVPs at command level.

`/subscribers/<subscriberId>/locators/resources/<contentName>/contexts/flexible-output`, to modify AVPs at service level.

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

- `attrName` attribute to `flexible-output`
- `attrValue` to `edt.<name_edt>(args)`, where `name_edt` is the name of an instance of `EDTarget MOC`.

**Note:** Time of day conditions within these policies does not perform time-based reauthorization for them.



### 2.1.1 Command Level

An example of policies configuration at **command level**, associated to a subscriber group, follows:

```
PUT /rules/rFlexibleOutput
{
  "condition" : "(AccessData.host.name == \"ggsnNodeHostname.nodeHostRealm.com\"
  "outputAttributes" :
  [
    {
      "result" : "permit",
      "attrName" : "flexible-output",
      "attrValue" : "edt.AddOperatorSpecificAVP()"
    }
  ],
  "ruleName" : "rFlexibleOutput"
}

PUT /policies/pFlexibleOutput
{
  "policyName" : "pFlexibleOutput",
  "ruleCombiningAlgorithm" : "permit-overrides",
  "rules" : [ "rFlexibleOutput" ]
}

PUT /dataplanes/Silver/locators/resources/any/contexts/flexible-output
{
  "policies" : [ "pFlexibleOutput" ]
}
```

#### Example 1 Configuration of Flexible Output Protocol policies

As a result of the previous configuration example, the SAPC applies flexible output protocol to all the outgoing messages related to Silver group for the Diameter peer whose Origin-Host is “ggsnNodeHostname.nodeHostRealm.com”.

The output attribute (with name “flexible-output”) value is an EDTTarget where the message transformation order is configured. See Section 2.2 on page 7.

### 2.1.2 Service Level

Next example shows policies configuration at **service level**, associated to a subscriber group:



```

PUT /rules/rFlexibleOutputService
{
  "condition" : "(AccessData.host.name == \"ggsnNodeHostname.nodeHostRealm.com\")",
  "outputAttributes" :
  [
    {
      "result" : "permit",
      "attrName" : "flexible-output",
      "attrValue" : "edt.ModifyPCCRule()"
    }
  ],
  "ruleName" : "rFlexibleOutputService"
}

PUT /policies/pFlexibleOutputService
{
  "policyName" : "pFlexibleOutputService",
  "ruleCombiningAlgorithm" : "permit-overrides",
  "rules": [ "rFlexibleOutputService" ]
}

PUT /dataplanes/Silver/locators/resources/Internet/contexts/flexible-output
{
  "policies" : [ "pFlexibleOutputService" ]
}

```

### Example 2 Configuration of Flexible Output Protocol policies

In this case, the SAPC applies flexible output protocol whenever “Internet” service is authorized for group “Gold” for the PCEF whose Origin-Host is “ggsnNodeHostname.nodeHostRealm.com”. Then, the outgoing message is modified according to message transformation order.

## 2.2 Configure Message Transformation Orders

To be able to use message transformation, use EDTarget containing a URL pointing to diameter outgoing message. The message transformation orders are configured within its fieldDef attribute.

---

---

### Do!

For command transformation orders affecting AVPs included into Grouped AVPs, specify the whole AVP structure.

---

---



---

---

## Warning!

Using flexible output protocol to modify AVPs that have associated a functional standard procedure is not recommended at all. The SAPC is not aware of this kind of logic modifications, causing unpredictable behaviors.

---

---

To build properly EDTarget for url set to DiameterOutgoingMessage, follow flexible output language syntax. For further information, see Section 3 on page 25.

### Applying Flexible Output Protocol to a Specific Protocol

To apply flexible output protocol to a specific protocol, use the Auth-Application-Id AVP set to the proper value inside the matching pattern attribute within the EDTarget.

Example 3 shows a message transformation order only for Gx protocol:



```

<edit-config>
  <target>
    <running />
  </target>
</config>
<ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
  <managedElementId>1</managedElementId>
  <dnPrefix>dc=ManagedElement</dnPrefix>
  <networkManagedElementId>1</networkManagedElementId>
  <userLabel>Managed Element</userLabel>
  <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
    <policyControlFunctionId>1</policyControlFunctionId>
    <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
      <entityDataId>1</entityDataId>
      <EDTargets xmlns="urn:com:ericsson:ecim:edtargetsmom">
        <eDTargetsId>1</eDTargetsId>
        <EDTarget xmlns="urn:com:ericsson:ecim:edtargetmom" xmlns:
          <eDTargetId>ModifyGx</eDTargetId>
          <definition>
            def ModifyGx() {
              dataTarget = {
                url = "diameterOutgoingMessage";
                matchingPattern = "(Auth-Application-Id([16777
              }
              fieldDef = {
                Charging-Rule-Install.Operator-Specific-AVP1 =
              }
            }
          </definition>
        </EDTarget>
      </EDTargets>
    </EntityData>
  </PolicyControlFunction>
</ManagedElement>
</config>
</edit-config>

```

### Example 3 Configuration of EDTarget for Gx Outgoing Messages

Example 3 adds Operator-Specific-AVP1 AVP with value "1234" for Gx CCA messages.

### Adding an AVP at Command Level

Example 4 shows the content of a EDTarget configuration, for the previously shown Example 1:



```

<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDTargets xmlns="urn:com:ericsson:ecim:edtargetsmom">
            <eDTargetsId>1</eDTargetsId>
            <EDTarget xmlns="urn:com:ericsson:ecim:edtargetmom" xmlns:nc="urn:com:ericsson:nc">
              <eDTargetId>AddOperatorSpecificAVP</eDTargetId>
              <definition>
                def AddOperatorSpecificAVP() {
                  dataTarget = {
                    url = "diameterOutgoingMessage:RA-Request";
                    matchingPattern = "Auth-Application-Id[16777238]";
                  }
                  fieldDef = {
                    Operator-Specific-AVP1 = add(1234);
                  }
                }
              </definition>
            </EDTarget>
          </EDTargets>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>

```

#### Example 4 Configuration of EDTarget for Reauthorization Request

Example 4 adds Operator-Specific-AVP1 AVP with value "1234", for RAR messages.

