MTAS Generic Supplementary Service Codes Management Guide

MTAS

USER GUIDE

88/1553-AVA 901 29/9 Uen B

Contents

1 Introduction

This document describes how to configure the Generic Supplementary Service Codes (GSSC) in the MTAS to enable users to access and control their supplementary service by using generic service code commands.

1.1 Prerequisites

It is assumed that the user of this document is familiar with the following topics:

- Operation and Maintenance (O&M), in general
- XML technology, refer to W3C XML 1.0, Extensible Markup Language (XML) 1.0
- XPath technology, refer to XML Path Language (XPath) Version 1.0
- XSLT technology and the use of Regular Expressions, refer to GNU Regular Expression Extensions
- XCAP protocol, refer to RFC 4825, The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)
- Ut (XCAP) interface in the MTAS, refer to *MTAS Ut Interface*
- XCAP schemas used in the MTAS, refer to *MTAS Ut Structure*
- Man–Machine Interface to supplementary services, refer to ETSI 300738, Man-machine Interface (MMI) to Public Network based Supplementary Services

1.1.1 Licenses

To enable basic services in the MTAS, the MMTel license must be installed.

The GenSSC reuses the XCAP functions, that is, the Ut interface license must be installed in the MTAS node.

For more information about the licenses, refer to *MTAS Licenses*.

For more information about the Ut interface, refer to the following documents:

- *MTAS Ut Interface*
- *MTAS Ut Structure*

1.1.2 Documents

Before starting any procedure in this document, ensure that the following documents are available:

- *Ericsson Command-Line Interface User Guide*
- *Managed Object Model (MOM)*

1.1.3 Conditions

The following condition must apply:

An Ericsson Command-Line Interface (ECLI) session in Exec mode is in progress.

2 Overview

The MTAS provides the following kinds of Supplementary Service Code (SSC) commands:

- System-defined SSC commands
- Generic SSC (GenSSC) commands

System-defined SSC commands represent SSC definitions with a system-defined Managed Object (MO) class per supported supplementary service and SSC action. System-defined SSC commands are available for a predefined set of supplementary services.

For more information about the SCC definitions, refer to *MTAS Supplementary Service Codes Management Guide*.

The GenSSC commands provide a flexible mechanism to customize definitions of SSC commands without the syntax restriction as of system defined SSC commands. The GenSSC commands can be used for all supported supplementary services and are described in this document.

Note: The GenSSC shares common traffic view, data view and subfunction set with the system-defined SSC functionality.

For more information about the SCC subfunctions, refer to *MTAS Supplementary Service Codes Management Guide*.

The GenSSC interacts with several components/functions within and outside the MTAS, as shown in Figure 1.

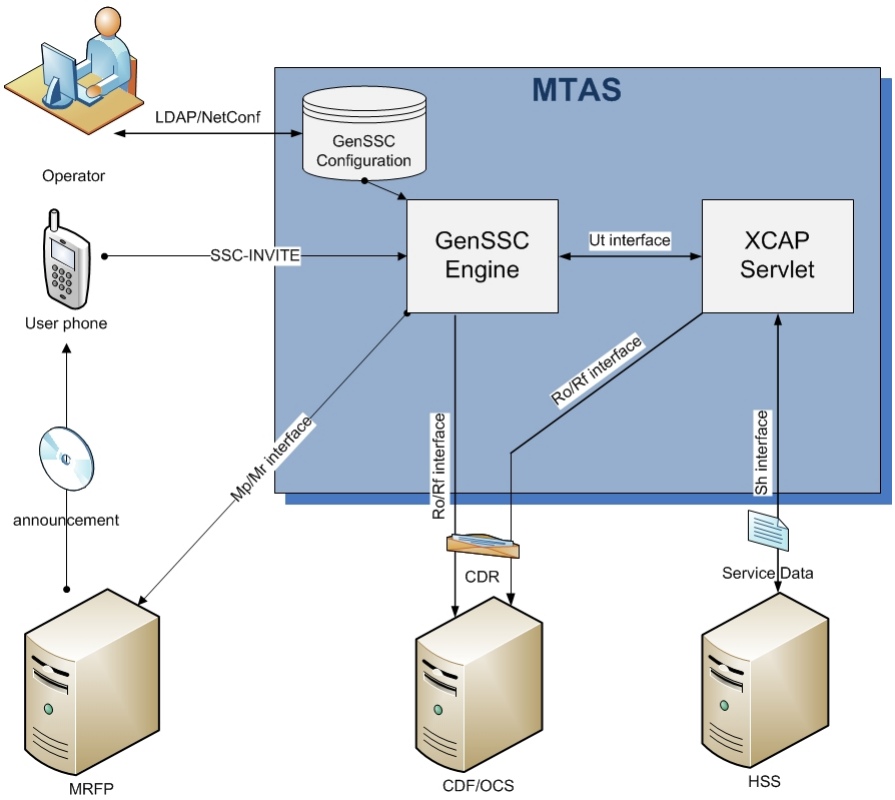

Figure 1 Components Affected by the GenSSC Function

The GenSSC commands provide the same set of supplementary service operations as is available to the user through the Ut self-service provisioning interface.

For more information about the Ut interface, refer to *MTAS Ut Interface*.

The GenSSC commands are defined to perform operations on the subscriber service data. The service data is considered as a large XML document containing the service data of the supplementary services supported by the MTAS.

The main use cases of the GenSSC commands are as follows:

- Interrogation of service data
- Update of service data
- Deletion of service data

The operator can access any piece of the service data using the XPath technology and can change any item in the service data using the XSLT technology. For more information, refer to the following standards:

- XML Path Language (XPath) Version 1.0

- XSL Transformations (XSLT) Version 1.0

The syntax of a GenSSC command is defined by a Regex. The syntax describes the layout of the input SSC command string and how to interpret or process it. The Regex capturing and substitution mechanisms are used for this purpose. For more information on Regex, refer to GNU Regular Expression Extensions.

The announcement to be played to the user must be configured by the operator in the form of an XML document. This document can be considered as a table containing announcement IDs connected to play conditions. The announcement of the first matching condition is played to the user.

To exclude the violation of the application rules valid for a service, the Ut interface is reused for the GenSSC commands. The Ut interface provides an access to the subscriber service data through the XCAP protocol.

For more information about the service data visible through the simservs.xml document, refer to *MTAS Ut Structure*.

The operator defines the information needed for the creation of an XCAP `GET`, `PUT`, or `DELETE` request.

The MTAS takes the Request-URI of the incoming `INVITE` and the operator configuration bound to the received Generic Service Code (GSC) and creates an XCAP request. This request is forwarded to the XCAP port of the node and processed as a regular XCAP request applying the application rules of the addressed supplementary service.

The XCAP response is processed and the following usual SSC tasks are performed:

- The proper Performance Management (PM) counters are incremented.
- ACR/CCR reports are sent to the CDF/OCS.
- An announcement is played to the user.

In general, the operator configures the following data for a GenSSC command:

- The leading digits of the command including the GSC
- The command syntax
- The XPath of the item (element or attribute) within the simservs.xml document
- The action to be performed on the item (interrogation, update, deletion, check if it exists, and so on)
- The new content for a service data update

The new content can be simple or complex. The simple content can be defined in the command syntax of the simple schema as a user input or as a predefined value.

If the content is complex, then the body of the XCAP operations must be configured by the operator in a layout XML document using the XSLT technology Composite schema. For more information, refer to XSL Transformations (XSLT) Version 1.0.

The XSLT style sheet in the layout document takes the GSC command parameters and other optional configuration data (such as the action code or the voice mail address) as the source and generates the XCAP body as the result of the XSL transformation. The operator can insert the source items in the proper place of the XCAP body using the XSLT operations. The MTAS performs the configured transformation and puts the result into the body of the generated XCAP request.

The advantage of the generic approach is that the GSC commands are not limited to preprogrammed solutions but they can manipulate the service data in arbitrarily.

The GenSSC uses XML documents by processing of the SSC command and by the operations on the service data. Figure 2 shows an overview of the used documents in a simple-type configuration and Figure 3 shows the documents used by a composite-type configuration.

2.1 XML Document Handling in Simple-Type Configurations

XML document handling in simple-type configurations is shown in Figure 2 and executed as follows:

If the content is complex, then the body of the XCAP operations must be configured by the operator in a layout XML document using the XSLT technology Composite schema. For more information, refer to XSL Transformations (XSLT) Version 1.0.

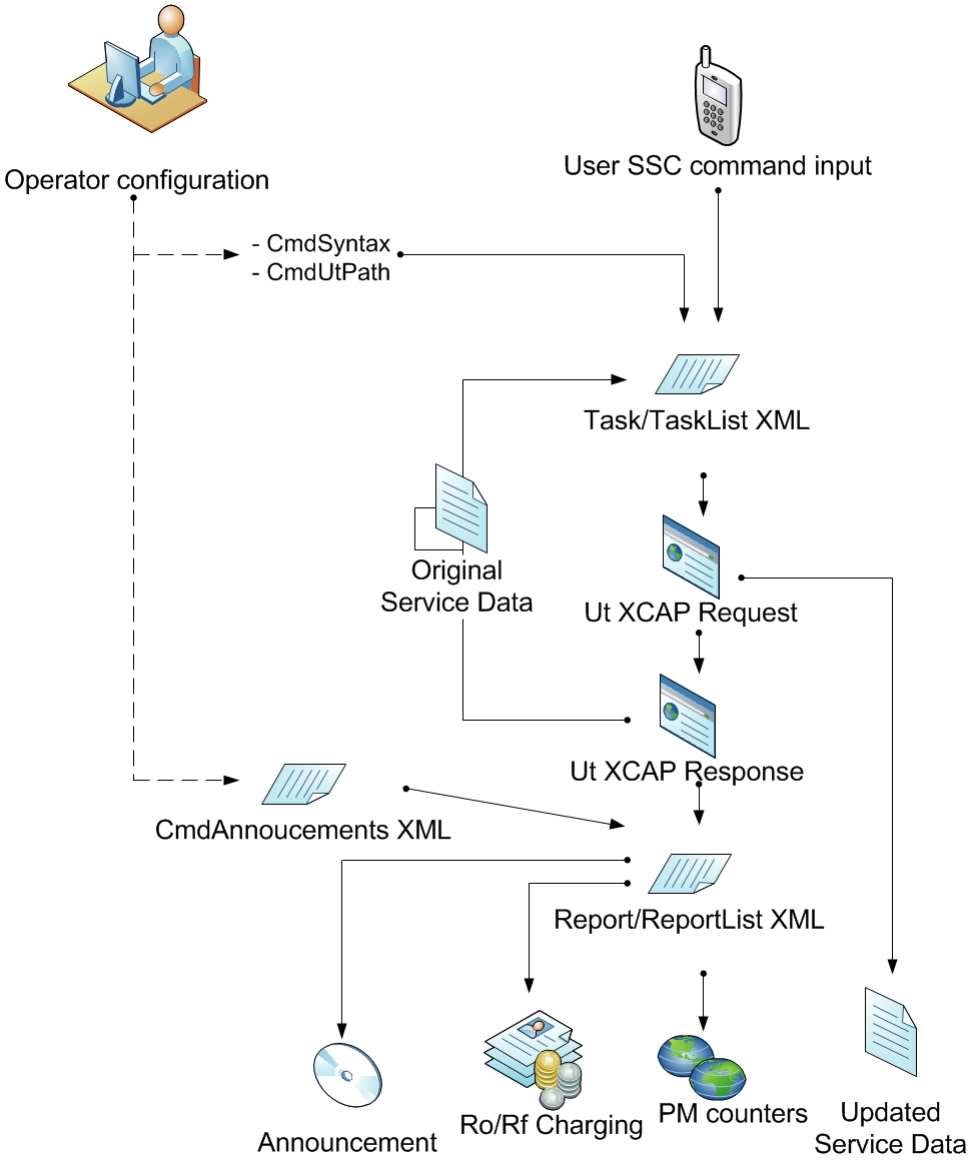

Figure 2 XML Documents Used in Simple-type Configurations

2.2 XML Document Handling in Composite-Type Configurations

The XML document handling in composite-type configurations is shown in Figure 3 and executed as follows:

- The GenSSC engine creates a source XML document from the `CmdSyntax` attribute, from the user input, and from the service data elements referred by the `CmdUtPath` attribute.

- The engine takes the Source XML and the XSLT style-sheet of the Layout XML and performs the XSL transformation on the style-sheet. The outcome must be either a Report/ReportList or a Task/TaskList XML. If the result is a Report/ReportList, then the processing continues with the last item.
- The Task or TaskList is processed and one or more XCAP requests are generated through the Ut interface. The XCAP request can update the service data.
- The XCAP responses are processed. The intermediate service data results can be reused in the processing of the subsequent task.
- A report XML document is generated from each XCAP response.
- The Report or ReportList XML is processed.
  - The announcement to be played is determined from the reports and from the CmdAnnouncements XML document provided by the operator.
  - Offline and online charging messages are generated from the report.
  - MTAS performance management counters are incremented based on the report.

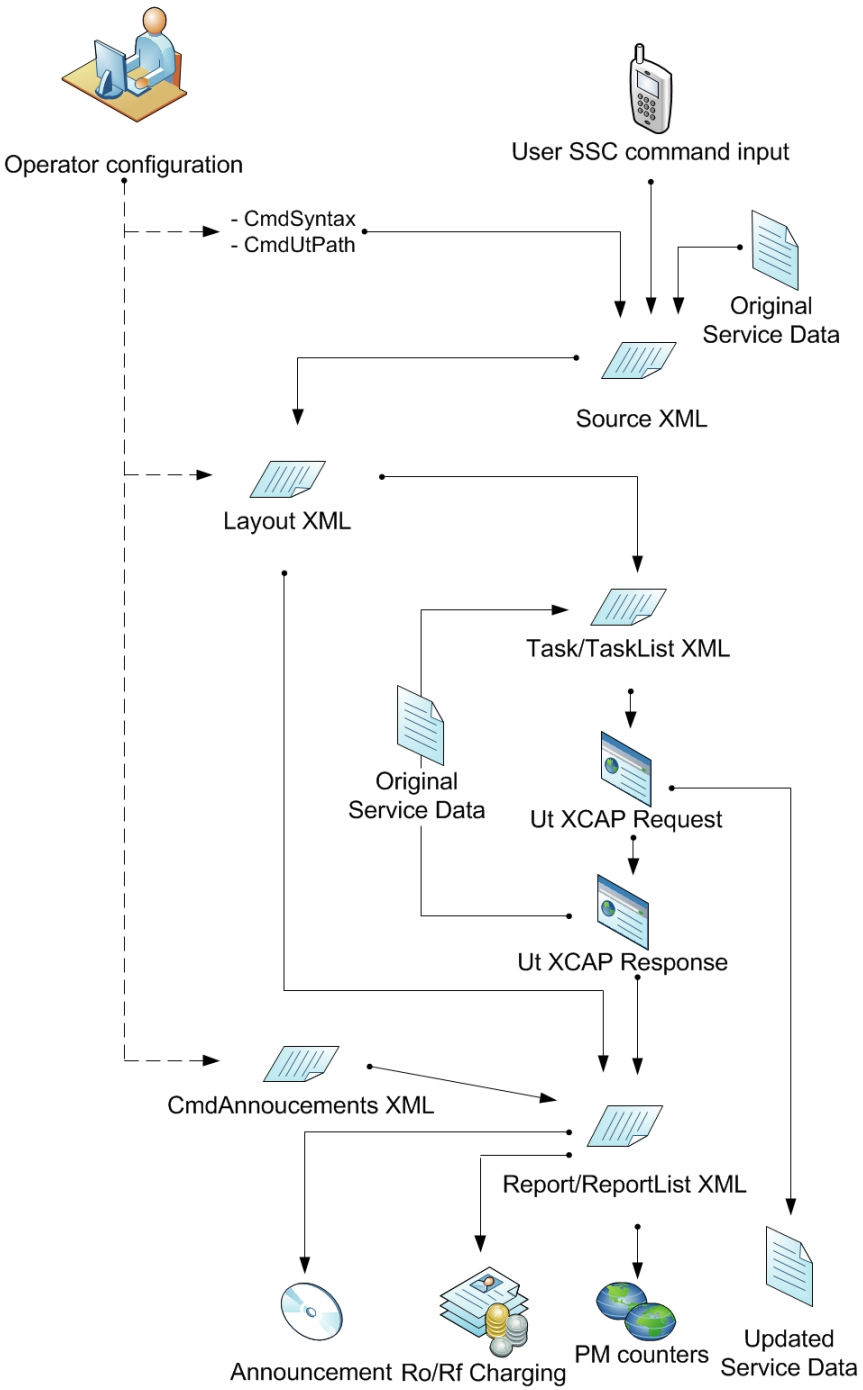

Figure 3 XML Documents Used in Composite-type Configurations

2.3 Interaction with Other Services

The GenSSC does not interact with other services.

2.4 Limitations

The following limitations apply for the GenSSC:

- The GenSSC supports only the Service Code (SC) commands issued outside a call while the terminal is idle. Therefore, the following functions (which are issued during the call setup phase when the call is initiated) are not supported by the GenSSC:
  - Invocation
  - Disabling
- In addition, the GenSSC does not support the following function:
  - Wholesale

  For more information about the wholesale functions, refer to ETSI 300 738.

3 GenSSC Configuration

The Configuration specified in this section pertains to the data set through the LDAP/NETCONF protocol. For a complete MTAS Managed Object Model (MOM), refer to *Managed Object Model (MOM)*.

The GenSSC uses Managed Object (MO) instances for processing of the SC command string in the `INVITE` Request-URI and for performing the operations associated with the given command.

The GenSSC related MOs are organized in a three-level tree as is shown in Figure 4.

The levels are as follows:

- The root of the tree is the *MtasGenSsc* MO. It is a system-created MO with only a single instance. It contains the default announcements of the GenSSC function.
- The second level serves for the organization of the GenSSC command configurations. The instances of the `MtasGenSscGroup` MO collect the GenSSC configurations of a given service/function in a group. The `MtasGenSscGroup` instances are created by the operator.
- The third level includes the instances of the operator created `MtasGenSscCmd` MO. Each instance contains the configuration of an individual GenSSC command.

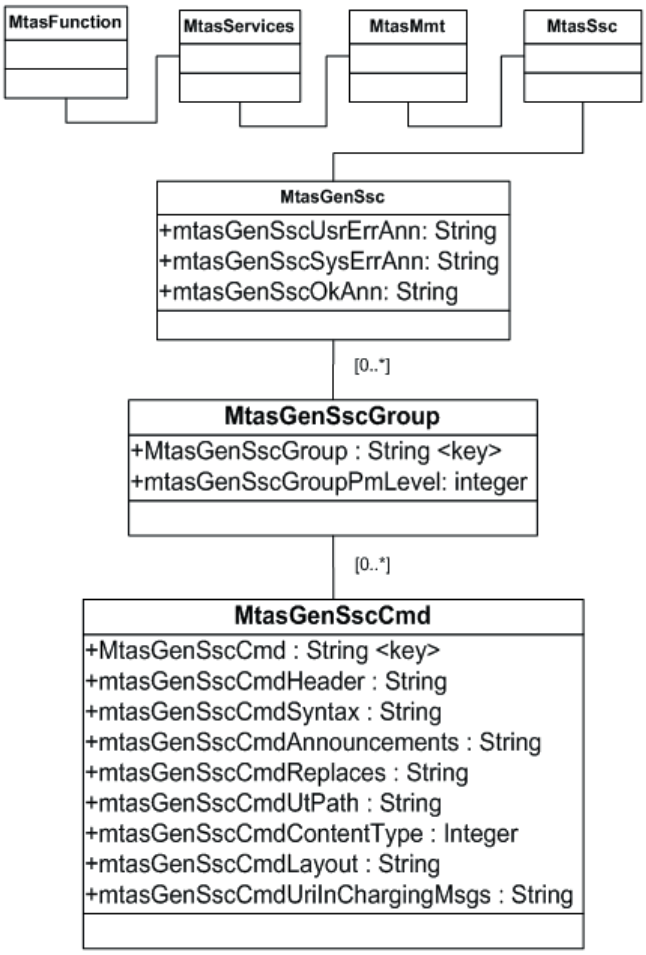

Figure 4 GenSSC MO Structure

3.1 Description of GenSSC CM Attributes

This section describes the `MtasGenSsc` MO and command syntax elements.