#### Transformation for Preconfigured PCC Rules

Example 5 shows a EDTarget configuration for previously shown Example 2, for preconfigured PCC rules:



```

<edit-config>
  <target>
    <running />
  </target>
</config>
  <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
    <managedElementId>1</managedElementId>
    <dnPrefix>dc=ManagedElement</dnPrefix>
    <networkManagedElementId>1</networkManagedElementId>
    <userLabel>Managed Element</userLabel>
    <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
      <policyControlFunctionId>1</policyControlFunctionId>
      <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
        <entityDataId>1</entityDataId>
        <EDTargets xmlns="urn:com:ericsson:ecim:edtargetsmom">
          <eDTargetsId>1</eDTargetsId>
          <EDTarget xmlns="urn:com:ericsson:ecim:edtargetmom" xmlns:
            <eDTargetId>ModifyPCCRule</eDTargetId>
            <definition>
              def ModifyPCCRule() {
                dataTarget = {
                  url = "diameterOutgoingMessage";
                }
                fieldDef =
                {
                  Charging-Rule-Install.Charging-Rule-Definition
                  Charging-Rule-Install.Charging-Rule-Definition
                }
              }
            </definition>
          </EDTarget>
        </EDTargets>
      </EntityData>
    </PolicyControlFunction>
  </ManagedElement>
</config>
</edit-config>

```

#### Example 5 Configuration of EDTarget for service outgoing messages modification, for preconfigured PCC rules

In the Example 5, the SAPC adds Operator-Specific-AVP2 AVP, inside Charging-Rule-Definition with value "internet\_skype" when Charging-Rule-Name is "1002" for both RAR and CCA messages. The SAPC removes Flow-Description AVP, inside Charging-Rule-Definition, when Charging-Rule-Name is "1002".

Flexible output protocol EDTarget with policies configured at service level are service qualification data. Therefore, any change in the set of transformation orders applied at service level for the same service causes its reauthorization.



The SAPC evaluates policies configured at service level whenever the corresponding service is authorized during the IP-CAN session reauthorization.

### Transformation for Dynamic PCC Rules

Example 6 shows an EDTarget configuration for the previously shown Example 2, for dynamic PCC rules:

```
<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDTargets xmlns="urn:com:ericsson:ecim:edtargetsmom">
            <eDTargetsId>1</eDTargetsId>
            <EDTarget xmlns="urn:com:ericsson:ecim:edtargetmom" xmlns:nc="urn:com:ericsson:nc">
              <eDTargetId>ModifyDynPCCRule</eDTargetId>
              <definition>
                def ModifyDynPCCRule() {
                  dataTarget = {
                    url = "diameterOutgoingMessage";
                  }
                  fieldDef =
                  {
                    Charging-Rule-Install.Charging-Rule-Definition[QoS]
                  }
                }
              </definition>
            </EDTarget>
          </EDTargets>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>
```

Example 6 Configuration of EDTarget for service outgoing messages modification, for dynamic PCC rules





In the Example 6, the SAPC adds Operator-Specific-AVP2 AVP with value "sponsored\_video" whenever QoS-Class-Identifier is 5 and Rating-Group is 2, for both RAR and CCA messages.

The Charging-Rule-Name cannot be used as a condition for dynamic PCC rules (because the SAPC automatically generates it). So, the EDTarget must contain a condition (as specific as possible) that makes transformation order applicable only for the desired dynamic PCC rules.

Flexible output protocol applies transformation orders regardless the way the services have been classified by the SAPC, that means that if two different media or submedia components are classified under the same SAPC dynamic service the same transformation orders apply to both media or submedia components.

### **Transformation Containing Dynamic Content for AVP Values**

Example 7 shows a transformation order profile using **dynamic content** for AVP values (use of Condition Policy Language expressions).



```

<edit-config>
  <target>
    <running />
  </target>
</config>
  <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
    <managedElementId>1</managedElementId>
    <dnPrefix>dc=ManagedElement</dnPrefix>
    <networkManagedElementId>1</networkManagedElementId>
    <userLabel>Managed Element</userLabel>
    <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
      <policyControlFunctionId>1</policyControlFunctionId>
      <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
        <entityDataId>1</entityDataId>
        <EDTargets xmlns="urn:com:ericsson:ecim:edtargetsmom">
          <eDTargetsId>1</eDTargetsId>
          <EDTarget xmlns="urn:com:ericsson:ecim:edtargetmom" xmlns:nc="urn:com:ericsson:nc">
            <EDTargetId>SubsIdsModify</EDTargetId>
            <definition>
              def SubsIdsModify(arg1 = 'AccessData.subscriber.msisdn'
              dataTarget = {
                url = "diameterOutgoingMessage:CC-Answer";
              }
              fieldDef =
              {
                Subscription-Id = add(Subscription-Id-Type = 0, S
                                add(Subscription-Id-Type = 1, S
              }
            </definition>
          </EDTarget>
        </EDTargets>
      </EntityData>
    </PolicyControlFunction>
  </ManagedElement>
</config>
</edit-config>

```

### Example 7 Configuration using AVP values with dynamic content

In the Example 7, two instances of Subscription-Id grouped AVP are added for CCA messages:

- Subscription-Id-Type set to 0 and Subscription-Id-Data AVP containing the resolution of AccessData.subscriber.msisdn policy tag
- and Subscription-Id-Type set to 1 and Subscription-Id-Data AVP containing the resolution of AccessData.subscriber.imsi policy tag



## 2.3 Configure Flexible Output Dictionary Syntax

Only configure flexible output dictionary if the SAPC uses an EDTarget that requires any AVP not included in the SAPC dictionary. To see the list of AVPs that the SAPC supports by default for Gx interface, see [Gx Interface Description](#).

Configuration of DIAMETER information for specific AVPs consists of configuring the desired AVPs, using entries through Netconf interface or COM CLI.

Configure an AVP entry per each AVP to be added, not included in the dictionary provided by default, within fieldDef attribute of the EDTarget. Otherwise, transformation orders introduced in fieldDef are meaningless, as the flexible output protocol does not know how to proceed.

An EDTarget driver for url set to diameterOutgoingMessage, uses the dictionary provided by default together with flexible output dictionary to interpret the conditions included in matchingPattern and transformations included in fieldDef attributes. For further information about flexible output protocol diameter driver, see Section 3 on page 25.

Following, the configuration of diameter dictionary syntax for the transformation order in Example 2:



```
<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <FlexibleDiameter>
          <flexibleDiameterId>1</flexibleDiameterId>
          <AVP>
            <avpId>Operator-Specific-AVP1</avpId>
            <avpCode>10003</avpCode>
            <avpVendor>50000</avpVendor>
            <avpDataType>Integer32</avpDataType>
            <avpMandatoryFlag>true</avpMandatoryFlag>
            <avpProtectedFlag>false</avpProtectedFlag>
          </AVP>
        </FlexibleDiameter>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>
```

#### Example 8 Addition of new AVP to Flexible Dictionary

In the Example 8, Operator-Specific-AVP1 AVP is defined.

Following, the configuration of flexible output dictionary syntax for the transformation orders in Example 5:



```

<edit-config>
  <target>
    <running />
  </target>
</edit-config>
<config>
  <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
    <managedElementId>1</managedElementId>
    <dnPrefix>dc=ManagedElement</dnPrefix>
    <networkManagedElementId>1</networkManagedElementId>
    <userLabel>Managed Element</userLabel>
    <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
      <policyControlFunctionId>1</policyControlFunctionId>
      <FlexibleDiameter>
        <flexibleDiameterId>1</flexibleDiameterId>
        <AVP>
          <avpId>Operator-Specific-AVP2</avpId>
          <avpCode>30000</avpCode>
          <avpVendor>50000</avpVendor>
          <avpDataType>OctetString</avpDataType>
          <avpMandatoryFlag>true</avpMandatoryFlag>
          <avpProtectedFlag>false</avpProtectedFlag>
        </AVP>
      </FlexibleDiameter>
    </PolicyControlFunction>
  </ManagedElement>
</config>
</edit-config>

```

#### Example 9 Addition of new AVP to Flexible Dictionary

The Example 9 defines Operator-Specific-AVP2 AVP.

Transformation order in Example 7 does not require configuration of Diameter AVPs since the AVPs involved are not operator-specific and as a consequence, are already defined.

## 2.4 Configuration Examples for Flexible Output Use Cases

### 2.4.1 Downloading New Operator Specific Subscriber Data

This chapter shows how to configure the following case: a new operator-specific attribute is required to be associated to the subscriber and needs to be downloaded to the Diameter peer using CCA or RAR messages.



```

PUT /subscribers/34600000002
{
  "operatorSpecificInfos":
  [
    {
      "attributeName" : "plan",
      "attributeValue" : "vip3"
    }
  ],
  "subscriberId": "34600000002"
}

```

### Example 10 Subscriber configuration

```

<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <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:com:ericsson:ecim:nc">
              <eDSourceId>Subscriber</eDSourceId>
              <definition>
                def Subscriber ( argId ) {
                  dataSource = {
                    url = "internaldb:";
                    query = "SubscriberPot:{argId}";
                  }
                  fieldDef = {
                    id = dataSourceField("id");
                    groups = dataSourceField("datapplans");
                    trafficIds = dataSourceField("trafficIds");
                    sharedDataplan = dataSourceField("sharedDataplan");
                    subscribedServices = dataSourceField("subscribedServices");
                    blacklistServices = dataSourceField("deniedContentFilteringProfileId");
                    contentFilteringProfileId = dataSourceField("contentFilteringProfileId");
                    chargingProfile = SubsChargingProfile(dataSourceField("chargingProfile"));
                    chargingSystem = OnlineChargingSystemProfile(dataSourceField("chargingSystem"));
                    customerId = dataSourceField("customerId");
                    maxBearerQosProfile = BearerQosProfile(dataSourceField("maxBearerQosProfile"));
                  }
                }
              </definition>
            </EDSource>
          </EDSources>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>

```



```

minBearerQosProfile = BearerQosProfile(dataSource
servicesToRedirect = dataSourceField("redirect
presenceReportingAreaNames = dataSourceField("
pdnGwListName = PdnGwListProfile(dataSourceFie
spid = dataSourceField("spid");
usageLimits = dataSourceField("usageLimits");
sms = dataSourceField("smsDestinations");
eventTriggers = dataSourceField("eventTriggers
tarifPlan = dataSourceField("operatorSpecificI
    }
}
</definition>
</EDSource>
</EDSources>
<EDTargets xmlns="urn:com:ericsson:ecim:edtargetsmom">
  <EDTargetsId>1</EDTargetsId>
  <EDTarget xmlns="urn:com:ericsson:ecim:edtargetmom" xmlns:
    <EDTargetId>AddSubsDataProf</EDTargetId>
    <definition>
      def AddSubsDataProf(arg1 = 'Subscriber.tarifPlan') {
        dataTarget = {
          url = "diameterOutgoingMessage:RA-Request";
        }
        fieldDef = {
          Operator-Specific-AVP1 = add(arg("arg1"));
        }
      }
    </definition>
  </EDTarget>
</EDTargets>
</EntityData>
</PolicyControlFunction>
</ManagedElement>
</config>
</edit-config>

```

### Example 11 Download new operator specific

In the Example 11, it is added the attribute `tarifPlan` in the Subscriber Entity Data Source (for more information, refer to [Database Access](#)). Subscriber “34600000002” is provisioned with plan attribute set to “vip3”. A EDTarget called “AddSubsDataProf” is defined, containing a transformation order where the new subscriber data is added within the desired AVP (Operator-Specific-AVP1, configured in the Diameter dictionary in Example 8).