3.1.1 MtasGenSsc MO

This section describes the `MtasGenSsc` MO.

3.1.1.1 mtasGenSscUsrErrAnn

The attribute `mtasGenSscUsrErrAnn` defines the announcement code that is to be used for the GenSSC service failure owing to a user error, for example, when the incoming SC command string is invalid. If it begins with the `ga:` prefix, then the remaining part is interpreted as the identifier of a generic announcement. In other cases, it contains a non-negative integer, which

identifies an audio resource in the Media Resource Function (MRF)/Media Resource Function Controller (MRFP).

3.1.1.2 mtasGenSscSysErrAnn

The attribute `mtasGenSscSysErrAnn` defines the announcement code that is to be used for the GenSSC service failure owing to a system error. The cause of the failure can be an internal MTAS fault (for example, a runtime validation fault occurred owing to a wrong configuration) or an error in the interacting node (for example, a Home Subscriber Server (HSS) error).

If it begins with the `ga:` prefix, then the remaining part is interpreted as the identifier of a generic announcement. In other cases, it contains a non-negative integer, a number which identifies an audio resource in the MRF/MRFP.

3.1.1.3 mtasGenSscoOkAnn

The attribute `mtasGenSscoOkAnn` defines the announcement code that is to be used for the successful use of GenSSC service. If it begins with the `ga:` prefix, then the remaining part is interpreted as the identifier of a generic announcement. In other cases, it contains a non-negative integer, which identifies an audio resource in the MRF/MRFP.

3.1.2 MtasGenSscGroup MO

This section describes the `MtasGenSscGroup` MO.

3.1.2.1 MtasGenSscGroup

The attribute `MtasGenSscGroup` includes the key of the `MtasGenSscGroup` MO instance. The key typically contains a name referring to the function or service of which the configuration is grouped by the given instance.

3.1.2.2 mtasGenSscGroupPmLevel

The attribute `mtasGenSscGroupPmLevel` defines the level of the performance measurements created for the corresponding GenSSC group. The measurements can be created in an aggregated way for the whole group or produced separately for each GenSSC command belonging to the given group.

Possible values: `0: group-level, 1: command-level`.

3.1.3 MtasGenSscCmd MO

This section describes the `MtasGenSscCmd` MO.

3.1.3.1 MtasGenSscCmd <key>

The attribute `MtasGenSscCmd <key>` includes the key of the `MtasGenSscCmd` MO class instance. It is a string and typically refers to the goal or subject of the MO instance.

3.1.3.2 mtasGenSscCmdHeader

The command header attribute includes the leading digits of the incoming SSC-URI. It can contain only numeric digits (0–9) and the star (*) and hash (#) symbols. It is defined as long or deep as possible to avoid conflict with other SSC commands.

Example for a command header configuration: `*21*`

If the incoming URI matches a command header, then it is considered a GSC command, otherwise the `INVITE` continues with the MTAS originating call procedures.

The Command Header can contain a list of leading digits separated by the colon (|) symbol.

Example for a multi-value command header configuration: `*21#|#21#|*#21#`.

If the incoming URI matches any of the list items, then it is considered a GSC command.

3.1.3.3 mtasGenSscCmdSyntax

The attribute `mtasGenSscCmdSyntax` describes the syntax of the SSC command string. It can include a list of syntax expressions separated by the `~~` string.

The generic command syntax is a Regex given in the following format:

`/<command pattern>/<command syntax>/`

The command pattern describes the expected format of the incoming GSCC string including the supplementary information fields.

If the URI matches one of the pattern values in the `mtasGenSscCmdHeader` attribute but does not match the command pattern, then it is considered a user mistake or typo and results in a negative acknowledgment to the user.

The command pattern also includes capturing expressions of which values to be inserted into the command syntax using the Regex substitution function.

The command syntax contains the following:

- The schema (simple or composite) of the command syntax; the schema is the only mandatory part of the syntax

- The action to be performed on the item (update, create, set, switch, delete, interrogate, test, or verify)
- One or more captured or predefined data items
- Functions to be performed in addition to the action. The following functions are supported:
  - `pinCheck(pin)`
  - `serviceState(state-action)`

    The state-actions activate, deactivate, interrogate, and switch are supported.

Example for a CmdSyntax configuration:

```
/^([*#]#?)21#$/schema=simple; function serviceState(\1)/
```

3.1.3.4 mtasGenSscCmdAnnouncements

The attribute `mtasGenSscCmdAnnouncements` defines the announcements to be played to the user and the conditions when a given announcement is played in the form of an `<announcements>` XML document. It is in fact a table processed from the top row and the first matching announcement is played to the user. The last announcement is not to include the `<play-condition>` element; it is be handled as the default announcement.

The value of the `mtasGenSscCmdAnnouncements` attribute is validated against the announcement.xsd during the configuration. For more details, refer to *Managed Object Model (MOM)*.

An example document for the announcement configuration-related to the state of a service is shown in Example 1.

```
<?xml version="1.0" encoding="utf-8"?>
<announcements xmlns="http://uri.etsi.org/ngn/params/xml
/simservs/xcap/genssc">
  <announcement comment="successfully activated" media-id="200">
    <play-condition response="^2" job="put" value="true" />
  </announcement>
  <announcement comment="successfully deactivated" media-id="201">
    <play-condition response="^2" job="put" value="false" />
  </announcement>
  <announcement comment="active" media-id="202">
    <play-condition response="^2" job="get" value="true" />
  </announcement>
  <announcement comment="inactive" media-id="203">
    <play-condition response="^2" job="get" value="false" />
  </announcement>
  <announcement comment="service not provisioned" media-id="401">
    <play-condition response="^404"/>
  </announcement>
  <announcement comment="update error" media-id="402">
    <play-condition response="^[3-6]" job="put" />
  </announcement>
  <announcement comment="interrogation error" media-id="404">
    <play-condition response="^[3-6]" job="get" />
  </announcement>
  <announcement comment="general fault" media-id="400" />
</announcements>
```

Example 1 Announcement Configuration Example

The `<announcements>` document contains a list of `<announcement>` elements. Each element has a `media-id` attribute defining the announcement code to be played and an optional comment attribute, which can store a remark for the human reader.

The `<announcement>` element includes one or more `<play-condition>` elements describing when the corresponding announcement is played. The conditions are evaluated in the order of the `<announcement>` elements within the document. The `<play-condition>` elements within one announcement are in logical `OR` relationship.

The `<play-condition>` element has the following attributes:

- `id`
- `response`
- `job`
- `type`
- `value`

- `list-length`

Each attribute refers to a property of the received report.xml. The response, the job, and the value attributes can contain the | delimiter. The value pieces are interpreted in a logical `OR` relation.

For example, the `job="put|delete"` expression means that the condition matches, if the performed XCAP operation is `PUT` or `DELETE`.

The attributes of the `<play-condition>` element are interpreted in logical `AND` relation. The condition matches only if all the attributes match their criteria.

3.1.3.4.1 Id Attribute

The `id` attribute refers to the identifier of a `<report>` document. If a report list is created as a result of the performed XCAP operations, then the individual report items are addressed by their unique identifier. The further attributes of the `<play-condition>` element subject to the report identified by the `id` attribute.

In general, the report inherits the identifier from the related task document. For a simple schema the GenSSC engine generates one of the following two kinds of tasks:

- Task created from the action attribute. This task inherits the identifier from the action value (set, delete, and so on).
- Task created from the `serviceState()` function. This task has a prefixed identifier: `service-state`.

In case of composite schema, the creator of the layout document is responsible for the task identification.

3.1.3.4.2 Response Attribute

The response attribute refers to the `<response>` element of a `<report>` document. The `<response>` element includes the HTTP response line of the related XCAP request. The response attribute contains a valid Regex, for example, as follows:

- `response="^200"`– 200 response code
- `response="^202 Accepted"`– 202 accepted response line
- `response="^201|^203"`– 201 or 203 response code
- `response="^4"` – 4xx response code
- `response="^[3-6]"` – not 2xx response code

3.1.3.4.3 Job Attribute

The job attribute refers to the `<job>` element of the report. It refers to the performed XCAP operation, to the validation, or to the data verification job.

The possible values are as follows:

- `get`
- `put`
- `delete`
- `validate`
- `verify`

The job attribute contains a valid Regex, for example, as follows:

- `job="get"`– job is `get`
- `job="put|delete"`– job is `put` or `delete`

3.1.3.4.4 Type Attribute

The type attribute refers to the type attribute of the `<value>` element of the report. The possible values are as follows:

- `empty`
- `simple`
- `complex`

3.1.3.4.5 Value Attribute

The value attribute refers the `<value>` element of the report. It is only allowed if the type attribute refers to the `simple` type. It can contain a valid Regex, for example, as follows:

- `value="true"`– value is `true`
- `value="15|20"` – value is 15 or 20
- `value="1[0-5]"` – value is between 10 and 15

It is used in relation to the `serviceState()` function to decide if the service is active or if it is activated or deactivated.

3.1.3.4.6 List-length Attribute

The `<list-length>` attribute refers to the length of the generated report list. If a single report has been generated, then the list length is zero.

3.1.3.5 mtasGenSscCmdReplaces

The attribute `mtasGenSscCmdReplaces` is used to override the default mapping of the action element in the command syntax and the `state-action` parameter in the `serviceState()` function.

The attribute defines one or two named lists of symbol-text value pairs. It is used to resolve the meaning of an action or state-action value, or both. If the attribute includes more than one list, then they are separated by a semicolon (;), for example, as follows:

- `action: 1=interrogate, 2=verify`
- `state-action: 1=deactivate, 2=activate, 3=interrogate`
- `action: 1=update, 2=create; state-action: 1=activate, 2=deactivate, 3=switch`

3.1.3.6 mtasGenSscCmdUtPath

The attribute `mtasGenSscCmdUtPath` plays different roles in the simple and composite schemas.

3.1.3.6.1 UtPath in Simple Schema

For a simple schema, the UtPath attribute includes the XPath to the corresponding element or attribute within the user simservs.xml document, for example `simservs/communication-diversion/@active`.

If the XPath expression refers to namespaces other than the default `simservs` namespace, the `UtPath` attribute also includes the namespace binding, as follows:

`http://uri.etsi.org/ngn/params/xml/simservs/xcap`

Example

```
simservs/communication-diversion/cp:ruleset/cp:rule[@ID="cfnrc"]/⇒
cp:conditions/mmt-serv:served-identity?xmlns⇒
(cp=urn:ietf:params:xml:ns:common-policy),xmlns(mmt-serv=⇒
http://schemas.ericsson.com/mmtel/services)
```

The namespace binding part begins with a question mark (?). The namespace items are separated by comma.

3.1.3.6.2 UtPath in Composite Schema

For a composite schema, the UtPath attribute can contain either a single XPath expression or a list of XPath items.

For information on format of the single XPath item, see Section 3.1.3.6.1 UtPath in Simple Schema on page 19 (including the namespace binding).

If the UtPath attribute includes multiple XPath items, they must be separated by the `~~` characters. The namespace binding is not used in this case, for example:

```
simservs/communication-diversion~~simservs/⇒
mmt-serv:flexible-identity-presentation
```

Note: The GenSSC engine performs only one XCAP `GET` operation regardless how many XPath items the UtPath attribute contains. For a single XPath expression, the addressed service data item is retrieved by an element or by attribute level XCAP `GET`, therefore the namespace binding must be given.

If multiple XPath items are configured, then the GenSSC engine fetches the whole simservs.xml document and extracts the addressed parts into the individual `service-data` items of the source document. The document level `GET` does not require namespace binding.

3.1.3.7 mtasGenSscCmdContentType

The attribute `mtasGenSscCmdContentType` includes the `Content-Type` of the XCAP request. The possible values are as follows:

- 0 – `empty` (default value)
- 1 – `application/xcap-el+xml`
- 2 – `application/xcap-att+xml`
- 3 – `application/simservs+xml`

3.1.3.8 mtasGenSscCmdLayout

The attribute `mtasGenSscCmdLayout` is used for the composite command configurations. It describes how the SSC command parameters must be inserted into the service data. Usually, it contains an XSLT style sheet. The value attribute is validated against the layout.xsd during the configuration. For more information, refer to *Managed Object Model (MOM)*.

Example 2 shows an example of how to deactivate the top-most Call Forwarding on Busy (CFB) rule.

```
<?xml version="1.0" encoding="UTF-8"?>
<layout version="1" ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <xsl:stylesheet version="1.0" ⇒
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" ⇒
xmlns:gs="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc" ⇒
xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap" ⇒
xmlns:cp="urn:ietf:params:xml:ns:common-policy" ⇒
exclude-result-prefixes="gs">
    <xsl:output method="xml" indent="yes"/>
    <xsl:template match="/">
      <xsl:variable name="rule_id">
        <xsl:value-of select="/gs:source/gs:service-data/⇒
ss:simservs/ss:communication-diversion/⇒
cp:ruleset/cp:rule[cp:conditions/ss:busy]/@id"/>
      </xsl:variable>
      <task id="deactivate-cdiv-rule" version="1" ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc" ⇒
xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap" ⇒
xmlns:cp="urn:ietf:params:xml:ns:common-policy">
        <head>
          <action>set</action>
          <content-type>application/xcap-el+xml</content-type>
          <ut-path>
            /simservs/communication-diversion/cp:ruleset/⇒
cp:rule[@id="<xsl:value-of select='$rule_id'/>"]/⇒
rule-deactivated
            ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)⇒
</ut-path>
        </head>
        <body>
          <ss:rule-deactivated/>
        </body>
      </task>
    </xsl:template>
  </xsl:stylesheet>
</layout>
```

Example 2 Deactivation of Topmost CFB Rule

3.1.3.9 mtasGenSscCmdUriInChargingMsgs

The attribute `mtasGenSscCmdUriInChargingMsgs` defines one or more regular expressions, which specify the rule how the incoming Request-URI is to be translated for charging purposes. The main goal is to mask the PIN value with zero digits in the Request-URI appearing in the charging messages. The rule uses the substitution function of the regular expressions in the format *<translation pattern>/<translation syntax>*.

Example for a URI translation:`/#47#[0-9]{4-8}#/#47#0000#/`

This example specifies the service code and the PIN in the translation pattern. The translation syntax includes the service code and replaces the digits of the PIN with four zeros in the URI.

It is enough if the translation pattern determines the position of the PIN within the SSC command string. The pattern does not need to specify the syntax of the whole SSC URI. That is, it does not need to define the beginning part of the URI or to describe the remaining part. The only task is to identify the location of the PIN unambiguously.

The translation syntax specifies how to modify the URI piece determined by the translation pattern. Typically, it replaces the PIN value with zero digits and copies the other parts.

3.2 Command Syntax Elements

This section includes an overview of the elements used in the command syntax. The `mtasGenSscCmdSyntax` attribute comprises two parts; a command pattern and a command syntax, see Section 3.1.3 MtasGenSscCmd MO on page 13. The command pattern is a Regex matching expression and not detailed here. The command syntax includes the information relevant for the further processing. This part is detailed in the subsequent sections.

The command syntax can include numerous elements. These numbers are separated by the semi-colon sign (`';'`) from each other.

3.2.1 Schema

The GenSSC support two schema types: the simple and the composite schema.

3.2.1.1 Simple Schema

The simple schema is used to configure a single item in the subscriber's service data.

The SSC command string includes a mandatory service code and can include an action code and a single data item. The latter two are optional as the action and the data can be preconfigured in the command syntax. In addition, the command string can also include a PIN.

The command syntax can include the `serviceState()` and `pinCkeck()` functions.

The `serviceState(state-action)` function provides the operator with a special handling for the state of a supplementary service. The service state is stored in the `active` attribute of the service's root element. This attribute can be managed together with a service item in one step using the `serviceState()` function in the command syntax.

The state-action parameter takes the prefix of the SSC command. The prefix holds a special meaning in this case as shown in the Table 1.

Table 1 State-action Prefix of the SSC Command

User Input	Meaning
*sc#	Activates the service.
#sc#	Deactivates the service.
*#sc#	Interrogates the current service state.
#*sc#	Alters the state of the service (activate ← → deactivate).

If the prefix cannot be used for this purpose, then the operator can define a service-state flag in the command string. An example is shown in Table 2.

Table 2 Service-state Flag in the Command String

User Input	Meaning
*sc*0#	Activates the service.
*sc*1#	Deactivates the service.
*sc*2#	Interrogates the current service state.
*sc*3#	Alters the state of the service (activate ← → deactivate).

Both the prefix and the service-state flag values can be redefined by the operator using the `mtasGenSscCmdReplaces` attribute.

A PIN can be also included in the command string. An example for service activation using the `'*'` prefix and a 4-digit pin code looks like the following: `*61*1234#`

Note: The GenSSC provides a `serviceState(state-action)` function in the command syntax. It can be started only in the simple schema. The GenSSC provides a `pinCheck(pin)` function in the command syntax. It can be started in both the simple and the composite schema.

Example of service activation, deactivation, or interrogation using operator defined service-state flag values:

SC command string: `*21*1#`

Command syntax:

```
/^\*21\*([1-3])#$/schema=simple;

function serviceState(\1)/
```

UtPath: `simservs/communication-diversion/@active`

Replaces: `state-action:1=activate, 2=deactivate, 3=interrogate`

3.2.1.2 Composite Schema

The composite schema is used in the following cases:

- Multiple data items are to be configured in one step.
- Query is needed to decide what and how must be updated in the service data.
- The actual content of the service data is evaluated to decide what modification type must be performed by the update.

The composite schema uses the layout.xml document which can include an XSLT style sheet. The command string contains the service code as usual and can include an action, a set of atomic data and can also include a PIN. The command syntax includes the description of the atomic data items and also can include the `pinCheck()` function.

3.2.2 Action

The second element of the command syntax is the action. The action starts a subfunction that enables a user to gain access to, and control of, the supplementary services through the performed actions on the service profile data.

The following actions are supported by the GenSSC:

```
1=update, 2=create, 3=set, 4=switch, 5=delete,
6=interrogate, 7=test, 8=verify.
```

The actions are mapped to XCAP requests as it is shown in Table 3.

Table 3 Mapping of Actions to XCAP Requests

Action	XCAP Request
Update	GET + PUT
Create	
Switch	
Set	PUT
Delete	DELETE
Interrogate	GET
Test	
Verify	

The individual actions have the following meanings:

Update Conditional modification. The service data is modified only if the referred item exists. It is checked in the first step if the item is present in the service profile (`GET`). If yes, then it is updated with the new content (`PUT`) otherwise a response ‘‘404 Element not found’’ is returned in the report.xml.

Create Conditional modification. The service data is modified only if the referred item does not exist. It is checked in the first step if the item is present in the service profile (`GET`). If not, it is created with the new content (PUT) otherwise a response ‘‘409 Element already exists’’ is returned in the report.xml.

Set Unconditional modification. The service data is modified (updated or created) with the new content (`PUT`). A response ‘‘200 OK’’ or ‘‘201 Created’’ is returned in the report.xml.

Switch Content dependent modification. It can be used on boolean data type only. The current value of the referred boolean item is fetched in the first step (`GET`). The value is negated and the service data is updated with the negated content (`PUT`). The response ‘‘200 OK’’ and the negated value are returned in the report.xml.

Delete Unconditional deletion. The referred item is removed from the service data. If the item does not exist, then the response is ‘‘204 No Content’’, if the item cannot be deleted (the removal is prohibited by the application rules), then the response is ‘‘409 Conflict’’, else, the response ‘‘200 OK’’ is returned in report.xml.

Interrogate Fetching a single data piece. Only simple data type is supported. If the item refers to a complex data element (element including further elements), then the response ‘‘409 Conflict’’ is returned in the report.xml. If the query was successful, then the response is ‘‘200 OK’’ and the content of the item is returned.

Test Checking if the item exists. It can be used for any items even for the complex ones. The response ‘‘200 OK’’ and the value `true` or `false` are returned. The value `true` indicates that the referred item exists, `false` indicates that the item does not exist.

Verify Checking if the value of the item is equal to the input value in the URI. This action is used for data verification. If the actual and the input values are equal, then the response is ‘‘200 OK’’, else, the response ‘‘205 Reset Content’’ is returned.

The command syntax can contain the action value in three formats:

- Predefined textual value

  The action value is defined by its name. For example, action=set.