Include “AddSubsDataProf” in a flexible output policy, as explained in Section 1.2 on page 2. Configure the flexible output policy in a similar way as explained in Section 2.1 on page 5.



## 2.4.2 Adding an Unsupported Standard AVP

This chapter shows how to configure the following case: the SAPC does not support a standard AVP, but such standard AVP is required in the Diameter peer. To see the list of AVPs that the SAPC supports by default for Gx interface, see [Gx Interface Description](#).

In the Example 12f, the Subscriber is subscribed to a preconfigured service (Service1002) and the service is always authorized. The transformation order is configured in a profile with policies at **service level**.

**Note:** Similarly, this can be done for dynamic PCC Rules, see Page 12.

```
PUT /contents/Service1002
{
  "contentName" : "Service1002",
  "flows":
  [
    {
      "destIpAddr" : "any",
      "destPort" : "",
      "direction" : "dl",
      "flowName" : "1",
      "protocol" : "ip",
      "sourceIpAddr" : "192.168.1.2",
      "sourcePort" : "5001-5050"
    },
    {
      "destIpAddr" : "any",
      "destPort" : "",
      "direction" : "dl",
      "flowName" : "2",
      "protocol" : "ip",
      "sourceIpAddr" : "192.168.1.2",
      "sourcePort" : "5101-5150"
    }
  ],
  "pccRuleId" : 1002,
  "pccRuleType" : 2,
  "staticQualification":
  {
    "contentQosProfileId" : "QCI9"
  }
}

PUT /profiles/content-qos/QCI9
{
  "mbrDownlink" : 5120,
  "mbrUplink" : 512,
  "profileId" : "QCI9",
  "qci" : 9
}
```





```
}
```

```
PUT /subscribers/34600000001
{
  "subscribedContents":
  [
    {
      "contentName" : "Service1002"
    }
  ],
  "subscriberId": "34600000001"
}
```

Example 12 Configuration of target service to send to the PCEF



```

<edit-config>
  <target>
    <running />
  </target>
</config>
  <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
    <managedElementId>1</managedElementId>
    <dnPrefix>dc=ManagedElement</dnPrefix>
    <networkManagedElementId>1</networkManagedElementId>
    <userLabel>Managed Element</userLabel>
    <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
      <policyControlFunctionId>1</policyControlFunctionId>
      <FlexibleDiameter>
        <flexibleDiameterId>1</flexibleDiameterId>
        <AVP>
          <avpId>Standard-AVP</avpId>
          <avpCode>11111</avpCode>
          <avpVendor>50000</avpVendor>
          <avpDataType>UTF8String</avpDataType>
          <avpMandatoryFlag>false</avpMandatoryFlag>
          <avpProtectedFlag>false</avpProtectedFlag>
        </AVP>
      </FlexibleDiameter>
    <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
      <entityDataId>1</entityDataId>
      <EDTargets xmlns="urn:com:ericsson:ecim:edtargetsmom">
        <eDTargetsId>1</eDTargetsId>
        <EDTarget xmlns="urn:com:ericsson:ecim:edtargetmom" xmlns:nc="urn:com:ericsson:nc">
          <eDTargetId>AddStandardAvp</eDTargetId>
          <definition>
            def AddStandardAvp () {
              dataTarget = {
                url = "diameterOutgoingMessage";
                matchingPattern = "Charging-Rule-Install.Charging-Rule-Definition[CH];"
              }
              fieldDef = {
                Charging-Rule-Install.Charging-Rule-Definition[CH];
              }
            }
          </definition>
        </EDTarget>
      </EDTargets>
    </EntityData>
  </PolicyControlFunction>
</ManagedElement>
</config>
</edit-config>

```

Example 13 Adding an unsupported standard AVP to the PCEF



The Example 13 defines an EDTarget called "AddStandardAvp", containing a transformation order to add Standard-AVP AVP, inside Charging-Rule-Definition, with value "AF12" in both RAR and CCA messages to the PCEF when Charging-Rule-Name is "1002" and QoS-Class-Identifier is 9.

Include "AddStandardAvp" in a flexible output policy, as explained in Section 1.2 on page 2. Configure the flexible output policy in a similar way as explained in Section 2.1 on page 5





### 3 Flexible Output Protocol Language Syntax

The flexible output protocol is an internal element designed to:

- Interpret the structure of an EDTarget when the url is set to `diameterOutgoingMessage`.
- Execute operations, within EDTarget fieldDef attribute, that modify outgoing messages content: **add/delete/replace** AVPs, considering a **matching pattern** for incoming messages.
- Access the information contained in both, the flexible output dictionary which, stores AVP definitions not included in the SAPC dictionary and the SAPC dictionary.

The language used in the flexible output protocol is based on an ABNF syntax.

Next figure shows an example of EDTarget with url set to `diameterOutgoingMessage` and how the internal flexible output protocol interprets each part:



Figure 4 Example of Message Transformation Orders In EDTarget

More examples and recommendations about message transformation orders in Section 3.2 on page 28.