- Predefined numeric value

  The action value is defined by its numeric code. For example, action=3.

- Substitution value

  The action value comes from the Request-URI using the Regex capturing and substitution mechanism. For example, action=\1.

If the action value is defined as a numeric or substitution value, then the GenSSC engine replaces the defined value with its textual representation using the above syntax.

The default encoding can be redefined in the `mtasGenSscCmdReplaces` attribute:

```
action:  0=verify, 1=set
```

If the Extracted Syntax includes a numeric action code after the Regex substitution, then the GenSSC obtains the textual value from the `mtasGenSscCmdReplaces` attribute. It is enough to define only those values in the `mtasGenSscCmdReplaces` attribute which are enabled in the command pattern. If the `mtasGenSscCmdReplaces` does not contain the action values, then the default values are used.

The following example verifies if the `NoReplyTimer` value of the CDIV service matches the user input.

UtPath: `simservs/communication-diversion/NoReplyTimer`

SSC command in the URI: `*65*0*20#`

Replaces: `action: 0=verify, 1=set`

Command syntax:

```
/\*65\*([01])\*([1-9][0-9]*)#

/schema=simple;action=\1;number timeout=\2/
```

Extracted Syntax:

```
schema=simple;action=verify;number timeout=20
```

3.2.3 Data Items

The data items contain the atomic data pieces (Supplementary Information) of the incoming SSC-URI. The command pattern captures and the command syntax store this information in form of data items.

Each data item has the following format in the command syntax:

```
<Type-Description><space><Variable-Name><equal-sign><Value>
```

Example: `number timeout=20`

The GenSSC supports the following data types:

- Boolean
- Number
- Digit string
- Opaque item
- List

3.2.3.1 Boolean

Description	Value
Identifier	`bool`
Value in SC command	`0` or `1`
Value in the Extracted Syntax	`true` or `false`
Example	
SSC command part	`0#`
Command Syntax part	`bool enable=\1`
Extracted Syntax part	`bool enable=false`

3.2.3.2 Number

Description	Value
Identifier	`number`
Value in SC command	String of decimal digits, for example: “20”

Value in the Extracted Syntax	Copied from the SC command string removing the leading zeros. The digits ‘0’-‘9’ are allowed only.
Example	
SC command part	`020`
Command Syntax part	`number timeout=\1`
Extracted Syntax part	`number timeout=20`

3.2.3.3 Digit String

Description	Value
Identifier	`digits`
Value in SC command	String of decadic digits, for example: “6670089”
Value in the Extracted Syntax	Copied from the SC command string. The digits ‘0’-‘9’ are allowed only.
Example	
SC command part	`6670089`
Command Syntax part	`digits target=\1`
Extracted Syntax part	`digits target=6670089`

3.2.3.4 Opaque Item

Description	Value
Identifier	`item`
Value in SC command	String of decadic and over-decadic digits, for example: `18*66700#89`
Value in the Extracted Syntax	Copied from the SC command string without a change or check.
Example	
SC command part	`18*66700#89`
Command Syntax part	`item value=\1`
Extracted Syntax part	`item value=18*66700#89`

3.2.3.5 List

Description	Value
Identifier	The list type does not have an identifier. It can be applied on any base type using the square brackets enclosing an optional separator character as list indicator: <base-type>[<separator>], for example: digits [*]
Comment 1	The list type enables users to define a sequence of data items of a base type separated by the 'separator' sign. The length of the list is calculated from the input or from the predefined value.
Comment 2	If the separator character is not defined, then the input is considered a string of digits where each digit represents a list item.
Comment 3	The value part is enclosed in curly brackets: { … }.
Comment 4	The semi-colon (;) symbol must not be used as list separator since it is reserved for the separation of the command syntax elements.
Value in SC command	List of data items with optional separator, for example: `21976541*22976542*97653`
Value in the Extracted Syntax	List of the individual values, for example: `{21976541 * 22976542 * 97653}`
Comment 5	The extraction creates an internal list object including the individual items. The leading and trailing white spaces are removed from the items by the extraction. The example in the previous row is a textual representation of the list object. The separator character is surrounded by white spaces to illustrate that the items have been separated and stored individually. This format is used in the document.
Example 1: list of identities	
SC command part	`21976541*22976542*97653`
Command Syntax part	`digits[*] identities={\1}`

Extracted Syntax part	`digits[*] identities= {21976541 * 22976542 * 97653}`
Example 2: predefined list of weekdays	
Command Syntax part	`item[,] weekdays={Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}`
Extracted Syntax part	`item[,] weekdays={Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}`
Example 3: MSN alias list, each digit represents an MSN alias – example for list without separator	
SC command part	`025`
Command Syntax part	`number[] aliases={\1}`
Extracted Syntax part	`number[] aliases={0 2 5}`

3.2.4 Functions

The functions start a built-in function of the GenSSC engine. The following functions are supported:

- `serviceState(state-action)`
- `pinCheck(pin)`

3.2.4.1 serviceState() Function

The `serviceState(state-action)` function can be used only in the simple schema.

The `state-action` parameter of the `serviceState()` function contains the following values:

- activate
- deactivate
- interrogate
- switch

The MTAS provides the following default mapping between the prefix/`state-action` and service-state-flag/`state-action` values:

```
state-action: *=activate, #=deactivate, *#=interrogate,
#*=switch, 0=deactivate, 1=activate, 2=interrogate,
3=switch
```

The `mtasGenSscCmdReplaces` attribute can be used to override the default mapping.

Example:

SC command string:`#65#`

Command syntax part:

```
function serviceState(\1)/
```

Extracted Syntax:

```
function serviceState(deactivate)
```

3.2.4.2 pinCheck() Function

The `pinCheck(pin)` function can be used in both the simple and the composite schemas. An example for its configuration is shown in the following example.

SC command string: `*21*5678#`

Value of `mtasGenSscCmdSyntax` attribute:

```
/^([*#]#?)\*21\*([0-9]{4-7})#$
 /schema=service;
   function serviceState(\1);
   function pinCheck(\2)/
```

Extracted command syntax:

```
schema=service;
   function serviceState(activate);
   function pinCheck(5678)
```

4 Configuration of GenSSC Commands

The configuration of the system-defined SSC commands can be described exactly since they are implemented in system-created MOs and form a predefined feature set. In contrast, the GenSSC commands are created by the operator arbitrarily, so they cannot be delineated in a similar extent as the system-defined SSC commands.

This section includes examples of typical configurations of the GenSSC commands.

4.1 Use of Simple Schema

This section includes an example for the use of a simple schema in the GenSSC configuration. The Hotline service is selected since it has small configuration data in the user service profile (simservs.xml) document, see Example 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<ss:simservs ⇒
xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap" ⇒
xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">
  <mmt-serv:hotline active="false">
    <mmt-serv:hotline-number>
      <mmt-serv:stored-number>1122333</mmt-serv:stored-number>
    </mmt-serv:hotline-number>
  </mmt-serv:hotline>
</ss:simservs>
```

Example 3 Representation of Hotline Service Data Within the Simservs.xml Document

4.1.1 Subject of Example

The following user story defines the subject of the example in this section:

An IMS user wants to control and monitor the Hotline service as follows:

- Activate and deactivate the service
- Interrogate the state of the service
- Change the stored number of the service
- Verify the stored number value against an entered value

The same service code (SC=111) must be used for this purpose.

4.1.2 Configuration Overview

This section includes an example configuration of an `MtasGenSscCmd` MO instance, which meets the expectations of the user story, see Section 4.1.1 Subject of Example on page 33.

The Hotline string identifies the following `MtasGenSscCmd` MO instance:

```
MtasGenSscCmd:  Hotline <key>
```

mtasGenSscCmdHeader

```
*111#|#111#|*#111#|*111*|*#111*
```

The command header attribute lists the variations of the leading digits related to this configuration. The first three items are used to maintain the state of the service (activation, deactivation, and interrogation). The fourth item defines the leading digits of the stored number update. The last item includes the leading digits of the data verification task.

mtasGenSscCmdSyntax

```
/^([*#]#?)111#$/schema=simple; function serviceState(\1)/~~/^\*111\⇒
*([0-9]{4-14})#$/schema=simple; action=update; digits st-num=\1;
function serviceState(activate)/~~/^\*#111\*([0-9]{4-14})#$/⇒
schema=simple;action=verify;
digits st-num=\1/
```

The command syntax consists of three syntax items separated by the `~~` characters.

The first syntax item configures the service state. The `serviceState()` function is used for this purpose. For details about the `serviceState()` function, see Section 3.2.4.1 serviceState() Function on page 30.

The pattern of the first command pattern item `^([*#]#?)111#$` shows that the incoming URI must begin with one of the following prefixes: `*, #, *#, ##`. The last (`##`) is filtered out by the command header values. `INVITE` is not considered the subject of this SSC configuration if the URI begins with `##`. The prefix value is captured and used as parameter of the `serviceState()` function later. The pattern also defines that the URI ends with `111#`.

The command syntax part `schema=simple;function serviceState(\1)` defines that it is a simple configuration. That is, no XSLT style sheet is used but the GenSSC engine must generate the appropriate XCAP requests. The syntax says that the `serviceState()` function must be used for the generation. In this case the possible state action values are activate, deactivate, and interrogate. Depending on the state-action value, the GenSSC engine generates an XCAP `PUT` request. This request activates or deactivates the service, or an XCAP `GET` request is generated, which fetches the current value of the service state.

The second syntax item configures the update of the stored number.

The command pattern `^\*111\*([0-9]{4-14})#$` declares that the Request URI must comply with the following:

- It must start with prefix `*111*`.
- It must be followed by a string of 4–14 digits.
- It must end with suffix `#`; the digits are captured and used later.

The command syntax part `schema=simple; action=update; digits st-num=\1;function serviceState(activate)` informs the GenSSC engine that an update action is created. The `st-num` variable contains a digit string coming from the user input. If the update is successful the service is activated. As a result, the engine generates a task list. The first task updates the stored number with the user input and the second activates the service. If the update task fails (for example, if the service is not provisioned for the user or the entered digit string is invalid), then the activation task is not performed.

The third syntax item configures the verification of the stored number.

The command pattern `^\*#111\*([0-9]{4-14})#$` declares that the Request URI must comply with the following:

- It must start with prefix `*111*`.
- It must be followed by a string of 4–14 digits.
- It must end with suffix `#`; the digits are captured and used later.

The command syntax part `schema=simple; action=verify; digits st-num=\1` informs the GenSSC engine that a verification action is created. The `st-num` variable contains a digit string coming from the user input. It is compared with the current value in the service data. The engine generates an XCAP `GET` request, which fetches the current stored number value and verifies it against the user input.

mtasGenSscCmdUtPath

```
simservs/mmt-serv:hotline/mmt-serv:hotline-number⇒
/mmt-serv:stored-number?xmlns(mmt-serv=⇒
http://schemas.ericsson.com/mmtel/services)
```

The `UtPath` attribute points to the `<mmt-serv:stored-number>` element within the simservs.xml document. Since the path refers to namespaces other than the default XCAP namespace (`http://uri.etsi.org/ngn/params/xml/simservs/xcap`), a namespace binding is included to define the namespace of the `mmt-serv` alias.

mtasGenSscCmdContentType

1 – `application/xcap-el+xml`

The attribute includes the XCAP Content-Type of the stored number-related XCAP request. The XCAP request generated from the `serviceState()` function includes the autogenerated `application/xcap-att+xml` content type.

mtasGenSscCmdAnnouncements

The attribute comprises the announcement table related to this configuration. The `<play-condition>` elements refer to the content pieces of the `<report>` document generated from the XCAP result. The `<report>` also reflects the outcome of the task, for example, to the result of the verification. The announcement table is shown in Example 4.

```
<?xml version="1.0" encoding="utf-8"?>
<announcements xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <announcement comment="service not provisioned" media-id="1404">
    <play-condition response="^404"/>
  </announcement>
  <announcement comment="user input error" media-id="1405">
    <play-condition response="^4" job="validate"/>
  </announcement>
  <announcement comment="active" media-id="200">
    <play-condition response="^2" job="get" list-length="0" id="service-state" value="true"/>
          </announcement>
  <announcement comment="inactive" media-id="201">
    <play-condition response="^2" job="get" list-length="0" id="service-state" value="false"/>
          </announcement>
  <announcement comment="successfully activated" media-id="202">
    <play-condition response="^2" job="put" list-length="0" id="service-state" value="true"/>
          </announcement>
  <announcement comment="successfully deactivated" media-id="203">
    <play-condition response="^2" job="put" list-length="0" id="service-state" value="false"/>
  </announcement>
  <announcement comment="successfully updated" media-id="204">
    <play-condition response="^2" job="put" list-length="2" id="update"/>
  </announcement>
  <announcement comment="data-mismatch" media-id="205">
    <play-condition response="^205" job="verify"/>
  </announcement>
  <announcement comment="data-match" media-id="206">
    <play-condition response="^200" job="verify"/>
  </announcement>
  <announcement comment="internal error" media-id="1800"/>
</announcements>
```

Example 4 Announcement Table of Hotline Service Configuration

Other Attributes

The attributes `mtasGenSscCmdReplace`, `mtasGenSscCmdUriInCharging Msgs`, and `mtasGenSscCmdLayout` of the `MtasGenSscCmd` MO are not used in this configuration; default values are assigned to them.

4.1.3 Workflow Overview

The configuration in Section 4.1.2 Configuration Overview on page 34 covers five use cases; activation, deactivation, and interrogation of the service state as well as update and verification of the stored number value. Two of these use cases, the interrogation of the service state and the update of the stored number value are described in this section.

Before reading the examples in this section, study the following XML schemas for detailed information:

- ETSI 300738, Man-machine Interface (MMI) to Public Network based Supplementary Services
- RFC 2616, Hypertext Transfer Protocol – HTTP/1.1
- RFC 4826, The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)
- W3C XML 1.0, Extensible Markup Language (XML) 1.0
- W3C Canonical XML, Canonical XML Version 1.0
- XML Path Language (XPath) Version 1.0
- XSL Transformations (XSLT) Version 1.0
- GNU Regular Expression Extensions

The generation of charging messages and the incrementation of the appropriate PM counters are not detailed in the examples.

4.1.3.1 Interrogation of Hotline Service State

This example illustrates the following:

- How the configuration data is mapped to the `<key>` document.
- How the XCAP requests are created from the `<task>`, that is, how the information in the task is put to the XCAP request.
- How the XCAP response is processed, that is, how the `<report>` document is generated from the XCAP response and the `<task>` document.
- How the announcement to be played is determined from the `<report>`.

The Request-URI contains the command string `*#111#`.

The first syntax item is selected. The extracted syntax contains the expression `schema=simple;function serviceState(interrogate)`.

The GenSSC engine generates a `<task>` document shown in Example 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<task id="service-state" version="1" ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
 <head>
  <action>interrogate</action>
  <content-type>application/xcap-att+xml</content-type>
  <ut-path>simservs/mmt-serv:hotline/@active
    ?xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
  </ut-path>
 </head>
</task>
```

Example 5 Task Document – Interrogation of State of Hotline Service

A `<task>` or a `<task-list>` document is always created if the user data must be modified. It is either generated by the GenSSC engine (simple schema) or extracted from the XSL transformation (composite schema). In the latter case, the operator defines the XSLT style sheet so that the output must be a `<task>` or a `<task-list>` document.

Note: There are use cases that do not require any change in the user data. It is then enough to create a proper report determining the announcement to be played to the user. For example, if the user tries to deactivate an already deactivated rule, then there is nothing to do with the user data. Instead, a voice message is sent to the user.

The task is the base of the further processing. The GenSSC engine processes the tasks and generates XCAP requests from them. With Example 5 as example, an XCAP `GET` request shown in Example 6 is sent to the XCAP servlet.

```
GET
/mtasxdms/simservs.ngn.etsi.org/users/sip:abc@ericsson.com/⇒
simservs.xml/~~/simservs/mmt-serv:hotline/@active
?xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services) HTTP/1.1
x-3gpp-asserted-identity: "sip:abc@ericsson.com"
Host: localhost:8090
Content-Length: 0
Connection: Keep-Alive
```

Example 6 XCAP GET Request to XCAP Servlet – Interrogation of State of Hotline Service

The XCAP servlet sends back the response shown in Example 7.

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache
ETag: 0
Content-Type: application/xcap-att+xml
Content-Length: 7
Date: Wed, 12 Dec 2012 15:56:40 GMT

"false"
```

Example 7 XCAP GET Response from XCAP Servlet – Interrogation of State of Hotline Service

On receipt of the XCAP `GET` response, the GenSSC engine generates a report shown in Example 8.

```
<?xml version="1.0" encoding="UTF-8"?>
<report version="1" id="service-state"⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <response>200 OK</response>
  <job>get</job>
  <value type="simple">false</value>
</report>
```

Example 8 Report Document – Interrogation of State of Hotline Service

In the last step, the announcement table (see Example 4) is evaluated against the `<report>` document. The following item matches:

```
<announcement comment="inactive" media-id="201">
   <play-condition response="^2" job="get" list-length="0"⇒
id="service-state" value="false"/>
</announcement>
```

The announcement “inactive” is sent to the user.

4.1.3.2 Update of Hotline Stored Number

The goal of this example is to illustrate the following:

- How the conditional actions work
- How the `<task-list>` is processed
- What report types can be generated during the processing of a `<task-list>`

The Request-URI contains the command string `*111*1122666#`.

The second syntax item is selected. The extracted syntax contains the following expression:

```
schema=simple; action=update; digits st-num=1122666;funct
ion serviceState(activate)
```

The GenSSC engine recognizes that the following two tasks must be done:

- The stored number must be updated.
- The service must be activated.

Therefore, the <task-list> document in Example 9 is generated.

```
<?xml version="1.0" encoding="UTF-8"?>
<task-list version="1"⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <task id="update" version="1">
  <head>
   <action>update</action>
   <content-type>application/xcap-el+xml</content-type>
   <ut-path>simservs/mmt-serv:hotline/mmt-serv:hotline-number/⇒
mmt-serv:stored-number
     ?xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
   </ut-path>
  </head>
  <body>1122666</body>
 </task>
 <task id="service-state" version="1">
  <head>
   <action>set</action>
   <content-type>application/xcap-att+xml</content-type>
   <ut-path>simservs/mmt-serv:hotline/@active
     ?xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
   </ut-path>
  </head>
  <body>true</body>
 </task>
</task-list>
```

Example 9 Task-list Document – Update of Hotline Stored Number

Taking the first task from the list, the GenSSC engine recognizes that it includes a conditional action: the update. It means that only an existing element can be modified. That is, an XCAP GET request as shown in Example 10 is sent to the XCAP servlet first.

```
GET
/mtasxdms/simservs.ngn.etsi.org/users/sip:abc@ericsson.com/⇒
simservs.xml/~~/simservs/mmt-serv:hotline/⇒
mmt-serv:hotline-number/mmt-serv:stored-number
?xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
HTTP/1.1
x-3gpp-asserted-identity: "sip:abc@ericsson.com"
Host: localhost:8090
Content-Length: 0
Connection: Keep-Alive
```

Example 10 XCAP GET Request to XCAP Servlet – Update of Hotline Stored Number

The XCAP servlet sends back one of the responses shown in Example 11 and Example 12.

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache
ETag: 0
Content-Type: application/xcap-el+xml
Content-Length: nnn
Date: Wed, 12 Dec 2012 15:56:40 GMT

<mmt-serv:stored-number>1122333</mmt-serv:stored-number>
```

Example 11 XCAP GET Positive Response – Update of Hotline Stored Number

```
HTTP/1.1 404 Not Found
Server: Apache-Coyote/1.1
Cache-Control: no-cache
ETag: 0
Content-Length: 0
Date: Wed, 12 Dec 2012 15:56:40 GMT
```

Example 12 XCAP GET Negative Response – Update of Hotline Stored Number

If there is a negative response, the processing of the `<task-list>` is interrupted and a negative report is generated, see Example 13.

```
<?xml version="1.0" encoding="UTF-8"?>
<report version="1" id="update"⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <response>404 Not Found</response>
  <job>get</job>
  <value type="empty"/>
</report>
```

Example 13 Negative Report – Update of Hotline Stored Number

On receipt of a positive response, the GenSSC engine sends an XCAP `PUT` request to the XCAP servlet, see Example 14.

```
PUT
/mtasxdms/simservs.ngn.etsi.org/users/sip:abc@ericsson.com/⇒
simservs.xml/~~/simservs/mmt-serv:hotline/⇒
mmt-serv:hotline-number/mmt-serv:stored-number
?xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services) HTTP/1.1
x-3gpp-asserted-identity: "sip:abc@ericsson.com"
Host: localhost:8090
Connection: Keep-Alive
Content-Type: application/xcap-el+xml
Content-Length: nnn

<mmt-serv:stored-number>1122666</mmt-serv:stored-number>
```

Example 14 XCAP PUT Request to XCAP Servlet – Update of Hotline Stored Number

An XCAP PUT response is sent in return, see Example 15.

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache
ETag: 1
Content-Length: 0
Date: Wed, 12 Dec 2012 15:56:40 GMT
```

Example 15 XCAP PUT Response – Update of Hotline Stored Number

When the first task finishes successfully, the GenSSC engine creates a `<report-list>` document and adds the first report to the list as shown in Example 16.

```
<?xml version="1.0" encoding="UTF-8"?>
<report-list version="1" list-length="1"⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <report version="1" id="update"
    <response>200 OK</response>
    <job>put</job>
    <value type="simple">1122666</value>
  </report>
</report-list>
```

Example 16 Report-list Document – Update of Hotline Stored Number

The processing of the `<task-list>` continues with the second item. The XCAP `PUT` request shown in Example 17 is sent to the XCAP servlet.

```
PUT
/mtasxdms/simservs.ngn.etsi.org/users/sip:abc@ericsson.com/⇒
simservs.xml/~~/simservs/mmt-serv:hotline/@active
?xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
HTTP/1.1
x-3gpp-asserted-identity: "sip:abc@ericsson.com"
Host: localhost:8090
Connection: Keep-Alive
Content-Type: application/xcap-att+xml
Content-Length: 6

"true"
```

Example 17 XCAP PUT Request to XCAP Servlet – Activation of Service

The XCAP response shown in Example 18 is returned.

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache
ETag: 2
Content-Length: 0
Date: Wed, 12 Dec 2012 15:56:40 GMT
```

Example 18 XCAP PUT Response – Activation of Service

On receipt of the XCAP response, a new report is added to the report list, see Example 19.

```
<?xml version="1.0" encoding="UTF-8"?>
<report-list version="1" list-length="2"⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <report version="1" id="update"
    <response>200 OK</response>
    <job>put</job>
    <value type="simple">1122666</value>
  </report>
  <report version="1" id="service-state"
    <response>200 OK</response>
    <job>put</job>
    <value type="simple">true</value>
  </report>
</report-list>
```

Example 19 Report Document – Interrogation of State of Hotline Service

In the last step, the announcement table (Example 4) is evaluated against the `<report-list>` document. The following item matches:

```
<announcement comment="successfully updated" media-id="204">
  <play-condition response="^2" job="put" list-length="2" id="update"/>
</announcement>
```

The announcement ‘‘successfully updated’’ is sent to the user.

4.2 Composite Configuration Using Task Pattern

The composite configuration can be used for various purposes. One of them is to define a task pattern describing how the items of the SSC command string must be inserted into the service data. This section illustrates a pattern for a task document, which copies multiple items from the user input to the task body. The example in this section shows how the list data type can be used in the SSC command configuration.

The condition `<mmt-serv:valid-days>` of a fictive ‘‘out-of-office’’ CDIV rule is used for this purpose. It can contain multiple `<mmt-serv:day>` elements, which are created from the user input. An example of the ‘‘out-of-office’’ CDIV rule representation within the `<simservs>` document is shown in Example 20.

```
<?xml version="1.0" encoding="UTF-8"?>
<ss:simservs ⇒
xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap" ⇒
xmlns:cp="urn:ietf:params:xml:ns:common-policy" ⇒
xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">
  <ss:communication-diversion>
    <cp:ruleset>
      <cp:rule id="out-of-office">
        <cp:conditions>
          <mmt-serv:valid-periods>
            <mmt-serv:valid-days>
              <mmt-serv:day>Wednesday</mmt-serv:day>
              <mmt-serv:day>Saturday</mmt-serv:day>
              <mmt-serv:day>Sunday</mmt-serv:day>
            </mmt-serv:valid-days>
          </mmt-serv:valid-periods>
        </cp:conditions>
        <cp:actions>
          <ss:forward-to>
            <ss:target>sip:voice-mail@telco.com</ss:target>
            <ss:notify-caller>true</ss:notify-caller>
          </ss:forward-to>
        </cp:actions>
      </cp:rule>
    </cp:ruleset>
  </ss:communication-diversion>
</ss:simservs>
```

Example 20 Representation of Out-of-Office CDIV Rule Within the simservs.xml Document

4.2.1 Subject of Example

The following user story defines the subject of the example in this section:

An IMS user wants to configure the ‘‘out-of-office’’ CDIV rule defining the list of weekdays when the incoming calls are forwarded to the voice mailbox.

The following SSC command format is to be used:

```
*<service-code=42>*<list-of-weekdays>#
```

The days are represented by digits 0–6, digit 0 stands for Sunday, digit 1 for Monday, digit 2 for Tuesday, and so on.

The days may or may not be separated by a star symbol:

- Alternate 1: *42*056#
- Alternate 2: *42*0*5*6#

4.2.2 Configuration Overview

This section includes an example configuration of an `MtasGenSscCmd` MO instance that is in line with the user story, see Section 4.2.1 Subject of Example on page 44.

MtasGenSscCmd

The `CdivOutOfOffice` string identifies the following `MtasGenSscCmd` MO instance:

`CdivOutOfOffice` *`<key>`*

mtasGenSscCmdHeader

```
*42*
```

The command header attribute defines that the SSC command must begin with the `*42*` string.

mtasGenSscCmdSyntax

```
/^\*42\*([0-6]+)#$/schema=composite; number[] days=\1;
item[,] weekdays={Sunday, Monday, Tuesday, Wednesday,
Thursday, Friday, Saturday}/~~

/^\*42\*([0-6*]+)#$/schema=composite; number[*] days=\1;
item[,] weekdays={Sunday, Monday, Tuesday, Wednesday,
Thursday, Friday, Saturday}/
```

The syntax definition includes two items since the SSC command can have two formats.

The first syntax item stands for alternate 1 in the user story where there is no separator between the day items.

The command pattern `^\*42\*([0-6]+)#$` declares that the Request URI must comply with the following:

- It must start with prefix `*42*`.
- It must be followed by one or more digits in range of 0–6; the digits are captured and used later.
- It must end with suffix `#`.

The command pattern matches the SSC command of the alternate 1 format.

The following command syntax part shows the GenSSC engine that a composite configuration is to be handled:

```
schema=composite; number[] days=\1; item[,] weekdays=
Sunday, Monday, Tuesday, Wednesday, Thursday, Friday,
Saturday}
```

The first data item (`days`) is a list of numbers. Since the list definition (`[]`) does not contain a separator character, it is considered a list of digits, each digit representing a number value.

The second data item (`weekdays`) is a predefined list of free-text items separated by commas.

The engine extracts the command syntax part in the first step. The user input in alternate 1 is taken in this example but it varies on the entered digits. The digit string is divided into individual digits and the numeric values are stored in the list of days. The weekdays are stored in a list object also during the extraction.

The extracted syntax contains the following expression:

```
schema=composite; number[] days={0 5 6};item[,]
weekdays={Sunday , Monday , Tuesday , Wednesday , Thursday
, Friday , Saturday}
```

The second syntax item stands for alternate 2 in the user story where there is a star separator between the day items.

The command pattern `^\*42\*([0-6*]+)#$` declares that the Request URI must comply with the following:

- It must start with prefix `*42*`.
- It must be followed by one or more digits in range of 0–7 separated by the star symbol; the whole string including the digits and the star symbols is captured and used later.
- It must end with suffix `#`.

The command pattern matches the SSC command of the alternate 2 format.

The following command syntax part is almost the same as in the first item:

```
schema=composite; number[*] days=\1; item[,] weekd
ays={Sunday, Monday, Tuesday, Wednesday, Thursday,
Friday,Saturday}
```

The only difference is that the days are separated by star symbols.

When the engine extracts the command syntax, the captured string is divided into individual digits using the star symbol as separator. The extracted syntax contains the following expression:

```
schema=composite; number[*] days={0 * 5 * 6}; item[,]
weekdays= {Sunday , Monday , Tuesday , Wednesday ,
Thursday , Friday , Saturday}
```

Thus, the extracted syntax is the same in both alternatives. The command syntax processing eliminates the differences of the two variants. The `<command-data>` element includes the same format. A common XSL style sheet can be used for both alternatives.

mtasGenSscCmdAnnouncements

The announcement table for the update of the out-of-office rule is shown in Example 21.

```
<?xml version="1.0" encoding="utf-8"?>
<announcements xmlns="http://uri.etsi.org/ngn/params/xml/simservs⇒
/xcap/genssc">
  <announcement comment="user input error" media-id="1405">
    <play-condition response="^4"/>
  </announcement>
  <announcement comment="successfully updated" media-id="204">
    <play-condition response="^2"/>
  </announcement>
  <announcement comment="update error" media-id="1800"/>
</announcements>
```

Example 21 Announcement Configuration – Update of CDIV Out-of-office Rule

It is a simple case; the update is either successful or fails owing to a user or system error.

mtasGenSscCmdLayout

For the layout document description, see Section 4.2.3.2 Layout Implementation on page 49.

Other Attributes

The attributes `mtasGenSscCmdUtPath`, `mtasGenSscCmdReplaces`, `mtasGenSscCmdUriInChargingMessages`, and `mtasGenSscCmdContentType`

of the MtasGenSscCmd MO are not used in this configuration, default values are assigned to them.

4.2.3 Workflow Overview

This section describes the workflow overview.

4.2.3.1 Preparations

After the command syntax has been extracted, the GenSSC engine generates a source document. The contents of the extracted syntax has been added to the <command-data> element of the source document. The generated source document is shown in Example 22, comments are included for better understanding.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This document is passed to the XSL Transform function as
     the source XML of the transformation. -->
<source version="1
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <!-- The <user-id> comes from the P-Asserted-Identity header
       of the SSC-INVITE. -->
  <user-id>sip:abc@ericsson.com</user-id>
  <!-- The <command-data> contains the elements of the extracted
       command syntax.
       It includes the schema type and the two lists in this case. -->
  <command-data version="1">
    <schema>composite</schema>
    <list name="days" length="3" base-type="number">
      <item index="0">0</item>
      <item index="1">5</item>
      <item index="2">6</item>
    </list>
    <list name="weekdays" length="7" base-type="item">
      <item index="0">Sunday</item>
      <item index="1">Monday</item>
      <item index="2">Tuesday</item>
      <item index="3">Wednesday</item>
      <item index="4">Thursday</item>
      <item index="5">Friday</item>
      <item index="6">Saturday</item>
    </list>
  </command-data>
```

Example 22 Generated <source> Document – Update of CDIV Out-of-Office Rule

4.2.3.2 Layout Implementation

This section includes an example for the composite configuration where a <task> document is used to define the pattern how the days are to be inserted in the out-of-office rule, see Example 23.

```
<?xml version="1.0" encoding="UTF-8"?>
<layout version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <xsl:stylesheet version="1.0" ⇒
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"⇒
xmlns:gs="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc" ⇒
xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap" ⇒
xmlns:cp="urn:ietf:params:xml:ns:common-policy" ⇒
xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services" ⇒
exclude-result-prefixes="gs">
    <xsl:output method="xml" indent="yes"/>
    <xsl:template match="/">
      <task id="out-of-office" version="1"
          xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
          xmlns:cp="urn:ietf:params:xml:ns:common-policy"
          xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">
        <head>
          <action>set</action>
          <content-type>application/xcap-el+xml</content-type>
          <ut-path>simservs/communication-diversion/cp:ruleset/cp:rule[@id="out-of-office"]
            ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)⇒
          </ut-path>
        </head>
        <body>
          <cp:rule id="out-of-office">
            <cp:conditions>
              <mmt-serv:valid-periods>
                <mmt-serv:valid-days>
                  <xsl:for-each select="/gs:source/gs:command-data/gs:list[@name='days']/gs:item">
                    <xsl:variable name="day">
                      <xsl:value-of select="text()"/>
                    </xsl:variable>
                    <mmt-serv:day>
                      <xsl:value-of
                          select="/gs:source/gs:command-data/gs:list[@name='weekdays'] ⇒
/gs:item[@index=$day]"/>
                    </mmt-serv:day>
                  </xsl:for-each>
                </mmt-serv:valid-days>
              </mmt-serv:valid-periods>
            </cp:conditions>
            <cp:actions>
              <ss:forward-to>
                <ss:target>sip:voice-mail@telco.com</ss:target>
                <ss:notify-caller>true</ss:notify-caller>
              </ss:forward-to>
            </cp:actions>
          </cp:rule>
        </body>
      </task>
    </xsl:template>
  </xsl:stylesheet>
</layout>
```

Example 23 Layout Configuration – Update of CDIV Out-of-Office Rule

The GenSSC engine takes the <xsl:stylesheet> element of the layout and performs the XSL transformations defined in the style sheet. The engine uses the <source> document as the source of the transformation.

The <xsl:stylesheet> element defines the following two namespace attributes:

- xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

The XSLT elements and functions can be accessed through this namespace.

- `xmlns:gs="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"`

  The elements and attributes of the `<source>` document can be addressed by this namespace. This is the default namespace of the `<source>` document but the XSLT cannot use the default namespace of the source. Therefore, the namespace must be defined using an alias. The alias value `gs` is recommended for the GenSSC namespace.

The third attribute of the `<xsl:stylesheet>` element tells the XSLT engine that the namespace declaration of the `gs` alias must be excluded from the output. It is used only by the XSLT elements and functions. The start tag of the `<task>` document lists all the namespaces used in the `<body>` later, except the `gs` namespace, which is not part of the output.

The namespace list in the root element of the output is an important and mandatory point. If it is missing, then the XSLT engine cannot create a well-formed XML.

Certain namespaces used in the output may be needed in XSL variable definitions before the output root is defined. These namespaces must be declared in the `<xsl:stylesheet>` element but must also be declared in the output root element.

If such a namespace is not defined in the output root, then the XSLT engine inserts the declaration at its first occurrence in the `<body>`. It makes the simservs.xml document full of redundant namespace definitions. As these namespaces are already defined in the simservs root element, the `<ut-path>` in the `<head>` element points to the position where the out-of-office rule is to be inserted into the simservs.xml document.

The `<body>` includes the pattern of the rule to be inserted into the simservs.xml document. There is a for-each cycle in the pattern, which creates the list of `<mmt-serv:day>` items within the `<mmt-serv:valid-days>` element.

Taking the source document of Example 22, the XSL transformation generates the `<task>` document shown in Example 24.

```
<?xml version="1.0" encoding="UTF-8"?>
<task version="1" id="out-of-office" ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc" ⇒
xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap" ⇒
xmlns:cp="urn:ietf:params:xml:ns:common-policy" ⇒
xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">
  <head>
    <action>set</action>
    <content-type>application/xcap-el+xml</content-type>
    <ut-path>/simservs/communication-diversion/cp:ruleset/⇒
cp:rule[@id="out-of-office"]
       ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
    </ut-path>
  </head>
  <body>
    <cp:rule id="out-of-office">
      <cp:conditions>
        <mmt-serv:valid-periods>
          <mmt-serv:valid-days>
            <mmt-serv:day>Sunday</mmt-serv:day>
            <mmt-serv:day>Friday</mmt-serv:day>
            <mmt-serv:day>Saturday</mmt-serv:day>
          </mmt-serv:valid-days>
        </mmt-serv:valid-periods>
      </cp:conditions>
      <cp:actions>
        <ss:forward-to>
          <ss:target>sip:voice-mail@telco.com</ss:target>
          <ss:notify-caller>true</ss:notify-caller>
        </ss:forward-to>
      </cp:actions>
    </cp:rule>
  </body>
</task>
```

Example 24 Generated Task – Update of CDIV Out-of-Office Rule

If the service data update is successful, the GenSSC engine generates the report shown in Example 25.

```
<?xml version="1.0" encoding="UTF-8"?>
<report version="1" id="out-of-office ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <response>200 OK</response>
  <job>put</job>
  <value type="complex"/>
</report>
```

Example 25 Report Generated for Successful Update of CDIV Out-of-Office Rule

The announcement `id="204"` (“successfully updated”) is sent to the user, see Example 21.

4.3 Composite Configuration Including Pre-Processing Logic and Various Use Cases

This section contains an example for a composite configuration, which analyzes the service data in the first step and according to the actual content either updates the service data or keeps it unchanged. The configuration plays an appropriate announcement to the user at the end. The example deactivates the top-most Call Forwarding on Busy (CFB) or Call Forwarding on Busy to Voice Mail (CFBVM) rule according to their relative position. These rules are members of the CDIV rule set, see Example 26.

```
ss:simservs ⇒
xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap" ⇒
xmlns:cp="urn:ietf:params:xml:ns:common-policy">
  <ss:communication-diversion active="true">
    <cp:ruleset>
      <cp:rule id="cdiv-busy-vm">
        <cp:conditions>
          <ss:busy/>
        </cp:conditions>
        <cp:actions>
          <ss:forward-to>
            <ss:target>voicemail:internal</ss:target>
            <ss:reveal-identity-to-caller>
              true
            </ss:reveal-identity-to-caller>
          </ss:forward-to>
        </cp:actions>
      </cp:rule>
      <cp:rule id="cdiv-anonymous">
        <cp:conditions>
          <ss:anonymous/>
        </cp:conditions>
        <cp:actions>
          <ss:forward-to>
            <ss:target>sip:blunt-message@example.com</ss:target>
          </ss:forward-to>
        </cp:actions>
      </cp:rule>
      <cp:rule id="cdiv-busy">
        <cp:conditions>
          <ss:busy/>
          <ss:rule-deactivated/>
        </cp:conditions>
        <cp:actions>
          <ss:forward-to>
            <ss:target>sip:secretary@example.com</ss:target>
            <ss:reveal-identity-to-caller>
              false
            </ss:reveal-identity-to-caller>
          </ss:forward-to>
        </cp:actions>
      </cp:rule>
    </cp:ruleset>
  </ss:communication-diversion>
</ss:simservs>
```

Example 26 Simservs.xml Document Including CFBVM and CFB Rules

4.3.1 Subject of Example

The following user story defines the subject of the example in this section:

An IMS user wants to deactivate the top-most of CFB or CFBVM rules according to their relative position using SSC command #67#.

The user wants to be informed of the following:

- Which rule has been deactivated
- If the other rule has been implicitly triggered by the deactivation

4.3.2 Configuration Overview

This section describes an example configuration of an `MtasGenSscCmd` MO instance, which is in line with the previous description.

The `CfbVmDeact` string identifies the following `MtasGenSscCmd` MO instance:

`MtasGenSscCmd: CfbVmDeact <key>`

mtasGenSscCmdHeader

`#67#`

The command header attribute includes the SSC command of the user story.

mtasGenSscCmdSyntax

`/^#67#$/schema=composite/`

The command pattern `^#67#$` declares the format of the Request URI.

The command syntax part `schema=composite;` informs the GenSSC engine that a composite configuration is to be processed. It means that the task or report is extracted from the XSLT style sheet included in the layout.xml document of the `mtasGenSscCmdLayout` attribute.

mtasGenSscCmdUtPath