The meaning of the EDTarget attributes is the following:

Table 1 EDTarget Attributes Definition

Element	Description	Comments
<dataTarget>	Identifies the data intended to be transformed.	
url	EDTarget including url attribute set to “diameterOutgoingMessage” indicates that flexible output protocol is required to interpret and apply the remaining attributes within the EDTarget.	Optionally, may include the Diameter message type, which indicates the target message to apply matching criteria and transformations.
matchingPattern	Refine url attribute indicating further conditions (AVPs including specific values) that outgoing diameter message must fulfill in order to transformation being applied.	In the previous example, transformations are only applied to Diameter RAR messages which, include AVP Monitoring-Key = “Ø” into one instance of Grouped AVP Usage-Monitoring-Information. The syntax of matchingPattern attribute is indicated in Section 4 on page 37.
<fieldDef>	Includes the transformations to be applied by flexible output protocol to the outgoing message coming from the SAPC.	

- The first element of the transformation, included in the fieldDef attribute, indicates the AVP intended to be applied a transformation. Following considerations must be taken into account related to Diameter messages (refer to Section 3.2 on page 28 for transformation examples and Section 4 on page 37 for transformations specification):
  - Messages can contain multiple instances of the same AVP, so a condition to indicate whether the transformation applies to every instance or just to a specific one must be provided.
  - Messages usually contain AVPs nested into Grouped AVP. To apply a transformation, the complete concatenation of Grouped AVPs containing the AVP intended to be applied a transformation must be indicated.
- The second element includes the type of transformation to be applied (add/delete/replace) and a value (if necessary).



- Operator “add” means an addition, “delete” a removal and “replace” a substitution in that position.
- For each AVP name, take into account the AVP type, refer to CPI Interface Description documentation for AVPs included into SAPC dictionary or avpDataType attribute of AVP MOClass for operator-specific AVPs.

---

### Warning!

For the AVP values of OctetString type, the operator needs to configure the ASCII characters corresponding to the hexadecimal values to be sent by the SAPC. For how to configure the values, see Section 3.1 on page 27.

---

The SAPC does not validate that the length of the AVP value configured is according to the AVP definition. It is responsibility of the operator that configures this AVP value to take into account reasonable lengths.

It is possible to include dynamic content in the AVP values, EDTarget supports input arguments by using Condition Formula Language expressions (refer to [Configuration Guide for Subscription and Policies](#)), configured as default argument as shown in [Page 25](#).

**Note:** If the AVP is a grouped one, an array is used as value, being each array element the contained AVP.

In the previous example the flexible output protocol **workflow** is the next:

1. Check if the outgoing diameter message fits the **matching pattern**, that is, it contains the AVP instance Monitoring-Key with value “0” included into AVP Usage-Monitoring-Info.
2. If previous step, then **add** in the outgoing message a new AVP called Vendor-Specific-AVP1 with dynamic value “AccessData.subscriber.imsi”, replace the value of AVP instance Charging-Rule-Base-Name with value “100” included into AVP Charging-Rule-Install with value “101” and **remove** the AVP called Allocation-Retention-Priority, regardless of its value, included into any instance of AVP QoS-Information.

For rest of AVPs, no transformation is done.

**Note:** Any grouped AVP left empty as result of a **remove** transformation, is also removed.

## 3.1 Configure AVP Values of OctetString Type

In the EDTarget fieldDef attribute, set “0x...” (case insensitive). The characters following “0x” are encoded in hexadecimal representation, each character with two digits (00 - FF). The length of the hexadecimal value must be even.



- Correct example: "0X4335".

**Note:** It is also possible to set ASCII characters such as "C5" for printable characters. The SAPC sends it in hexadecimal representation ("4335").

- Incorrect examples: "0X43353" (odd length following the prefix), "0X43353g" (out of range 00 - FF). If the values are invalid, the SAPC shows configuration error.

In the following configuration example, the SAPC sends hexadecimal representation "0A0E" (2 bytes).

```
def ModifyMessage(){
  dataTarget = {
    url = "diameterOutgoingMessage";
  }
  fieldDef = {
    QoS-Rule-Install.QoS-Rule-Definition[QoS-Rule-Name[1001]].Resource-Allocation
    QoS-Rule-Install.QoS-Rule-Definition[QoS-Rule-Name[1001]].ToS-Traffic-Class
  }
}
```

## 3.2 Flexible Output Protocol Message Transformation Orders

Following, there is the list of supported types of outgoing message transformations, together with their corresponding configuration examples.

Table 2 Flexible Output Protocol Transformation Orders

Operation	Message Transformation Orders in EDTarget	Comments
Single AVP conditional addition/removal	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage:CC-Answer";   }   fieldDef = {     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]].Vendor-Specific-AVP1 with value "value1"     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]].Flow   } }</pre>	<p>For CCAs whose Charging-Rule-Name is "2000":</p> <ul style="list-style-type: none"><li>• Add Vendor-Specific-AVP1 with value "value1"</li><li>• Remove Flow-Description AVP, no matter its value (wildcard null is used).</li></ul>