```
simservs/communication-diversion/cp:ruleset
?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
```

The `UtPath` attribute has a special meaning in the composite schema. It contains a list of XPath expressions. Each list item points to one or more elements in the simservs.xml document. The referred elements are copied from the `<simservs>` document to the `<service-data>` elements of the `<source>` document. For details, see Section 4.3.3 Workflow Overview on page 55.

The `UtPath` attribute points to the `<cp:ruleset>` element of the CDIV service.

mtasGenSscCmdAnnouncements

For the announcement table, see Section 4.3.3.3 Announcement Configuration on page 58.

mtasGenSscCmdLayout

For the layout document, see Section 4.3.3.6 Layout Implementation on page 61.

Other Attributes

The attributes `mtasGenSscCmdReplace`, `mtasGenSscCmdContentType`, and `mtasGenSscCmdUriInChargingMessages` of the `MtasGenSscCmd` MO are not used in this configuration, default values are assigned to them.

4.3.3 Workflow Overview

This section describes the workflow overview.

4.3.3.1 Understanding Details

The user story seems to be simple but the details can be difficult to understand. Analyze the service data pieces related to the CFB/CVBVM rules to understand the technical details and to discover the optional difficulties.

A CDIV rule is of type ‘‘forwarding on busy’’ if its condition includes the `<ss:busy/>` element.

A rule is of type ‘‘forwarding to voice mail’’ if its forward-to target contains a ‘‘voicemail’’ string. It means, if the forward-to target value is `voicemail:internal` or `voicemail@example.com`, and so on, then the rule is considered of type ‘‘forwarding to voice mail’’.

A rule can be deactivated by inserting an `<ss:rule-deactivated/>` element into the rule condition or can be deleted by removing the rule from the rule set. These different technical solutions result in the same user experience, that is, the rule is not active. Therefore, a rule is active if it exists and is not deactivated. Deactivation of an inactive (deactivated or deleted) rule is considered a successful attempt.

The ‘‘busy’’ condition can be combined with other conditions like ‘‘validity’’ and ‘‘identity’’ but this example focuses on the simple busy rules, the combined rules are out of the scope. In theory, the CDIV rule set can have more than one simple CFB or CFBVM rule. This case can be considered a provisioning error and the user is notified about it. The deactivated simple rules must also be taken into the account.

It must be ensured that the service data contains maximum one active or deactivated simple CFB, and maximum one active or deactivated simple CFBVM rule. In other cases, a negative voice message is sent to the user.

4.3.3.2 Clarification of Use Cases

The user service profile includes the following information:

- Which rules are provisioned for the user
- Which rules are active/deactivated
- The relative order of the rules, that is, which is upper/lower in the rule set

This information defines the use cases of the user story. They are the input parameters, which help to determine the following:

- The rule to be deactivated
- The announcement to be played to the user

That is, not only the service data must be modified but the user must be provided with a meaningful announcement also.

Table 4 summarizes the use cases of the individual SSC commands and determine the rules to be deactivated and the announcement to be played for each use case.

Table 4 Use Cases of SSC Command for Deactivating Top Busy Rule

SSC Command: #67#			Deactivates Top Busy Rule		
Use Case	**Provisioned Rule**	**Active Rule**	**Top Rule**	**Rule to Be Deactivated**	**Announcement**
UC-0	More than one simple CFBVM or CFB rule	Irrelevant	Irrelevant	None	“Provisioning error”
UC-1	Irrelevant but valid	None	Irrelevant	None	“Both CFB and CFBVM are already inactive”
UC-2	CFB	CFB	CFB	CFB	“CFB has been deactivated, CFBVM is already inactive”
UC-3	CFBVM	CFBVM	CFBVM	CFBVM	“CFBVM has been deactivated, CFB is already inactive”
UC-4	Both	CFB	CFB	CFB	“CFB has been deactivated, CFBVM is already inactive”

Table 4 Use Cases of SSC Command for Deactivating Top Busy Rule

SSC Command: #67#			Deactivates Top Busy Rule		
Use Case	**Provisioned Rule**	**Active Rule**	**Top Rule**	**Rule to Be Deactivated**	**Announcement**
UC-5	Both	CFB	CFBVM	None	‘‘CFBVM is already inactive, CFB is triggered’’
UC-6	Both	CFBVM	CFBVM	CFBVM	‘‘CFBVM has been deactivated, CFB is already inactive’’
UC-7	Both	CFBVM	CFB	None	‘‘CFB is already inactive, CFBVM is triggered’’
UC-8	Both	Both	CFB	CFB	‘‘CFB has been deactivated, CFBVM will be triggered’’
UC-9	Both	Both	CFBVM	CFBVM	‘‘CFBVM has been deactivated, CFB will be triggered’’

A summary of the announcements is shown in Table 5.

Table 5 Summary of Announcements – CFB/CFBVM Configuration

Announcement	Media ID	Use Cases
‘‘Provisioning error’’	800	0
‘‘CFB is already inactive, CFBVM is triggered’’	230	7
‘‘CFBVM is already inactive, CFB is triggered’’	231	5
‘‘Both CFB and CFBVM are already inactive’’	232	1
‘‘CFB has been deactivated, CFBVM is already inactive’’	220	2, 4
‘‘CFBVM has been deactivated, CFB is already inactive’’	221	3, 6
‘‘CFB has been deactivated, CFBVM will be triggered’’	222	8
‘‘CFBVM has been deactivated, CFB will be triggered’’	223	9

For the use cases in Table 5, the following apply:

- Use case 0 represents provisioning error cases. The service data is invalid and must not be modified. An appropriate report is to be created, which determines the announcement to be played.

- Use cases 1, 5, and 7 represent dummy tasks. The service data must not be changed since the current value and the new value are the same. It is enough to create a proper report.
- Use cases 2, 3, 4, 6, 8, and 9 represent real tasks. The service data must be modified according to the use case.

Note: The clarification of the use cases is an important step of the workflow. It can be omitted only if the task defined by the user story is trivial like the one in Section 4.2 Composite Configuration Using Task Pattern on page 44.

The layout creates a negative `<report>` document for use case 0, a positive `<report>` for use cases 1, 5, and 7, and generates a `<task>` document for use cases 2, 3, 4, 6, 8, and 9.

4.3.3.3 Announcement Configuration

The layout is to be constructed so it performs the required modifications on the service data and provides the necessary information for the announcement.

The use cases and the corresponding announcements must be determined before the layout implementation. The layout design often starts with the creation of the announcement table.

A convention is used in this example: the `id` attribute of the tasks and reports is constructed from a `d` letter (referring to the deactivation) and from the numeric ID of the use case shown in Table 4. This trick simplifies the configuration of the announcement table in Example 27.

Note: The `id` attribute is a good place to provide extra information about the result of the XSL transformation. This information can be used in the `<play-condition>` elements of the announcement table.

```
<?xml version="1.0" encoding="utf-8"?>
<announcements xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <!-- Provisioning error. -->
  <announcement comment="Provisioning error" media-id="800">
          <play-condition id="d0"/>
  </announcement>
  <!-- Dummy tasks. -->
  <announcement comment="CFB is already inactive, CFBVM is triggered" media-id="230">
          <play-condition id="d7"/>
  </announcement>
  <announcement comment="CFBVM is already inactive, CFB is triggered" media-id="231">
        <play-condition id="d5"/>
  </announcement>
  <announcement comment="Both CFB and CFBVM are already inactive" media-id="232">
        <play-condition id="d1"/>
  </announcement>
  <!-- Real tasks. -->
  <announcement comment="CFB has been deactivated, CFBVM is already inactive" media-id="220">
          <play-condition id="d2|d4" response="^2"/>
  </announcement>
  <announcement comment="CFBVM has been deactivated, CFB is already inactive" media-id="221">
          <play-condition id="d3|d6" response="^2"/>
  </announcement>
  <announcement comment="CFB has been deactivated, CFBVM will be triggered" media-id="222">
          <play-condition id="d8" response="^2"/>
  </announcement>
  <announcement comment="CFBVM has been deactivated, CFB will be triggered" media-id="223">
          <play-condition id="d9" response="^2"/>
  </announcement>
  <!-- Something went wrong. -->
  <announcement comment="System Error" media-id="1800"/>
</announcements>
```

Example 27 Announcement Table of CFB/CFBVM Deactivation

4.3.3.4 Constructing Source Document Variants

An example for the CDIV rule set provisioning is shown in Example 26. It is recommended to create service data variants, which represent the use cases. The few instances of the `<source>` document are constructed manually, which represent combinations of the user input and the service data variants.

In reality, the `<source>` document is generated by the GenSSC engine. The content of the extracted syntax is added to the `<command-data>` element. The pieces of the service data referred by the items of the `mtasGenSscCmdUtPath` attribute are copied from the simservs.xml document to the source XML. The rule set of the CDIV service is copied to the `<source>` document according to the `mtasGenSscCmdUtPath` configuration in Section 4.3.2 Configuration Overview on page 54.

An instance of the possible `<source>` document variations is shown in Example 28. This instance represents use case 6. The comments are not generated by the GenSSC engine, they serve only for better understanding.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This document is passed to the XSL Transform function as
     the source XML of the transformation.
     The <service-data> element represents a provisioning
     of the use case 6. -->
<source version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <!-- The <user-id> comes from the P-Asserted-Identity header
       of the SSC-INVITE. -->
  <user-id>sip:abc@ericsson.com</user-id>
  <!-- The <command-data> contains the elements of the extracted
       command syntax.
       It includes the schema type and the service code in this case. -->
  <command-data version="1">
    <schema>composite</schema>
  </command-data>
  <!-- The <service-data> contains a piece of the user's simservs
document. The mtasGenSscCmdUtPath determines which elements
shall be taken over from the user's service data.
It includes the <cp:ruleset> of the CDIV service in this case. -->
  <service-data index="0" valid="true" version="1" ⇒
path="simservs/communication-diversion/cp:ruleset
    ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)">
    <cp:ruleset xmlns:cp="urn:ietf:params:xml:ns:common-policy"
      xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap">
<!-- The "cdiv-anonymous" rule is a combined 'busy' and 'anonymous'
rule. Although it is the top-most busy rule it is not a simple
rule that is why it is out of scope and is filtered out. -->
      <cp:rule id="cdiv-anonymous">
        <cp:conditions>
          <ss:anonymous/>
          <ss:busy/>
        </cp:conditions>
        <cp:actions>
          <ss:forward-to>
            <ss:target>sip:blunt-message@example.com</ss:target>
          </ss:forward-to>
        </cp:actions>
      </cp:rule>
 <!-- The "cdiv-busy-vm" rule is the top-most simple busy rule.
 It is an active rule having a forward-to target
 to the voicemail. It will be deactivated. -->
      <cp:rule id="cdiv-busy-vm">
        <cp:conditions>
          <ss:busy/>
        </cp:conditions>
        <cp:actions>
          <ss:forward-to>
            <ss:target>voicemail:internal</ss:target>
            <ss:reveal-identity-to-caller>true</ss:reveal-identity-to-caller>
          </ss:forward-to>
        </cp:actions>
      </cp:rule>
      <!-- The "cdiv-busy" rule is a deactivated simple busy rule.
           Its state will be reported to the user. -->
      <cp:rule id="cdiv-busy">
        <cp:conditions>
          <ss:busy/>
          <ss:rule-deactivated/>
        </cp:conditions>
        <cp:actions>
          <ss:forward-to>
            <ss:target>sip:secretary@example.com</ss:target>
            <ss:reveal-identity-to-caller>
              false
            </ss:reveal-identity-to-caller>
          </ss:forward-to>
        </cp:actions>
      </cp:rule>
    </cp:ruleset>
  </service-data>
</source>
```

Example 28 Generated <source> Document

4.3.3.5 Layout Design

There are many cases where the creation of a layout is a kind of programming. It means more effort than the configuration of command syntaxes in the system-defined MOs but it provides much more flexibility than the system-defined SSC commands.

Similarly to the other programming tasks, it must be clarified at the beginning what the layout produces and what internal data is necessary. It is the phase in which the layout designer collects the information needed for producing the required output and defines a set of `<xsl:variable>` elements. The contents and the format of the internal variables are concocted in this step.

The following internal data is defined in this example:

- Rule counters containing the number of the provisioned CFB/CFBVM rules. All the active and deactivated rules are counted (`cfb_rule_cnt`, `cfbvm_rule_cnt`)
- Name of the top-most simple busy rule (`top_rule_name`)
- Names of the CFB/CFBVM rules (`cfb_rule_name, cfbvm_rule_name`)
- List of the active busy rules (`active_rules`) in the following format:
  - `;` – No any active rule
  - `|rule_name|;` – One active rule
  - `|rule1_name||rule2_name|;` – Two active rules

  Note: The `||` substring indicates that there are two active rules.
- Use case indicator containing the d0–d9 strings (`use_case`)

4.3.3.6 Layout Implementation

This section includes a concrete layout implementation of the user story, see Section 4.3.1 Subject of Example on page 53. The top part of the layout includes the definition of the `<xsl:variable>` elements, see Example 29.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This layout document includes an XSLT stylesheet which
deactivates the top-most CFB/CFBVM rule. -->
<layout version="1" name="CFB/CFBVM deactivation" ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
 <xsl:stylesheet version="1.0" ⇒
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" ⇒
xmlns:gs="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc" ⇒
xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap" ⇒
xmlns:cp="urn:ietf:params:xml:ns:common-policy" ⇒
exclude-result-prefixes="gs">
    <xsl:output method="xml" indent="yes"/>
    <xsl:template match="/">
      <!-- *** VARIBALE DEFINITIONS *** -->

  <!-- $cfb_rule_cnt includes the number of the simple CFB rules. -->
  <xsl:variable name="cfb_rule_cnt">
```

```
  <xsl:value-of select="count(/gs:source/gs:service-data[@index='0'] ⇒
/cp:ruleset/cp:rule[cp:conditions/ss:busy ⇒
and ((count(cp:conditions/*)=1) ⇒
or ((count(cp:conditions/*)=2) ⇒
and (cp:conditions/ss:rule-deactivated))) ⇒
and not(contains(cp:actions/ss:forward-to/ss:target, 'voicemail'))])"/>
  </xsl:variable>

    <!-- $cfbvm_rule_cnt includes the number of the simple CFBVM rules. -->
    <xsl:variable name="cfbvm_rule_cnt">
      <xsl:value-of select="count(/gs:source/gs:service-data[@index='0'] ⇒
/cp:ruleset/cp:rule[cp:conditions/ss:busy ⇒
and ((count(cp:conditions/*)=1) ⇒
or ((count(cp:conditions/*)=2) ⇒
and (cp:conditions/ss:rule-deactivated))) ⇒
and contains(cp:actions/ss:forward-to/ss:target, 'voicemail')])"/>
    </xsl:variable>

<!-- $top_rule_name includes the name of the top-most simple
         CFB/CFBVM rule or an empty string.
         Note: If the source document does not include an active
         CFB/CFBVM rule then the $top_rule_name will be
         an empty string. It doesn't cause a runtime error. -->
  <xsl:variable name="top_rule_name">
   <xsl:value-of select="string(/gs:source/gs:service-data[@index='0'] ⇒
/cp:ruleset/cp:rule[cp:conditions/ss:busy ⇒
and ((count(cp:conditions/*)=1) ⇒
or ((count(cp:conditions/*)=2) ⇒
and (cp:conditions/ss:rule-deactivated)))]/@id)"/>
   </xsl:variable>

  <!-- $cfb_rule_name includes the name of the CFB rule or
         an empty string. -->
  <xsl:variable name="cfb_rule_name">
    <xsl:value-of select="string(/gs:source/gs:service-data[@index='0'] ⇒
/cp:ruleset/cp:rule[cp:conditions/ss:busy ⇒
and ((count(cp:conditions/*)=1) ⇒
or ((count(cp:conditions/*)=2) ⇒
and (cp:conditions/ss:rule-deactivated))) ⇒
and not(contains(cp:actions/ss:forward-to/ss:target, 'voicemail'))]/@id)"/>
    </xsl:variable>

    <!-- $cfbvm_rule_name includes the name of the CFBVM rule or
         an empty string. -->
    <xsl:variable name="cfbvm_rule_name">
      <xsl:value-of select="string(/gs:source/gs:service-data[@index='0'] ⇒
/cp:ruleset/cp:rule[cp:conditions/ss:busy ⇒
and ((count(cp:conditions/*)=1) ⇒
or ((count(cp:conditions/*)=2) ⇒
and (cp:conditions/ss:rule-deactivated))) ⇒
and contains(cp:actions/ss:forward-to/ss:target, 'voicemail')]/@id)"/>
    </xsl:variable>

    <!-- $active_rules collects the name of the active
         CFB and CFBVM rules. -->
    <xsl:variable name="active_rules">
      <xsl:for-each select="/gs:source/gs:service-data[@index='0'] ⇒
/cp:ruleset/cp:rule[cp:conditions/ss:busy ⇒
and count(cp:conditions/*)=1 ]">
        <xsl:value-of select="concat('|', @id, '|')"/>
      </xsl:for-each>
      <xsl:value-of select="';'"/>
    </xsl:variable>

<!-- $use_case tells the ID of the use case.
       It is included in the @id attribute of the report or task.
        - d0-d9: use cases described in the user guide
        - system_error: something went wrong -->
    <xsl:variable name="use_case">
      <xsl:choose>
    <!-- WARNING: the '<' and '>' signs in the attribute values need
     to be escaped since the XSL stlyesheet must be
     a well-formed XML document! -->
      <xsl:when test="($cfb_rule_cnt &gt; 1) or ($cfbvm_rule_cnt &gt; 1)">
        <xsl:value-of select="'d0'"/>
        </xsl:when>
```

```
        <xsl:when test="$active_rules = ';'">
         <xsl:value-of select="'d1'"/>
        </xsl:when>
      <xsl:when test="($cfb_rule_cnt + $cfbvm_rule_cnt = 1) ⇒
and ($active_rules = concat('|', $cfb_rule_name,'|;'))">
        <xsl:value-of select="'d2'"/>
        </xsl:when>
        <xsl:when test="($cfb_rule_cnt + $cfbvm_rule_cnt = 1) ⇒
and ($active_rules = concat('|', $cfbvm_rule_name,'|;'))">
        <xsl:value-of select="'d3'"/>
        </xsl:when>
        <xsl:when test="($cfb_rule_cnt + $cfbvm_rule_cnt = 2) ⇒
and ($active_rules = concat('|', $cfb_rule_name,'|;')) ⇒
and ($top_rule_name = $cfb_rule_name)">
          <xsl:value-of select="'d4'"/>
        </xsl:when>
        <xsl:when test="($cfb_rule_cnt + $cfbvm_rule_cnt = 2) ⇒
and ($active_rules = concat('|', $cfb_rule_name,'|;')) ⇒
and ($top_rule_name = $cfbvm_rule_name)">
          <xsl:value-of select="'d5'"/>
        </xsl:when>
        <xsl:when test="($cfb_rule_cnt + $cfbvm_rule_cnt = 2) ⇒
and ($active_rules = concat('|', $cfbvm_rule_name,'|;')) ⇒
and ($top_rule_name = $cfbvm_rule_name)">
          <xsl:value-of select="'d6'"/>
        </xsl:when>
        <xsl:when test="($cfb_rule_cnt + $cfbvm_rule_cnt = 2) ⇒
and ($active_rules = concat('|', $cfbvm_rule_name,'|;')) ⇒
and ($top_rule_name = $cfb_rule_name)">
          <xsl:value-of select="'d7'"/>
        </xsl:when>
        <xsl:when test="($cfb_rule_cnt + $cfbvm_rule_cnt = 2) ⇒
and contains($active_rules, '||') ⇒
and ($top_rule_name = $cfb_rule_name)">
          <xsl:value-of select="'d8'"/>
        </xsl:when>

<xsl:when test="($cfb_rule_cnt + $cfbvm_rule_cnt = 2) ⇒
and contains($active_rules, '||')
and ($top_rule_name = $cfbvm_rule_name)">
          <xsl:value-of select="'d9'"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="'system_error'"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
```

Example 29 Variable Definitions

The lower part of the layout document comprises the generation of the output. It is divided into several sections. They reflect to the blocks of Table 5 extended with a default error handling part.

The first section refers to use case 0 in Table 5, see Example 30.

```
<!-- *** OUTPUT GENERATION *** -->

      <xsl:choose>

        <!-- *** PROVISIONING ERROR *** -->

      <!-- Error use case (d0, system_error):
           a negative report shall be generated. -->
      <xsl:when test="contains('d0,system_error', $use_case)">
        <report version="1" ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
          <xsl:attribute name="id">
            <xsl:value-of select="$use_case"/>
          </xsl:attribute>
          <response>500 Internal Error</response>
          <job>put</job>
          <value type="empty"/>
        </report>
      </xsl:when>
```

Example 30 Treatment of Provisioning Defects – Negative Report Generated

The use case 1, 5, and 7 dummy tasks are shown in Example 31.