Table 2 Flexible Output Protocol Transformation Orders

Operation	Message Transformation Orders in EDTarget	Comments
Single AVP value replacement	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage:CC-Answer";   }   fieldDef ={     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]].Reporting-Level   } }</pre>	<p>For CCAs whose Charging-Rule-Name is "2000":</p> <p>Replace Reporting-Level AVP with value "1" for value "0".</p>
Single AVP value replacement	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage:CC-Answer";   }   fieldDef ={     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]].Reporting-Level   } }</pre>	<p>For CCAs whose Charging-Rule-Name is "2000":</p> <p>Replace Reporting-Level AVP, regardless its value for "0".</p>
Grouped AVP unconditional addition	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage:CC-Answer";   }   fieldDef ={     My-Non-Standard-Grouped-AVP = add(Vendor-Specific-AVP1,Another-Vendor-Specific-AVP2)   } }</pre>	<p>For CCAs, it is added the grouped My-Non-Standard-Grouped-AVP AVP, containing:</p> <ul style="list-style-type: none"> <li>• Vendor-Specific-AVP1 with value "value1".</li> <li>• Another-Vendor-AVP AVP, with value 800.</li> </ul>
Grouped AVP unconditional removal	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage";   }   fieldDef ={     Charging-Rule-Install.Charging-Rule-Definition.QoS-Information = delete();   } }</pre>	<p>Removal of the grouped QoS-Information AVP within Charging-Rule-Definition, regardless of its content.</p>



Table 2 Flexible Output Protocol Transformation Orders

Operation	Message Transformation Orders in EDTarget	Comments
Grouped AVP conditional removal	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage";   }   fieldDef ={     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000] &amp;&amp; Service-Identifier[2000]]   } }</pre>	<p>Removal of Charging-Rule-Definition instance that contains:</p> <ul style="list-style-type: none"> <li>• Charging-Rule-Name with value "2000"</li> <li>• and Service-Identifier with value "service1"</li> </ul>
Multivalued AVPs, unconditional addition	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage:CC-Answer";   }   fieldDef ={     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000] &amp;&amp; Service-Identifier[2000]]   } }</pre>	<p>For CCAs, for every instance of Charging-Rule-Definition AVP, add the My-Non-Standard-AVP 2 AVP, containing value "value2".</p> <p>When several policies select the same EDTarget with same arguments, the SAPC only applies the contained transformation orders one time per each outgoing message.</p>
Multivalued AVPs, multiple unconditional addition	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage:CC-Answer";   }   fieldDef ={     Event-Trigger = add(value1) &amp;&amp; add(value2);   } }</pre>	<p>For CCAs:</p> <ul style="list-style-type: none"> <li>• Add Event-Trigger AVP with value "value1".</li> <li>• Add Event-Trigger AVP, with value "value2".</li> </ul> <p>When several policies select the same EDTarget with same arguments, the SAPC only applies the contained transformation orders one time per each outgoing message.</p>



Table 2 Flexible Output Protocol Transformation Orders

Operation	Message Transformation Orders in EDTarget	Comments
Multivalued AVPs, conditional addition	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage:CC-Answer";   }   fieldDef ={     Charging-Rule-Install.Charging-Rule-Definition.Charging-Rule-Name[2].Flow-Information   } }</pre>	<p>For CCAs, for Charging-Rule-Definition AVP that contains Charging-Rule-Name with value 2, within every instance of Charging-Rule-Definition AVP, add My-Traffic-Class AVP set to value "CS5"</p> <p>When several policies select the same EDTarget with same arguments, the SAPC only applies the contained transformation orders one time per each outgoing message.</p>
Multivalued AVPs, unconditional removal	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage:RA-Request";   }   fieldDef ={     Event-Trigger = delete();   } }</pre>	For RARs, remove all instances of Event-Trigger AVP.

**Note:** Conditional transformation orders refer to matching content.

To avoid unexpected results in the transformation orders, follow next recommendations.



Table 3 Flexible Output Protocol Considerations

Operation	Message Transformation Orders in EDTarget	Comments
Matching pattern use	<pre>def ModifyMessage() {   dataTarget = {     url = "diameterOutgoingMessage";     matchingPattern = "Usage-Monitoring-Information";   }   fieldDef = {     Usage-Monitoring-Information.Monitoring-Key[0].Usage-Monitoring-Report = add     Usage-Monitoring-Information.Monitoring-Key[0].Usage-Monitoring-Level = add     Usage-Monitoring-Information.Monitoring-Key[300].Usage-Monitoring-Report =     Usage-Monitoring-Information.Monitoring-Key[300].Usage-Monitoring-Level =   } }</pre>	<p>Matching pattern provides the ability to filter outgoing messages from SAPC before executing the transformation orders.</p> <p>A proper matching pattern configuration can provide a better SAPC throughput since you save unnecessary processing</p>



Table 3 Flexible Output Protocol Considerations

Operation	Message Transformation Orders in EDTarget	Comments
Simultaneous additions and removals of AVPs	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage:CC-Answer";   }   fieldDef ={     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]].D     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]].D     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]].P     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]].C     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]].C   } }</pre>	<p>When configuring simultaneous additions and removals, the transformations order within the fieldDef attribute is not relevant.</p> <p>Every transformation is evaluated taking into account the original Diameter outgoing message and not the result of previous transformations.</p> <p>The transformation order is independent of the AVP occurrence order in the message to be sent to the network.</p>
Removal as part of matching pattern.	<pre>def ModifyMessage() {   dataTarget={     url = "diameterOutgoingMessage:CC-Answer";   }   fieldDef ={     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]] &amp;&amp;     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]] &amp;&amp;     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]] &amp;&amp;     Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[2000]] &amp;&amp;   } }</pre>	<p>When removing several AVPs in a row, make sure that they are all present in the original message.</p> <p><b>Remove transformation also involves matching so if any AVP does not appear or it does but not matching its specified value, the transformation is not applied.</b></p> <p>Within Charging-Rule-Definition AVP containing Charging-Rule-Name with value 2000, the following AVPs are removed: Flow-Description, Precedence, Service-Identifier, Rating-Group if and only if both Flow-Description value is 900 and Service-Identifier value is 250.</p>

### 3.2.1 Transformation Orders: Be as Much Specific as Possible

Depending on how transformations orders are specified, there can be ambiguous situations (either in add, delete, or replace operations) for which flexible output protocol affects to AVP instances where it is not desired to apply. To avoid these not desired impacts, do specify the conditions in flexible output protocol transformation: the greater attention on a detailed specific transformation, the lesser on the risk of ambiguity.