```
  <!-- *** DUMMY USE CASES *** -->

      <!-- Dummy use cases (d1, d5 and d7):
           a positive report shall be generated. -->
      <xsl:when test="contains('d1,d5,d7', $use_case)">
        <report version="1" ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
          <xsl:attribute name="id">
            <xsl:value-of select="$use_case"/>
          </xsl:attribute>
          <response>205 Reset Content</response>
          <job>put</job>
          <value type="simple">false</value>
        </report>
      </xsl:when>
```

Example 31 Dummy Task Use Cases – Positive Report Generated

The ‘‘service data update’’ session creates a `<task>` document to modify the service data according to the user input. Once the `<task>` has been generated, for further processing, see Section 4.1.3 Workflow Overview on page 36.

An appropriate XCAP request is sent to the XCAP servlet, which applies the service-specific application rules on the request and updates the HSS with the modified service data. The GenSSC engine creates a `<report>` from the XCAP response (the report ID is inherited from the task ID). The report is compared with the announcement table and the matching voice message is sent to the user. The tasks of the service data update are shown in Example 66.

```
<!-- *** SERVICE DATA UPDATE *** -->

 <!-- Use cases of service data update (d2, d3, d4, d6, d8 and d9):
 a task shall be generated. -->
 <xsl:when test="contains('d2,d3,d4,d6,d8,d9', $use_case)">
   <task version="1" ⇒
xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap" ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
     <xsl:attribute name="id">
       <xsl:value-of select="$use_case"/>
         </xsl:attribute>
         <head>
           <action>set</action>
           <content-type>application/xcap-el+xml</content-type>
           <xsl:element name="ut-path">simservs/⇒
communication-diversion/cp:ruleset/cp:rule[@id=⇒
"<xsl:value-of select="$top_rule_name"/>"]⇒
/cp:conditions/rule-deactivated
              ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
           </xsl:element>
         </head>
         <body>
           <ss:rule-deactivated/>
         </body>
        </task>
      </xsl:when>
```

Example 32 Service Data Update – Task Generated

The output generation and the layout end with a final error check, as shown in Example 33.

```
<!-- *** DEFAULT CASE *** -->

      <xsl:otherwise>
 <!-- If the processing gets to here then something went wrong above!
  Anyway, the service data must not be touched.The call is rejected
with a "System Error" response. -->
   <report version="1" id="stylesheet_error" ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
          <response>500 Internal Error</response>
          <job>put</job>
          <value type="empty"/>
        </report>
      </xsl:otherwise>
      </xsl:choose>

    </xsl:template>
  </xsl:stylesheet>
</layout>
```

Example 33 End of Layout – Final Error Check

Studying the source document again, it shows the current provisioning and the user input results in the `task[id=d6]`; the top-most CFBVM rule is deactivated. The generated task is shown in Example 34.

```
<?xml version="1.0"?>
<task version="1" id="d6" ⇒
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc" ⇒
xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap">
  <head>
    <action>set</action>
    <content-type>application/xcap-el+xml</content-type>
    <ut-path>
      simservs/communication-diversion/cp:ruleset⇒
/cp:rule[@id="cdiv-busy-vm"]/cp:conditions/ss:rule-deactivated
    </ut-path>
  </head>
  <body>
    <ss:rule-deactivated/>
  </body>
</task>
```

Example 34 Result of XSLT Transformation – task[id=d6]

4.4 Composite Configuration for MSN Support

This section contains examples for a composite configuration of GenSSC for basic SSC operations on Multi-Subscriber Number (MSN) enhanced Communication Diversion (CDIV) rule set.

MSN enhanced service rule is a rule containing condition on the served identity of the subscriber, that is, main or alias Public User Identity (PUI) being used in signaling for a given MMT call. If the served identity condition is the only condition in a rule or it is combined with one of the following conditions: no reply, not reachable, not logged in or busy, then such a rule is considered a simple rule.

```
<?xml version="1.0" encoding="UTF-8"?>
<ss:simservs
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">
  <ss:communication-diversion active="true">
    <cp:ruleset>
      <cp:rule id="simple-cfu-alias-2-to-alice">
        <cp:conditions>
          <mmt-serv:served-identity>
            <mmt-serv:one id="sip:+46111111112@operator.com"/>
          </mmt-serv:served-identity>
        </cp:conditions>
        <cp:actions>
          <ss:forward-to>
            <ss:target>sip:alice@example.com</ss:target>
          </ss:forward-to>
        </cp:actions>
      </cp:rule>
      <cp:rule id="simple-cfnr-alias-3-to-bob">
        <cp:conditions>
          <ss:no-answer/>
          <mmt-serv:served-identity>
            <mmt-serv:one id="sip:+46111111113@operator.com"/>
          </mmt-serv:served-identity>
        </cp:conditions>
        <cp:actions>
          <ss:forward-to>
            <ss:target>sip:bob@example.com</ss:target>
          </ss:forward-to>
        </cp:actions>
      </cp:rule>
    </cp:ruleset>
  </ss:communication-diversion>
</ss:simservs>
```

Example 35 Simservs.xml Document Including the MSN Enhanced CDIV Simple Rules

The following conditions must be satisfied to enable subscribers with operating on MSN enhanced CDIV rules using SSC:

1. CDIV service has been activated at the node and subscriber levels.
2. Flexible Identity Presentation (FIP) service has been activated at the node and subscriber levels.
3. Multi-Subscriber Number License is present and valid.
4. Ut Interface License is present and valid.
5. Served-identity condition has been activated by an operator for a given subscriber, see Example 36.

6. Mapping of integer alias IDs to subscriber's MSN alias PUIs has been defined in the FIP service data, see Example 37.

7. The subscriber has been notified by the operator of the FIP mapping, so that the subscriber knows which integer ID to use in an SSC command to refer to a particular MSN alias PUI. The subscriber must also be notified of every change done to the FIP mapping by the operator.

8. GenSSC has been configured to support required operations.

9. All announcements referenced in the GenSSC configuration have been provided by the operator.

```
<mmt-op:operator-communication-diversion activated="true">
  <mmt-op:conditions>
    <mmt-op:served-identity activated="true"/>
  </mmt-op:conditions>
</mmt-op:operator-communication-diversion>
```

Example 36 Served-identity Condition Activated by Operator in the CDIV Service Data

```
<mmt-serv:flexible-identity-presentation active="true">
  <mmt-serv:fip-identity>sip:something@operator.com</mmt-serv:fip-identity>
  <mmt-serv:msn-fip-identity id="1">
    <mmt-serv:identity>sip:+46111111111@operator.com</mmt-serv:identity>
  </mmt-serv:msn-fip-identity>
  <mmt-serv:msn-fip-identity id="2">
    <mmt-serv:identity>sip:+46111111112@operator.com</mmt-serv:identity>
  </mmt-serv:msn-fip-identity>
  <mmt-serv:msn-fip-identity id="3">
    <mmt-serv:identity>sip:+46111111113@operator.com</mmt-serv:identity>
  </mmt-serv:msn-fip-identity>
</mmt-serv:flexible-identity-presentation>
```

Example 37 Mapping of Integer Alias IDs to the Subscriber's MSN Alias PUIs in the FIP Service Data

The example GenSSC configuration support for the following SSC operations: Set (create or update a rule), Get (interrogate or verify a rule) and Delete, for the following kinds of MSN enhanced CDIV rules: Call Forwarding Unconditional (CFU), Call Forwarding on No Reply (CFNR), Call Forwarding on Not Reachable (CFNRc), Call Forwarding on Not Logged In (CFNL), and Call Forwarding on Busy (CFB).

The configuration can be organized into the following MO groups, see Example 38.

```
+MtasGenSsc
+-MtasGenSscGroup["MsnCDiv"]
+--mtasGenSscCmd["MsnCfuSet"] //creation/update of a CFU rule
+--mtasGenSscCmd["MsnCfuDel"] //deletion of a CFU rule
+--mtasGenSscCmd["MsnCfuGet"] //interrogation/verification of a CFU rule
+--mtasGenSscCmd["MsnCDivCondSet"] //creation/update of a CFNR, CFNRc, CFNL or CFB rule
+--mtasGenSscCmd["MsnCDivCondDel"] //deletion of a CFNR, CFNRc, CFNL or CFB rule
+--mtasGenSscCmd["MsnCDivCondGet"] //interrogation/verification of a CFNR, CFNRc, CFNL or CFB rule
```

Example 38 Grouping GenSSC Configurations

A regular CFU rule by name (unconditional) does not contain any condition, in this document it is assumed that an MSN enhanced CFU rule contains one condition – on the served user.

Example GenSSC configuration described in Section 4.4 Composite Configuration for MSN Support on page 66 assumes the following SSC command syntax:

- Set (update): `*11*<aliases>#*<service_code>*<target>#`

  SSC command example for setting a CFU target to aliases 1, 3:

  `*11*13#*21*0046111111111#`

- Delete: `*11*<aliases>##<service_code>#`

  SSC command example for deletion of MSN CFB rules of aliases 2, 3:

  `*11*23##67#`

- Interrogate: `*11*<alias>#*#<service_code>#`

  SSC command example for interrogation of MSN CFNR rule of alias 2:

  `*11*2#*#61#`

- Verify: `*11*<alias>#*#<service_code>*<target>#`

  SSC command example for verifying the CFNL target of alias 1:

  `*11*1#*#63*0046333333333#`

where:

- `alias, aliases`: One or a list of one-digit alias IDs, which are mapped to proper alias PUIs in the FIP service data, see Example 37. For the interrogation and verification operations only one alias ID is allowed in the SSC command, for setting and deletion it is possible to use multiple alias IDs.

- `code`: A code denoting certain kinds of CDIV, for example, 21 for CFU, 61 for CFNR, 62 for CFNRc, 63 for CFNL, 67 for CFB.

- `target`: A telephone number to which calls falling into a given rule must be diverted, for example, 0046222222222.

The command syntax chosen in the document allows for using of three aliases at most, owing to the limitations of the digit tree containing the leading part of the SSC commands. Using more aliases is possible with a different SSC syntax. Examples are listed below.

4.4.1 Support for MSN CFU Set Command

This section contains an example for a composite configuration which analyzes the service data in the first step. According to the actual content, it either updates the service data or keeps it unchanged and plays an appropriate announcement to the user at the end. The example sets a new or updates an existing simple MSN enhanced CFU (call forwarding unconditional) rule. These rules are members of the CDIV rule set, see Example 35.

4.4.1.1 Subject of the Example

The following user story defines the subject of the example in this section:

An IMS user wants to set or update one or more simple MSN enhanced CFU rules using the SSC command format: `*11*<aliases>#*21*<target>#` so that all incoming calls to the first, second, and third (123) alias PUIs are unconditionally (21) diverted to the entered target.

Example for a related SSC command: `*11*23#*21*0046222222222#`

The user wants to be informed:

- Which new rule has been created
- Which existing rule has been updated
- If such a rule exists
- If alias IDs in the SSC command are not mapped to any PUIs in FIP or the service data is broken
- If an error occurred

4.4.1.2 Configuration Overview

This section shows an example configuration of an `MtasGenSscCmd` MO instance which is in line with the example in Section 4.4.1.1 Subject of the Example on page 70.

`MtasGenSscCmd`: MsnCfuSet (key)

The “MsnCfuSet” string identifies this `MtasGenSscCmd` MO instance.

```
mtasGenSscCmdHeader:
*11*1#*21|*11*2#*21|*11*3#*21|*11*12#*21|*11*13#*21|*11*23#*21|⇒
*11*123#*21
```

The command header attribute lists the variations of leading digits related to this configuration. They differ with the alias IDs in the middle.

The command syntax chosen in this example allows for using of three aliases at most, owing to the decoding tree limitations. Using more aliases would be possible with a different SSC syntax, for example, `*1121*<aliases>#*<target>#`. In that case, the command header attribute would have the value of `*1121*`.

```
mtasGenSscCmdSyntax:
/^\*11\*([1-3]{1,3})#\*21([0-9]+)#$/schema=composite;⇒
digits aliases=\1; item target=tel:\2/
```

The Command Pattern `^\*11\*([1-3]{1,3})#\*21([0-9]+)#$` declares the format of the Request URI.

The Command Syntax part `schema=composite; digits aliases=\1; item target=tel:\2` informs the GenSSC engine that a composite configuration is processed. It means that the task or report is extracted from the XSLT style sheet included in the layout.xml document of the `mtasGenSscCmdLayout` attribute. It also declares two variables to be fetched from an SSC command: aliases of digits type to be taken from the first variable part of the command and `target` of `item` type to be taken from the second variable part.

```
mtasGenSscCmdUtPath:
simservs/communication-diversion~~simservs/mmt-serv:⇒
flexible-identity-presentation
```

The UtPath points to the CDIV and FIP services, they are copied from the *<service-data>* to the *<source>* document.

`mtasGenSscCmdAnnouncements`:

The announcement table is described in Section 4.4.1.3.3 Announcement Configuration on page 75.

`mtasGenSscCmdLayout`:

The layout document is described in Section 4.4.1.3.6 Layout Implementation on page 78.

Other attributes:

The attributes `mtasGenSscCmdReplace`, `mtasGenSscCmdContentType`, and `mtasGenSscCmdUriInChargingMessages` of the `MtasGenSscCmd` MO are not used in this configuration, the default values are assigned to them.

4.4.1.3 Workflow Overview

This section describes the workflow overview.

4.4.1.3.1 Understanding the Details

Analyze the service data pieces related to the MSN enhanced CFU rules to understand the technical details and to discover the optional difficulties.

A CDIV rule is type of ‘‘call forwarding unconditional’’ if it does not contain any condition. However, an MSN-enhanced CFU rule includes the `<mmt-serv:served-identity>` condition containing one or more `<mmt-serv:one>` elements, with one subscriber’s alias PUI each.

A rule can be deactivated by inserting the `<ss:rule-deactivated/>` element into the conditions of the rule or can be deleted by removing the rule from the rule set. These different technical solutions result in the same user experience, that is, the rule is not active. A rule is active if it exists and is not deactivated. Deactivated rules are not analyzed or updated by the GenSSC configured according to this example. Only simple CDIV MSN CFU rules, that is, the ones rules containing exactly one ‘‘served-identity’’ condition are processed.

The ‘‘served-identity’’ condition can be combined with other conditions (like validity and identity). The example focuses on the simple MSN enhanced CFU rules, the combined rules are out of the scope.

For example, configuration against simple MSN enhanced Conditional CDIV conditional rules, where the ‘‘served-identity’’ condition is combined with one of the following conditions: no reply, not reachable, not logged in or busy, see Section 4.4.2 Support for MSN CFU Delete Command on page 85.

A CDIV rule with the same name as the name composed by the GenSSC according to the configured template may already exist in the subscribers rule set. This case can be considered a provisioning error and the user is notified about it.

4.4.1.3.2 Clarification of the Use Cases

The user’s service profile includes the following information:

- If all alias IDs in the SSC command are mapped to subscriber’s alias PUIs in the FIP service data
- If any simple active MSN enhanced CFU rules for alias IDs given in the SSC command already exist in the CDIV rule set
- If any simple active MSN enhanced CFU rules for the target given in the SSC command already exist in the CDIV rule set
- If any simple active MSN enhanced CFU rules for all alias IDs and the target given in the SSC command already exist in the CDIV rule set

The combinations of the information in the Section 4.4.1.3.2 Clarification of the Use Cases on page 72 define the use cases of the user story. They are the input parameters which help to determine:

- The new rule to be created or new entries to be added to an existing rule
- The aliases from the existing rules and whole rules to be deleted
- The announcement to be played to the user

The service data must be modified with the condition that the user be provided with a meaningful announcement.

Table 6 summarizes the use cases to be considered and the announcements to be played for each use case.

Table 6 Use Cases of SSC Command for Setting or Updating an MSN Enhanced CDIV CFU Rule

Use Case	All Alias IDs Mapped to Alias PUIs in FIP	Simple Rules for Given Aliases Exist	Simple Rules for a Given Target Exist	A Simple Rule for Both Given Aliases and a Given Target Exists	To Be Created or Updated	To Be Deleted	Announcement
UC-0	Yes	No	No	No	A new rule is created	Nothing	A new MSN CFU rule has been created with the requested identities and the entered target number.
UC-1	Yes	No	No	No	A new rule is created	Aliases or whole existing rule is removed	A new MSN CFU rule has been created with the requested identities and the entered target number.
UC-2	Yes	No	Yes	No	New entries are added to the existing rule	Nothing	The requested identities have been added to your MSN CFU rule of the entered target number.

Use Case	All Alias IDs Mapped to Alias PUIs in FIP	Simple Rules for Given Aliases Exist	Simple Rules for a Given Target Exist	A Simple Rule for Both Given Aliases and a Given Target Exists	To Be Created or Updated	To Be Deleted	Announcement
UC-3	Yes	No	No	No	New entries are added to the existing rule	Aliases or whole existing rule is removed	The requested identities have been added to your MSN CFU rule of the entered target number.
UC-4	Yes	No	Yes	Yes	Nothing	Nothing	The referred identities are already included in your MSN CFU rule of the entered target number.
UC-5	No	Irrelevant	Irrelevant	Irrelevant	Nothing	Nothing	The MSN CFU rules in your service data are corrupt or the alias ID is incorrect.

A summary of the announcement variations is shown in Table 7.

Table 7 Summary of Announcement Variations - Setting or Updating an MSN Enhanced CDIV CFU Rule

Announcement	Media ID	Use Cases
A new MSN CFU rule has been created with the requested identities and the entered target number.	1321	0, 1
The requested identities have been added to your MSN CFU rule of the entered target number.	1311	2, 3
The referred identities are already included in your MSN CFU rule of the entered target number.	251	4
The MSN CFU rules in your service data are corrupt or the alias ID is incorrect.	800	5

Table 7 is divided into three parts, as follows:

- Use cases “0, 1, 2, and 3” represents real tasks. The service data is modified according to the use case.
- Use case “4” represents a dummy task. There is no need to change service data since the current value and the new value are the same. It is enough to create a proper report.
- Use case “5” represents a provisioning error cases. The service data is invalid and must not be modified. An appropriate report is created which determines the announcement to be played.

4.4.1.3.3 Announcement Configuration

The layout is constructed in a way which:

- Performs the required modifications on the service data
- Provides the necessary information for the announcement

The use cases and the corresponding announcements must be determined before the layout implementation. The layout design often starts with the creation of the announcement table.

In Example 39, the “id” attributes of the tasks and reports are given descriptive values, which enhance readability and simplify the configuration of the announcement table.