Next example shows a case where an ambiguous use of flexible output protocol transformation provokes unexpected results.

#### Transformation Order, erroneous effect

Imagine the case where flexible output protocol is used to delete Precedence AVP in the P2P service (Charging-Rule) when it is installed in a CCA message.

The configured EDTarget is the following:

```
def ModifyMessage(){
  dataTarget = {
    url = "diameterOutgoingMessage:CC-Answer";
  }
  fieldDef = {
    Charging-Rule-Install.Charging-Rule-Definition.Precedence = delete();
  }
}
```

P2P is supposed to be the only service that can be installed in the CCA. The transformation used does not specify a condition regarding the Charging-Rule-Name AVP. That is a possible configuration but too ambiguous.

What happens if the CCA message does not only include P2P service (as initially thought), but it also includes another VoIP Charging-Rule? See next Figure:

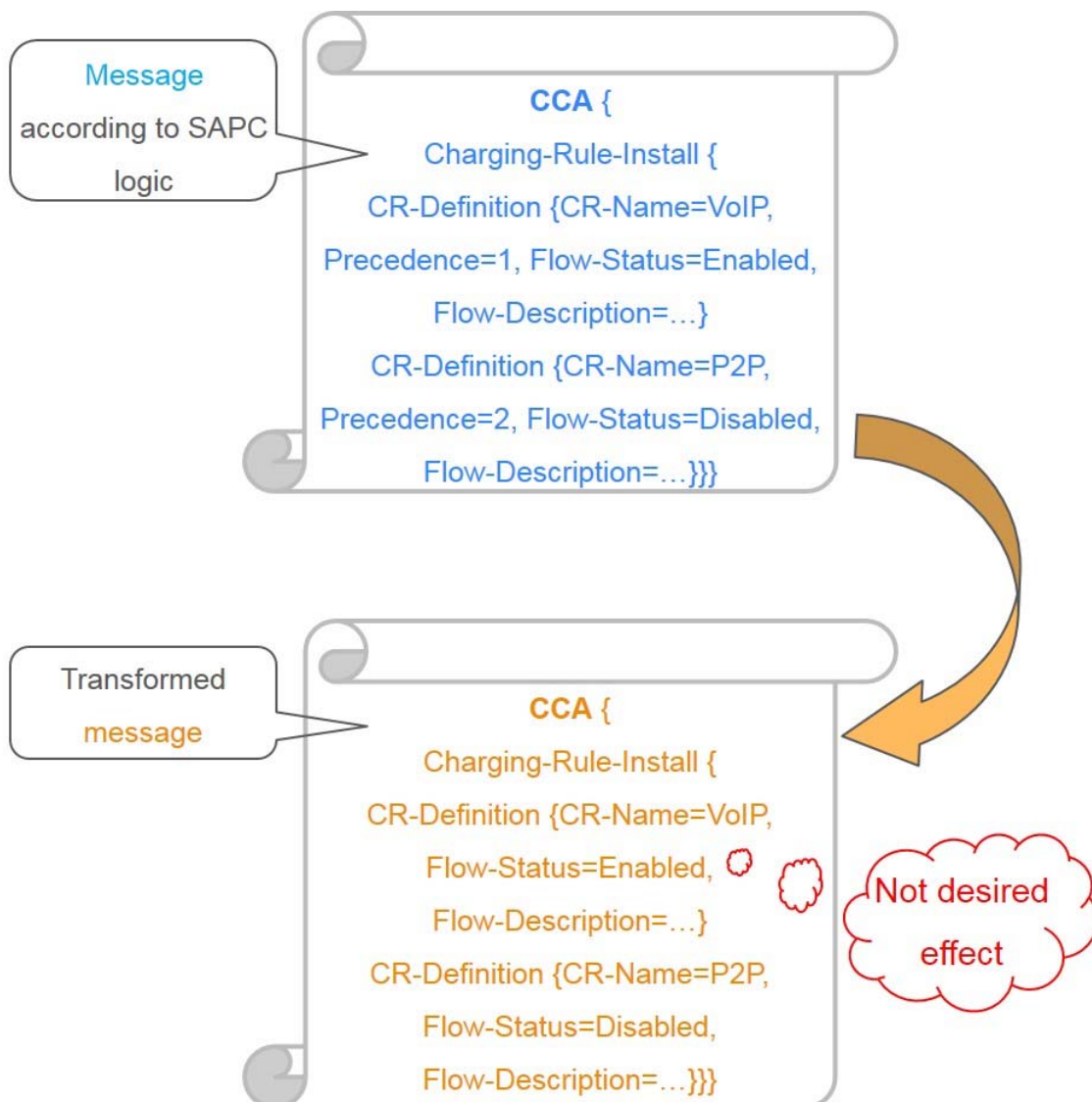


Figure 5 Ambiguous Transformation

An erroneous (at least, not the expected) transformation result is obtained: Precedence AVP is removed from VoIP and P2P Charging-Rule, as no Charging-Rule-Name condition is specified.