```
<?xml version="1.0" encoding="utf-8"?>
<announcements xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
<announcement comment="The MSN CFU rules in your service data are corrupt or the alias id is
incorrect." media-id="800">
<play-condition id="provisioning_error"/>
</announcement>   <announcement comment="The referred identities are already included in your ⇒
MSN CFU rule of the entered target number." media-id="251">
<play-condition id="already_activated"/>
</announcement>
<announcement comment="Internal error occurred" media-id="1800">
<play-condition list-length="0"/>
</announcement>
<announcement comment="The requested identities have been added to your MSN CFU rule of the
entered target number." media-id="1311">
<play-condition id="extend"/>
</announcement>   <announcement comment="A new MSN CFU rule has been created with the requested
identities and the entered target number." media-id="1321">
    <play-condition id="create"/>
  </announcement>
 <announcement comment="Internal error occurred" media-id="1800"/>
</announcements>
```

Example 39 Announcement Table for Setting or Updating an MSN Enhanced CFU Rule

4.4.1.3.4 Constructing Source Document Variants

A sample CDIV rule set provisioning is shown in Example 35. It is recommended to create service data variants which represent the use cases. The A few

instances of the <source> document are constructed manually which represent combinations of the user input and the service data variants. In reality, the <source> document is generated by the GenSSC engine. The content of the Extracted Syntax is added to the *<command-data>* element and the pieces of the service data referred by the items of the `mtasGenSscCmdUtPath` attribute are copied from the simservs.xml document to the source XML. The contents of the CDIV and FIP services are copied to the <source> document according to the `mtasGenSscCmdUtPath` configuration in Section 4.4.1.2 Configuration Overview on page 70.

An instance of the possible <source> document variants is shown in Example 40. The instance represents the use case "3". The comments in the Example 40 are not generated by the GenSSC engine, they serve only for better understanding.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This document is passed to the XSL Transform function as the source XML of the transformation.
     The <service-data> element represents a provisioning of the use case 3. -->
<source version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">

  <!-- The <user-id> comes from the P-Asserted-Identity header of the SSC-INVITE. -->
  <user-id>sip:abc@ericsson.com</user-id>

  <!-- The <command-data> contains the elements of the extracted command syntax.
       It includes the schema type in this case. -->
  <command-data version="1">
    <schema>composite</schema>
  </command-data>

  <!-- The <service-data> contains a piece of the user's simservs document. The mtasGenSscCmdUtPath determines
       which elements shall be taken over from the user's service data.
       There are two <service-data> elements in this case, including the CDIV and FIP services. -->
  <service-data index="0" valid="true" version="1"
      path="simservs/communication-diversion">
    <ss:communication-diversion active="true">
      <cp:ruleset>
        <!-- The "msn-video-to-alice" rule is a combined 'served-identity' and 'media' rule. It is not a
             simple rule that is why it is out of scope and is filtered out. -->
        <cp:rule id="msn-video-to-alice">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111112@operator.com"/>
            </mmt-serv:served-identity>
            <ss:media>video</ss:media>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:alice@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfu-to-0046222222222" rule is an active simple MSN enhanced CFU rule. It has
the same forward-to target as requested in the SSC command of the user story but different alias
(1,that is the main number of the subscriber). Aliases 2 and 3 will be added to the served-identity
             condition of this rule. -->
        <cp:rule id="simple-msn-cfu-to-0046222222222">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111111@operator.com"/>
            </mmt-serv:served-identity>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>tel:0046222222222</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
```

```
        <!-- The "simple-msn-cfu-to-bob" rule is an active simple MSN enhanced CFU rule. It has the
             same alias as one of the aliases requested in the SSC command of the user story but different
             forward-to target. Because there are no other aliases in the served-identity condition of the
             rule,the whole rule will be deleted. -->
        <cp:rule id="simple-msn-cfu-to-bob">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111113@operator.com"/>
            </mmt-serv:served-identity>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:bob@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
       </cp:ruleset>
    </ss:communication-diversion>
  </service-data>

  <service-data index="1" valid="true" version="1"
      path="simservs/mmt-serv:flexible-identity-presentation">
    <mmt-serv:flexible-identity-presentation active="true">
      <!-- The fip-identity element is not analyzed by the Generic SSC according to this example ⇒
configuration.
           It can contain either a router address or a main number of the subscriber. In the latter case, the
           main number should also be configured in the msn-fip-identity with id="1". -->
      <mmt-serv:fip-identity>sip:something@operator.com</mmt-serv:fip-identity>
      <!-- Alias ID "1" in an SSC command will be replaced with alias PUI sip:+46111111111@operator.com in
           operations on the CDIV service data. This is the subscriber's main (first alias) number . -->
      <mmt-serv:msn-fip-identity id="1">
        <mmt-serv:identity>sip:+46111111111@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "2" in an SSC command will be replaced with alias PUI sip:+46111111112@operator.com in
           operations on the CDIV service data. This is the subscriber's second alias number. -->
      <mmt-serv:msn-fip-identity id="2">
        <mmt-serv:identity>sip:+46111111112@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "3" in an SSC command will be replaced with alias PUI sip:+46111111113@operator.com in
           operations on the CDIV service data. This is the subscriber's third alias number. -->
      <mmt-serv:msn-fip-identity id="3">
        <mmt-serv:identity>sip:+46111111113@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
    </mmt-serv:flexible-identity-presentation>
  </service-data>
</source>
```

Example 40 Generated <source> Document

4.4.1.3.5 Layout Design

The following internal data is defined in this example:

- The alias IDs from the SSC command (aliases)
- The target from the SSC command including the ‘‘tel:’’ prefix (target)
- List of the identities (alias PUIs) addressed by the aliases (alias IDs) separated by the ‘‘|’’ symbol, that is, ‘‘identity-1|identity-2|’’. ‘‘|’’ means that there are no matching identities (identities).
- List of the IDs of the simple MSN-enhanced CFU rules which include a different target and contain PUIs stored in the ‘identities’ variable, separated by the ‘‘|’’ symbol, for example, ‘‘|id-1|id-2|’’. ‘‘|’’ means that there are no matching rules (impactedRules).

- List of the IDs of the simple MSN enhanced CFU rules which have to be updated. A subset of the affected rules to be kept which contain alias PUIs to be removed, separated by the ‘‘|’’ symbol, for example, ‘‘|id-1|id-2|’’. ‘‘|’’ means that there are no such rules (rulesToBeUpdated).

- List of the IDs of the simple MSN enhanced CFU rules which have to be deleted – a subset of the affected rules complementary to the set of rules to be updated – separated by the ‘‘|’’ symbol, for example, ‘‘|id-1|id-2|’’. ‘‘|’’ means that there are no such rules (rulesToBeDeleted).

- ID of the target rule, either an existing one or a new one created with the generated name: `msn_cfu_<target-without-tel-prefix>`. ‘‘|’’ means that the new rule cannot be created because a rule with the generated ID exists (targetRule).

- List of the identities in the served-identity condition of the target rule separated by the ‘‘|’’ symbol, for example, ‘‘|identity-1|identity-2|’’. ‘‘|’’ means that there are no identities yet, that is, this is a new rule (targetRuleIdentities).

- List of the missing identities in the target rule, that is, the identities addressed by the aliases in the SSC command, which are not yet included in the target rule's served-identity condition, separated by the ‘‘|’’ symbol, for example, ‘‘|identity-1|identity-2|’’. ‘‘|’’ means all the identities are already included in the target rule, that is, there is nothing to add (missingIdentities).

- What to do with the target rule: update an existing one, create a new one, nothing, or report an error (targetRuleAction)

4.4.1.3.6 Layout Implementation

This section includes a concrete layout implementation of the user story in Section 4.4.1.1 Subject of the Example on page 70.

The top part of the layout includes definitions of two helper templates, see Example 41.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This layout document contains an XSL style sheet which adds the referred served identities
     to the corresponding MSN CFU rule.
     It extends the cp:condition/mmt-serv:served-identity element if there is an existing rule having
     the MSN CFU type and target number, otherwise a new MSN CFU rule is created.
     The identities are removed from the other MSN CFU rules.
     Input parameters coming from the command data:
      - aliases: includes the MSN aliases referring to the identities to be added
      - target: forward-to target number including the 'tel:' prefix
     Service data elements used by the style sheet:
      - 0: /simservs/communication-diversion
      - 1: /simservs/flexible-identity-presentation
-->

<layout version="1"
    name="Update of the corresponding MSN CFU rule with the referred identities"
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:gs="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
      xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
      xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
      xmlns:cp="urn:ietf:params:xml:ns:common-policy"
      exclude-result-prefixes="gs">

    <xsl:output method="xml" indent="yes"/>

    <!-- Copy the mmt-serv:one elements of a given rule not including the referred identities. -->
    <xsl:template name="copyNotReferredOnes" match="/">
        <xsl:param name="ruleName"/>
        <xsl:param name="identities"/>
      <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion/cp:ruleset
          /cp:rule[@id=$ruleName]/cp:conditions/mmt-serv:served-identity
          /mmt-serv:one[not(contains($identities, concat('|', @id, '|')))]">
          <xsl:for-each select="@*">
            <xsl:element name="mmt-serv:one">
              <xsl:attribute name="id">
                <xsl:value-of select="."/>
              </xsl:attribute>
            </xsl:element>
          </xsl:for-each>
      </xsl:for-each>
    </xsl:template>

    <!-- Create mmt-serv:one elements including the referred identities. -->
    <xsl:template name="createReferredOnes" match="/">
        <xsl:param name="aliases"/>
      <!-- Create mmt-serv:one elements including the referred MSN identities. -->
      <xsl:for-each select="/gs:source/gs:service-data[@index='1']/mmt-serv:flexible-identity-presentation/
          mmt-serv:msn-fip-identity[contains($aliases, @id)]/mmt-serv:identity">
        <mmt-serv:one>
          <xsl:attribute name="id">
            <xsl:value-of select="text()"/>
          </xsl:attribute>
        </mmt-serv:one>
      </xsl:for-each>
    </xsl:template>

    <!-- Main template. -->
    <xsl:template match="/">
```

Example 41 Helper Templates

The next part of the layout includes the definitions of the *<xsl:variable>* elements, see Example 42.

```
<!-- ***************************************
           * Variable definitions                *
           *************************************** -->

      <!-- Store the alias IDs from the SSC command in a variable. -->
      <xsl:variable name="aliases">
        <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='aliases']"/>
```

```
</xsl:variable>

<!-- Store the target from the SSC command including the 'tel:' prefix in a variable. -->
<xsl:variable name="target">
  <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='target']"/>
</xsl:variable>

<!-- Collect the identities (alias PUIs) addressed by the aliases (alias IDs) in a string and separate
     them by colon '|' symbol.
     Example: |identity-1|identity-2|
     "|" means there are no matching identities -->
<xsl:variable name="identities">
  <xsl:for-each select="/gs:source/gs:service-data[@index='1']
      /mmt-serv:flexible-identity-presentation/mmt-serv:msn-fip-identity[contains($aliases, @id)]
      /mmt-serv:identity">
    <xsl:value-of select="concat('|', text())"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Collect the ids of the simple MSN enhanced CFU rules which include a different target and
     contain PUIs stored in the $identities. -->
<xsl:variable name="impactedRules">
  <xsl:for-each select="/gs:source/gs:service-data[@index='0']
      /ss:communication-diversion/cp:ruleset/cp:rule[
      count(cp:conditions/*)=1
      and (cp:actions/ss:forward-to/ss:target != $target)
      and count(cp:conditions/mmt-serv:served-identity
      /mmt-serv:one[contains($identities, concat('|', @id, '|'))]) != 0
      ]">
    <xsl:value-of select="concat('|', @id)"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Collect the ids of the simple MSN enhanced CFU rules which have to be updated: a subset
     of impacted rules to be kept which contain PUIs to be removed. -->
<xsl:variable name="rulesToBeUpdated">
  <xsl:for-each select="/gs:source/gs:service-data[@index='0']
      /ss:communication-diversion/cp:ruleset/cp:rule[contains($impactedRules, concat('|', @id, '|'))
      and count(cp:conditions/mmt-serv:served-identity
      /mmt-serv:one[not(contains($identities, concat('|', @id, '|')))]) != 0
      ]">
    <xsl:value-of select="concat('|', @id)"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Collect the ids of the simple MSN enhanced CFU rules which have to be deleted:
     a subset of the impacted rules complementary to the set of rules to be updated. -->
<xsl:variable name="rulesToBeDeleted">
  <xsl:for-each select="/gs:source/gs:service-data[@index='0']
      /ss:communication-diversion/cp:ruleset/cp:rule[
      contains($impactedRules, concat('|', @id, '|'))
      and not(contains($rulesToBeUpdated, concat('|', @id, '|')))
      ]">
    <xsl:value-of select="concat('|', @id)"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Find out the name (id) of the target rule. -->
<xsl:variable name="targetRule">
  <xsl:choose>
    <!-- Try to find an existing rule. -->
    <xsl:when test="count(/gs:source/gs:service-data[@index='0']
        /ss:communication-diversion/cp:ruleset/cp:rule[
        count(cp:conditions/*)=1
        and count(cp:conditions/mmt-serv:served-identity)=1
        and (cp:actions/ss:forward-to/ss:target = $target)
        ]) != 0">
      <xsl:value-of select="/gs:source/gs:service-data[@index='0']
          /ss:communication-diversion/cp:ruleset/cp:rule[
          count(cp:conditions/*)=1
          and count(cp:conditions/mmt-serv:served-identity)=1
          and (cp:actions/ss:forward-to/ss:target = $target)
```

```
              ]/@id"/>
          </xsl:when>
          <!-- New rule shall be created with the generated name: msn_cfu_<target-without-tel-prefix>.
               Check if there is an existing rule with the same name. -->
          <xsl:when test="count(/gs:source/gs:service-data[@index='0']
              /ss:communication-diversion/cp:ruleset/cp:rule[
              @id=concat('msn_cfu_', substring($target,5))
              ]) = 0">
            <xsl:value-of select="concat('msn_cfu_', substring($target,5))"/>
          </xsl:when>
          <!-- Indicate the error case: the new rule cannot be created since the generated rule name is
               reserved. -->
          <xsl:otherwise><xsl:value-of select="'|'"/></xsl:otherwise>
        </xsl:choose>
      </xsl:variable>

      <!-- Collect the identities in the target rule's served-identity condition in a string.
           "|" means there are no identities yet, that is this will be a new rule -->
      <xsl:variable name="targetRuleIdentities">
        <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion/cp:ruleset
            /cp:rule[@id=$targetRule]/cp:conditions/mmt-serv:served-identity/mmt-serv:one">
          <xsl:value-of select="concat('|', @id)"/>
        </xsl:for-each>
        <xsl:value-of select="'|'"/>
      </xsl:variable>

      <!-- Collect the 'missing' identities in the target rule, that is the identities addressed by the ⇒
 aliases in
           the SSC command, which are not yet included in the target rule's served-identity condition.
           "|" means all the identities are already included in in the target rule, that is there is ⇒
nothing to add
      -->
      <xsl:variable name="missingIdentities">
        <xsl:for-each select="/gs:source/gs:service-data[@index='1']
            /mmt-serv:flexible-identity-presentation/mmt-serv:msn-fip-identity[contains($aliases, @id)]
            /mmt-serv:identity">
          <xsl:if test="not(contains($targetRuleIdentities, concat('|', text(), '|')))">
            <xsl:value-of select="concat('|', text())"/>
          </xsl:if>
        </xsl:for-each>
        <xsl:value-of select="'|'"/>
      </xsl:variable>

      <!-- Find out what to do with the target rule:
           update:  update an existing rule
           create:  create a new rule
           nothing: there is nothing to do since all the referred
                    identities are already included in its condition
           error:   error report shall be generated -->
      <xsl:variable name="targetRuleAction">
        <xsl:choose>
          <!-- Check if the target rule is valid. -->
          <xsl:when test="$targetRule = '|'">
            <xsl:value-of select="'error'"/>
          </xsl:when>
          <!-- Check if the target rule exists.
               If not then a new rule shall be created. -->
          <xsl:when test="count(/gs:source/gs:service-data[@index='0']
              /ss:communication-diversion/cp:ruleset/cp:rule[
              @id=$targetRule]) = 0">
            <xsl:value-of select="'create'"/>
          </xsl:when>
          <!-- Check if there is any missing identity. -->
          <xsl:when test="$missingIdentities = '|'">
            <xsl:value-of select="'nothing'"/>
          </xsl:when>
          <!-- There is at least one missing identity: update. -->
          <xsl:otherwise>
            <xsl:value-of select="'update'"/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:variable>
```

Example 42 Variable Definitions

The bottom part of the layout document comprises the generation of the output. It is divided into several sections. They refer to the use cases of Table 6.

The first section refers to the use case ‘‘5’’ of Table 6, see Example 43.

```
<!-- ************************************
     * Output generation                *
     ************************************ -->

      <xsl:choose>
        <xsl:when test="($targetRuleAction='error') or ($identities='|')">
          <!-- The new rule cannot be created since there is an existing rule with
the same name or the alias IDs in the SSC command are not mapped to subscriber's
PUIs in the FIP service data. It is a provisioning error. -->
          <report version="1" id="provisioning_error"
              xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
            <response>500 Internal Error</response>
            <job>put</job>
            <value type="empty"/>
          </report>
        </xsl:when>
```

Example 43 Treatment of Provisioning Defects – Negative Report is Generated

Use case ‘‘4’’ which is a dummy task is reflected in Example 44.

```
<!-- There is nothing to do: calls to the referred identities are already redirected
to the target number and there are no other simple MSN enhanced CFU rules including
these identities in the served-identity condition. -->
        <xsl:when test="($targetRuleAction='nothing') and ($impactedRules='|')">
          <report version="1" id="already_activated"
              xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
            <response>200 OK</response>
            <job>put</job>
            <value type="empty"/>
          </report>
        </xsl:when>
```

Example 44 Dummy Task Use Case – Positive Report is Generated

The ‘service data update’ section creates *<task>* documents to modify the service data according to the user input. Once the <task> has been generated, the further processing is the same as described in Section 4.1.3 Workflow Overview on page 36. An appropriate XCAP request is sent to the XCAP servlet which applies the service-specific application rules on the request and updates the HSS with the modified service data. The GenSSC engine creates a <report> from the XCAP response (the report ID is inherited from the task ID). The report is compared with the announcement table and the matching voice message is sent to the user. The tasks of the service data update are shown in Example 45.

```
<!-- The service data needs to be modified. -->
        <xsl:otherwise>
          <task-list version="1"
              xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
              xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
              xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
              xmlns:cp="urn:ietf:params:xml:ns:common-policy">
            <!-- Remove the rules to be deleted. -->
            <xsl:if test="$rulesToBeDeleted != '|'">
              <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion ⇒
/cp:ruleset
                  /cp:rule[contains($rulesToBeDeleted, concat('|', @id, '|'))]">
                <task version="1">
                  <xsl:attribute name="id">
```

```
                    <xsl:value-of select="concat('delete-', @id)"/>
                  </xsl:attribute>
                  <head>
                    <action>delete</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='@id'/>"]
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                    </xsl:element>
                  </head>
                </task>
              </xsl:for-each>
            </xsl:if>
            <!-- Remove the referred identities from the rules to be updated. -->
            <xsl:if test="$rulesToBeUpdated != '|'">
              <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion ⇒
/cp:ruleset
                  /cp:rule[contains($rulesToBeUpdated, concat('|', @id, '|'))]">
                <task version="1">
                  <xsl:attribute name="id">
                    <xsl:value-of select="concat('update-', @id)"/>
                  </xsl:attribute>
                  <head>
                    <action>set</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='@id'/>"]
                        /cp:conditions/mmt-serv:served-identity
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                        xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
                    </xsl:element>
                  </head>
                  <body>
                    <mmt-serv:served-identity>
                      <!-- Copy the mmt-serv:one elements not including the referred identities. -->
                      <xsl:call-template name="copyNotReferredOnes">
                         <xsl:with-param name="ruleName" select="@id"/>
                         <xsl:with-param name="identities" select="$identities"/>
                      </xsl:call-template>
                    </mmt-serv:served-identity>
                  </body>
                </task>
              </xsl:for-each>
            </xsl:if>
            <!-- Handle the target rule. -->
            <xsl:choose>
              <!-- Extend the target rule with the missing identities. -->
              <xsl:when test="$targetRuleAction = 'update'">
                <task version="1" id="extend">
                  <head>
                    <action>set</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='$targetRule'/>"]
                        /cp:conditions/mmt-serv:served-identity
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                        xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
                    </xsl:element>
                  </head>
                  <body>
                    <mmt-serv:served-identity>
                      <!-- Copy the mmt-serv:one elements not including the referred identities. -->
                      <xsl:call-template name="copyNotReferredOnes">
                         <xsl:with-param name="ruleName" select="$targetRule"/>
                         <xsl:with-param name="identities" select="$identities"/>
                      </xsl:call-template>
                      <!-- Create mmt-serv:one elements including the referred identities. -->
                      <xsl:call-template name="createReferredOnes">
                         <xsl:with-param name="aliases" select="$aliases"/>
                      </xsl:call-template>
                    </mmt-serv:served-identity>
                  </body>
                </task>
              </xsl:when>
              <!-- Create a new target rule including the referred identities. -->
              <xsl:when test="$targetRuleAction='create'">
```

```
            <task version="1" id="create">
               <head>
                 <action>set</action>
                 <content-type>application/xcap-el+xml</content-type>
                 <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                     /cp:rule[@id="<xsl:value-of select='$targetRule'/>"]
                     ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                 </xsl:element>
               </head>
               <body>
                 <cp:rule>
                   <xsl:attribute name="id">
                     <xsl:value-of select='$targetRule'/>
                   </xsl:attribute>
                   <cp:conditions>
                     <mmt-serv:served-identity>
                       <!-- Create mmt-serv:one elements including the referred identities. -->
                       <xsl:call-template name="createReferredOnes">
                          <xsl:with-param name="aliases" select="$aliases"/>
                       </xsl:call-template>
                     </mmt-serv:served-identity>
                   </cp:conditions>
                   <cp:actions>
                       <ss:forward-to>
                         <ss:target><xsl:value-of select="$target"/></ss:target>
                         <ss:notify-caller>true</ss:notify-caller>
                       </ss:forward-to>
                   </cp:actions>
                 </cp:rule>
               </body>
             </task>
           </xsl:when>
         </xsl:choose>
       </task-list>
     </xsl:otherwise>
   </xsl:choose>
 </xsl:template>
</xsl:stylesheet>
</layout>
```

Example 45 Service Data Update – Tasks are Generated

Reviewing the source document once again, it turns out that the current provisioning and the user input result in two tasks: the simple-msn-cfu-to-bob rule is deleted and the simple-msn-cfu-to-0046222222222 is updated by adding aliases 2 and 3 to the served-identity condition. Both of the generated tasks are shown in Example 46.