#### Suggested Solution

The solution is to configure the Charging-Rule-Name condition so that is satisfied for P2P service:

```
def ModifyMessage(){
  dataTarget = {
```



```
url = "diameterOutgoingMessage:CC-Answer";  
}  
fieldDef = {  
  Charging-Rule-Install.Charging-Rule-Definition[Charging-Rule-Name[P2P]].P  
}  
}
```

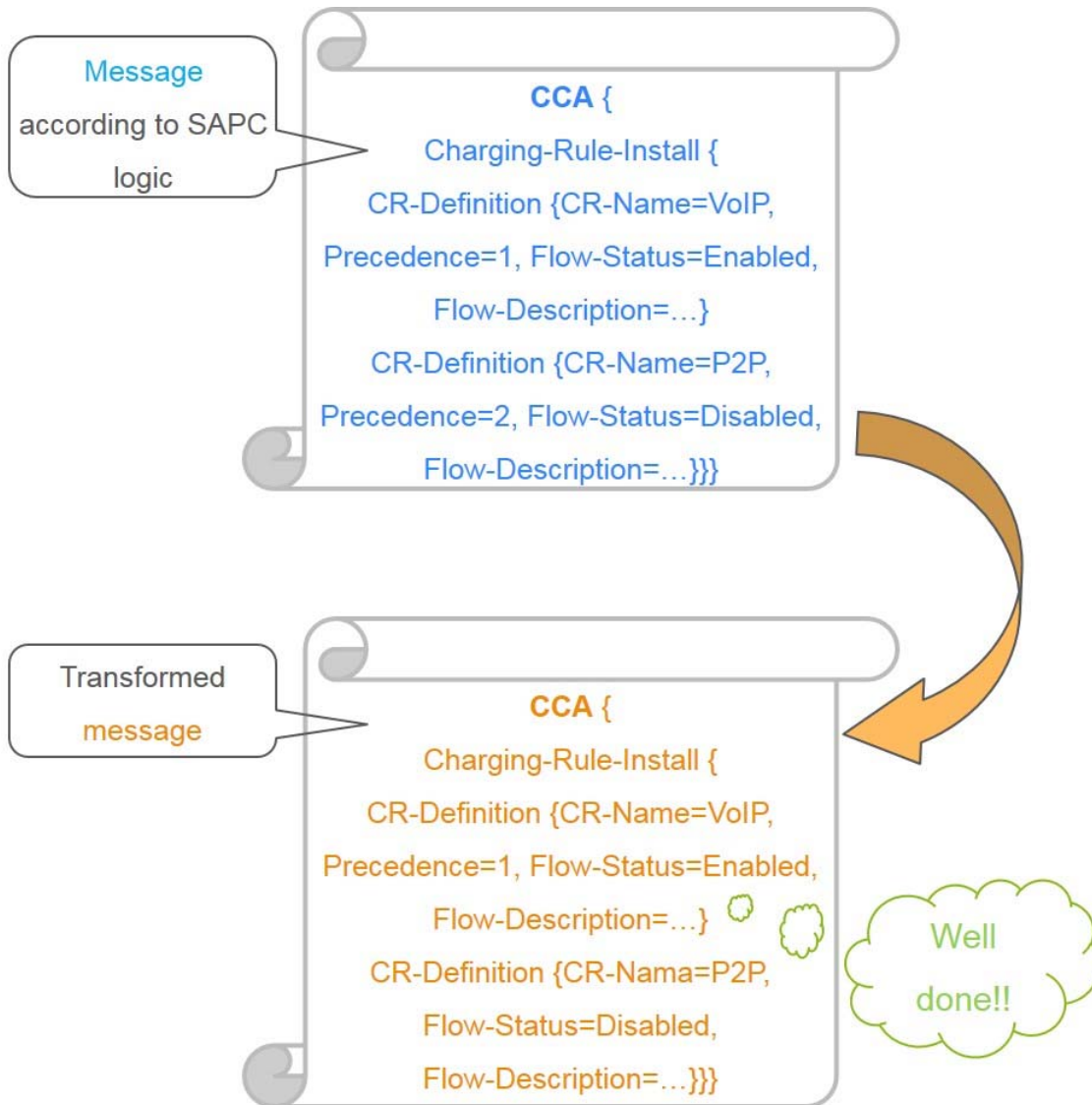


Figure 6 Specific Transformation





## 4 Transformation and Matching Criteria Specification

The ABNF grammar used for syntax in EDTarget fieldDef is:

```
Transformation = Element "=" Modification
Element = Attribute *("."Attribute)
Attribute = AVP Condition
AVP = VCHAR
Condition = *1 "[" (SingleCondition / GroupedCondition) "]"
SingleCondition = VCHAR
GroupedCondition = NestedAVP "["Single"]" *("&&" NestedAVP "["Single "]")
NestedAVP = AVP *("." AVP)
Modification = SingleModification *("&&" SingleModification)
SingleModification = Operator "(" Value ")"
Operator = ("add" / "delete" / "replace")
Value = *1(SingleValue/GroupedValue)
SingleValue = VCHAR
GroupedValue = GroupedInstance *("," GroupedInstance)
GroupedInstance = AVP "=" (SingleValue/ "("GroupedValue")")
```

### Example 14 ABNF Grammar for Transformations

The ABNF grammar used for syntax in EDTarget matchingPattern is:

```
MatchingCriteria = Element *("&&" Element)
Element = Attribute *("."Attribute)
Attribute = AVP Condition
AVP = VCHAR
Condition = "[" (SingleCondition / GroupedCondition) "]"
SingleCondition = VCHAR
GroupedCondition = NestedAVP "["Single"]" *("&&" NestedAVP "["Single "]")
NestedAVP = AVP *("." AVP)
```

### Example 15 ABNF Grammar for Matching Criteria





# 5 Appendix A. Flexible Output Policy Types

Next figure shows the policy type related to flexible output that can be configured in the SAPC.

Flexible Output


Policy Type	Policy Locator			Output Attributes
	Context	Resource	Subject	
Flexible Output	flexible-output	any	<subscriberId> <dataplanId> 	permit flexible-output <edt_name(args)>

Figure 7 Flexible Output Policies in the SAPC