```
<?xml version="1.0"?>
<task-list version="1"
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy">

  <task version="1" id="delete-simple-msn-cfu-to-bob">
    <head>
      <action>delete</action>
      <content-type>application/xcap-el+xml</content-type>
      <ut-path>
        simservs/communication-diversion/cp:ruleset
            /cp:rule[@id="simple-msn-cfu-to-bob"]
            ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
      </ut-path>
    </head>
  </task>

  <task version="1" id="extend">
    <head>
      <action>set</action>
      <content-type>application/xcap-el+xml</content-type>
      <ut-path>
        simservs/communication-diversion/cp:ruleset
            /cp:rule[@id="simple-msn-cfu-to-0046222222222"]/cp:conditions
            /mmt-serv:served-identity?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
            xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
      </ut-path>
    </head>
    <body>
      <mmt-serv:served-identity>
        <mmt-serv:one id="sip:+46111111111@operator.com"/>
        <mmt-serv:one id="sip:+46111111112@operator.com"/>
        <mmt-serv:one id="sip:+46111111113@operator.com"/>
      </mmt-serv:served-identity>
    </body>
  </task>

</task-list>
```

Example 46 Result of the XSLT Transformation – A Task List with Two Tasks

The announcement played to the user is the following: `The requested identities have been added to your MSN CFU rule of the entered target number.` (media-id=”1311”)

4.4.2 Support for MSN CFU Delete Command

This section contains an example for a composite configuration which analyzes the service data in the first step and according to the actual content either it updates the service data or keeps it unchanged and plays an appropriate announcement to the user at the end. The example deletes the referred alias identities from the ‘served-identity’ condition of a simple MSN enhanced CFU rule. If the referred aliases are the only identities in the ‘served-identity’ condition, the whole rule is removed. These rules are members of the CDIV rule set, see Example 35.

4.4.2.1 Subject of the Example

The user story declares a subject of this example.

An IMS user wants to delete one or more simple MSN enhanced CFU rules using the SSC command format: `*11*<aliases>##21#` so that incoming calls to the first, second, third (123), or both alias PUIs are not unconditionally (21) diverted to any target.

Example for a related SSC command:

```
*11*23##21#
```

The user wants to be informed of the following:

- If the referred identities have been removed from the MSN CFU rules
- If MSN CFU rules containing the referred identities have been removed
- If there are no active MSN CFU rules including the referred identities
- If there are no active MSN CFU rules
- If an error occurred

4.4.2.2 Configuration Overview

This section shows an example configuration of an `MtasGenSscCmd` MO instance which is in line with the description in Section 4.4.2.1 Subject of the Example on page 85.

MtasGenSscCmd: MsnCfuDel (key)

The “MsnCfuDel” string identifies this MtasGenSscCmd MO instance.

```
mtasGenSscCmdHeader:

*11*1##21|*11*2##21|*11*3##21|*11*12##21|*11*13##21|*11
*23##21|*11*123##21
```

The command header attribute lists the variations of leading digits related to this configuration. They differ with the alias IDs in the middle.

The command syntax chosen in this example allows for using of three aliases at most, owing to the decoding tree limitations. Using more aliases would be possible with a different SSC syntax, for example,`#1121*<aliases>#`. In that case, the command header attribute would have the value of `#1121*`.

```
mtasGenSscCmdSyntax:

/^\*11\*([1-3]{1,3})##21#$/schema=composite; digits aliases=\1/
```

The Command Pattern `^\*11\*([1-3]{1,3})##21#$` declares the format of the Request URI.

The Command Syntax part `schema=composite; digits aliases=\1` informs the GenSSC engine that a composite configuration is processed. It means that the task or report is extracted from the XSLT style sheet included in the layout.xml document of the `mtasGenSscCmdLayout` attribute. It also declares one variable to be fetched from an SSC command: aliases of digits type to be taken from the first variable part of the command.

`mtasGenSscCmdUtPath:`

```
simservs/communication-diversion~~simservs/mmt-serv:⇒
flexible-identity-presentation
```

The `UtPath` points to the CDIV and FIP services, they are copied from the *<service-data>* to the *<source>* document.

`mtasGenSscCmdAnnouncements:`

The announcement table is described in Section 4.4.2.3.3 Announcement Configuration on page 89.

`mtasGenSscCmdLayout:`

The layout document is described inSection 4.4.2.3.6 Layout Implementation on page 91.

Other attributes:

The attributes `mtasGenSscCmdReplace`, `mtasGenSscCmdContentType`, and `mtasGenSscCmdUriInChargingMessages` of the `MtasGenSscCmd` MO are not used in this configuration, the default values are assigned to them.

4.4.2.3 Workflow Overview

This section shows the workflow overview.

4.4.2.3.1 Understanding the Details

The details of the Delete command are similar to the Set command detailed inSection 4.4.1.3.1 Understanding the Details on page 72.

4.4.2.3.2 Clarification of the Use Cases

The user's service profile includes the following information:

- If any simple MSN enhanced CFU rules exist in the CDIV rule set
- If any simple MSN enhanced CFU rules for alias IDs given in the SSC command exist in the CDIV rule set

- If the existing simple MSN enhanced CFU rules for the alias IDs given in the SSC command contain also other alias identities in the ‘served-identity’ condition or the referred ones only

The combinations of these CFU rule information categories define the use cases of the user story. They are the input parameters which help to determine the following:

- The aliases from the existing rules to be deleted
- The whole rules to be deleted
- The announcement to be played to the user

The service data must be modified and the user must be provided with a meaningful announcement as well.

Table 8 summarizes the use cases to be considered and the announcements to be played for each use case.

Table 8 Use Cases of SSC Command for Deleting MSN Enhanced CFU Rules

Use Case	Simple Rules Exist	Simple Rules for Given Aliases Exist	A Simple Rule for Given Aliases Contains Also Other Aliases	To Be Deleted	Announcement
UC-0	Yes	Yes	Yes	Aliases from the existing rules and optionally whole rules	The referred identities have been removed from the MSN CFU rules. You still have active MSN CFU rules.
UC-1	Yes	Yes	No	Whole rules	Whole MSN CFU rules have been removed.
UC-2	Yes	No	No	Nothing	There are no active MSN CFU rules including the referred identities.
UC-3	Yes	No	No	Nothing	You have no active MSN CFU rules.

A summary of the announcement variations is shown in Table 9.

Table 9 Summary of Announcement Variations - Deleting MSN Enhanced CFU Rules

Announcement	Media ID	Use Cases
The referred identities have been removed from the MSN CFU rules. You still have active MSN CFU rules.	1181	0

Announcement	Media ID	Use Cases
Whole MSN CFU rules have been removed.	1182	1
There are no active MSN CFU rules including the referred identities.	1191	2
You have no active MSN CFU rules.	1192	3

Table 9 is divided into two parts, as follows:

- Use cases “0” and “1”represent real tasks. The service data is modified according to the use case.
- Use cases “2” and “3”represent dummy tasks. There is no need to change service data since the current value and the new value are the same. It is enough to create a proper report.

The layout creates a positive *<report>* for the use cases “2” and “3” and generates a *<task>* document for the use cases “0” and “1”.

4.4.2.3.3 Announcement Configuration

The layout is constructed in a way which:

- Performs the required modifications on the service data.
- Provides the necessary information for the announcement.

In this example, the “id” attributes of the tasks and reports are given descriptive values, which enhance readability and simplify configuration of the announcement table, Example 47.

```
<?xml version="1.0" encoding="utf-8"?>
 <announcements xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
   <announcement comment="There are no active MSN CFU rules including the referred identities.⇒
" media-id="1191">
     <play-condition id="already_deactivated"/>
   </announcement>
   <announcement comment="You have no active MSN CFU rules." media-id="1192">
     <play-condition id="no_active_rule"/>
</announcement>
<announcement comment="Internal error occurred" media-id="1800">
     <play-condition list-length="0"/>
   </announcement>
   <announcement comment="The referred identities have been removed from the MSN CFU rules.
You still have active
         MSN CFU rules." media-id="1181">
     <play-condition id="update"/>
   </announcement>   <announcement comment="Whole MSN CFU rules have been removed."⇒
media-id="1182">     <play-condition id="delete"/>
   </announcement>
   <announcement comment="Internal error occurred" media-id="1800"/>
 </announcements>
```

Example 47 Announcement Table for Deleting MSN Enhanced CFU Rules

4.4.2.3.4 Constructing Source Document Variants

An instance of the possible <source> document variants is shown in Example 48. This instance represents the use case “0”. The comments are not generated by the GenSSC engine, they serve only for better understanding.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This document is passed to the XSL Transform function as the source XML of the transformation.
     The <service-data> element represents a provisioning of the use case 0. -->
<source version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">

  <!-- The <user-id> comes from the P-Asserted-Identity header of the SSC-INVITE. -->
  <user-id>sip:abc@ericsson.com</user-id>

  <!-- The <command-data> contains the elements of the extracted command syntax.
       It includes the schema type in this case. -->
  <command-data version="1">
    <schema>composite</schema>
  </command-data>

  <!-- The <service-data> contains a piece of the user's simservs document. The mtasGenSscCmdUtPath
       determines which elements shall be taken over from the user's service data.
       There are two <service-data> elements in this case, including the CDIV and FIP services. -->
  <service-data index="0" valid="true" version="1"
      path="simservs/communication-diversion">
    <ss:communication-diversion active="true">
      <cp:ruleset>
        <!-- The "msn-video-to-alice" rule is a combined 'served-identity' and 'media' rule. It is not a
             simple rule that is why it is out of scope and is filtered out. -->
        <cp:rule id="msn-video-to-alice">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111112@operator.com"/>
            </mmt-serv:served-identity>
            <ss:media>video</ss:media>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:alice@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfu-to-0046222222222" rule is an active simple MSN enhanced CFU rule.
             Alias 2 will be removed from the served-identity condition of this rule. -->
        <cp:rule id="simple-msn-cfu-to-0046222222222">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111111@operator.com"/>
              <mmt-serv:one id="sip:+46111111112@operator.com"/>
            </mmt-serv:served-identity>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>tel:0046222222222</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfu-to-bob" rule is an active simple MSN enhanced CFU rule. Because there are
             no other aliases in the served-identity condition of this rule, the whole rule will be deleted.-->
        <cp:rule id="simple-msn-cfu-to-bob">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111113@operator.com"/>
            </mmt-serv:served-identity>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:bob@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
```

```
      </cp:ruleset>
    </ss:communication-diversion>
  </service-data>

  <service-data index="1" valid="true" version="1"
      path="simservs/mmt-serv:flexible-identity-presentation">
    <mmt-serv:flexible-identity-presentation active="true">
      <!-- The fip-identity element is not analyzed by the Generic SSC according to this example ⇒
configuration.
           It can contain either a router address or a main number of the subscriber. In the latter case,
           the main number should also be configured in the msn-fip-identity with id="1". -->
      <mmt-serv:fip-identity>sip:something@operator.com</mmt-serv:fip-identity>
      <!-- Alias ID "1" in an SSC command will be replaced with alias PUI sip:+46111111111@operator.com in
           operations on the CDIV service data. This is the subscriber's main (first alias) number . -->
      <mmt-serv:msn-fip-identity id="1">
        <mmt-serv:identity>sip:+46111111111@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "2" in an SSC command will be replaced with alias PUI sip:+46111111112@operator.com in
           operations on the CDIV service data. This is the subscriber's second alias number. -->
      <mmt-serv:msn-fip-identity id="2">
        <mmt-serv:identity>sip:+46111111112@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "3" in an SSC command will be replaced with alias PUI sip:+46111111113@operator.com in
           operations on the CDIV service data. This is the subscriber's third alias number. -->
      <mmt-serv:msn-fip-identity id="3">
        <mmt-serv:identity>sip:+46111111113@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
    </mmt-serv:flexible-identity-presentation>
  </service-data>
</source>
```

Example 48 Generated <source> Document

4.4.2.3.5 Layout Design

The following internal data is defined in this example:

- List of the identities (alias PUIs) addressed by the aliases (alias IDs) separated by the "|" symbol, that is, "|identity-1|identity-2|". "|" means that there are no matching identities (identities).
- List of the IDs of the simple MSN enhanced CFU rules which contain PUIs stored in the 'identities' variable, separated by the "|" symbol, for example, "|id-1|id-2|". "|" means that there are no matching rules (impactedRules).
- List of the IDs of the simple MSN enhanced CFU rules which have to be updated, a subset of the affected rules to be kept which contain alias PUIs to be removed, separated by the "|" symbol, for example, "|id-1|id-2|". "|" means that there are no such rules (rulesToBeUpdated).
- List of the IDs of the simple MSN enhanced CFU rules which have to be deleted – a subset of the affected rules complementary to the set of rules to be updated, separated by the "|" symbol, for example, "|id-1|id-2|". "|" means that there are no such rules (rulesToBeDeleted).

4.4.2.3.6 Layout Implementation

This chapter includes a concrete layout implementation of the user story in Section 4.4.2.1 Subject of the Example on page 85.

The top part of the layout includes a definition of one helper template, see Example 49.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This layout document contains an XSL style sheet which removes the referred served
     identities from the MSN CFU rules.
     Input parameters coming from the command data:
      - aliases: includes the MSN aliases referring to the identities to be removed
     Service data elements used by the style sheet:
      - 0: /simservs/communication-diversion
      - 1: /simservs/flexible-identity-presentation
-->
<layout version="1"
    name="Removal of the referred served identities from the MSN CFU rules"
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:gs="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
      xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
      xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
      xmlns:cp="urn:ietf:params:xml:ns:common-policy"
      exclude-result-prefixes="gs">

    <xsl:output method="xml" indent="yes"/>

    <!-- Copy the mmt-serv:one elements of a given rule not including the referred
     identities. -->
    <xsl:template name="copyNotReferredOnes"
        match="/">
        <xsl:param name="ruleName"/>
        <xsl:param name="identities"/>
        <xsl:for-each select="/gs:source/gs:service-data[@index='0']/
          ss:communication-diversion/cp:ruleset
          /cp:rule[@id=$ruleName]/cp:conditions/mmt-serv:served-identity
          /mmt-serv:one[not(contains($identities, concat('|', @id, '|')))]">
          <xsl:for-each select="@*">
            <xsl:element name="mmt-serv:one">
              <xsl:attribute name="id">
                <xsl:value-of select="."/>
              </xsl:attribute>
            </xsl:element>
          </xsl:for-each>
        </xsl:for-each>
    </xsl:template>

    <!-- Main template. -->
    <xsl:template match="/">
```

Example 49 Helper Template

The next part of the layout includes the definitions of the *<xsl:variable>* elements: see Example 50.

```
<!-- ****************************************
     * Variable definitions                 *
     **************************************** -->

<!-- Store the alias IDs from the SSC command in a variable. -->
<xsl:variable name="aliases">
  <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='aliases']"/>
</xsl:variable>

<!-- Collect the identities (alias PUIs) addressed by the aliases (alias IDs) in a string and separate
     them by colon '|' symbol.
     Example: |identity-1|identity-2|
     "|" means there are no matching identities -->
<xsl:variable name="identities">
  <xsl:for-each select="/gs:source/gs:service-data[@index='1']
       /mmt-serv:flexible-identity-presentation/mmt-serv:msn-fip-identity[contains($aliases, @id)]
       /mmt-serv:identity">
    <xsl:value-of select="concat('|', text())"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Collect the ids of the simple MSN enhanced CFU rules which contain PUIs stored in the ⇒
$identities.-->
<xsl:variable name="impactedRules">
  <xsl:for-each select="/gs:source/gs:service-data[@index='0']
       /ss:communication-diversion/cp:ruleset/cp:rule[
       count(cp:conditions/*)=1
       and count(cp:conditions/mmt-serv:served-identity/mmt-serv:one[
       contains($identities, concat('|', @id, '|'))]) != 0
       ]">
    <xsl:value-of select="concat('|', @id)"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Collect the ids of the simple MSN enhanced CFU rules which have to be updated: a subset
     of impacted rules to be kept which contain PUIs to be removed. -->
<xsl:variable name="rulesToBeUpdated">
  <xsl:for-each select="/gs:source/gs:service-data[@index='0']
       /ss:communication-diversion/cp:ruleset/cp:rule[contains($impactedRules, concat('|', @id, '|'))
       and count(cp:conditions/mmt-serv:served-identity
       /mmt-serv:one[not(contains($identities, concat('|', @id, '|')))]) != 0
       ]">
    <xsl:value-of select="concat('|', @id)"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Collect the ids of the simple MSN enhanced CFU rules which have to be deleted:
     a subset of the impacted rules complementary to the set of rules to be updated. -->
<xsl:variable name="rulesToBeDeleted">
  <xsl:for-each select="/gs:source/gs:service-data[@index='0']
       /ss:communication-diversion/cp:ruleset/cp:rule[
       contains($impactedRules, concat('|', @id, '|'))
       and not(contains($rulesToBeUpdated, concat('|', @id, '|')))
       ]">
    <xsl:value-of select="concat('|', @id)"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>
```

Example 50 Variable Definitions

The bottom part of the layout document comprises the generation of the output. It is divided into several sections. They refer to the use cases of Table 8.

The first section refers to the use cases “2 ” and “3 ”of Table 8, see Example 51.

```
<!-- ****************************************
     * Output generation                    *
     **************************************** -->

<xsl:choose>
  <!-- There is nothing to do: the referred identities are not included in any MSN CFU rule. -->
  <xsl:when test="$impactedRules='|'">
    <report version="1" id="already-deactivated"
        xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
      <xsl:attribute name="id">
        <xsl:choose>
          <xsl:when test="count(/gs:source/gs:service-data[@index='0']
              /ss:communication-diversion/cp:ruleset/cp:rule[
              count(cp:conditions/*)=1
              and count(cp:conditions/mmt-serv:served-identity)=1
              ]) != 0">
            <xsl:value-of select="'already_deactivated'"/>
          </xsl:when>
          <xsl:otherwise><xsl:value-of select="'no_active_rule'"/></xsl:otherwise>
        </xsl:choose>
      </xsl:attribute>
      <response>200 OK</response>
      <job>delete</job>
      <value type="empty"/>
    </report>
  </xsl:when>
```

Example 51 Dummy Task Use Cases – Positive Reports are Generated

The ‘service data update’ section creates `<task>` documents to modify the service data according to the user input. Once the `<task>` has been generated, the further processing is the same as described in Section 4.1.3 Workflow Overview on page 36. An appropriate XCAP request is sent to the XCAP servlet which applies the service-specific application rules on the request and updates the HSS with the modified service data. The GenSSC engine creates a `<report>` from the XCAP response (the report ID is inherited from the task ID). The report is compared with the announcement table and the matching voice message is sent to the user. The tasks of the service data update are shown in Example 52.

```
        <!-- The service data needs to be modified. -->
        <xsl:otherwise>
          <task-list version="1"
              xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
              xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
              xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
              xmlns:cp="urn:ietf:params:xml:ns:common-policy">
            <!-- Remove the rules to be deleted. -->
            <xsl:if test="$rulesToBeDeleted != '|'">
              <xsl:for-each select="/gs:source/gs:service-data[@index='0']⇒
/ss:communication-diversion/cp:ruleset
                  /cp:rule[contains($rulesToBeDeleted, concat('|', @id, '|'))]">
                <task version="1">
                  <xsl:attribute name="id">
                    <xsl:value-of select="concat('delete-', @id)"/>
                  </xsl:attribute>
                  <head>
                    <action>delete</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='@id'/>"]
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                    </xsl:element>
                  </head>
                </task>
              </xsl:for-each>
            </xsl:if>
            <!-- Remove the referred identities from the rules to be updated. -->
            <xsl:if test="$rulesToBeUpdated != '|'">
              <xsl:for-each select="/gs:source/gs:service-data[@index='0']⇒
/ss:communication-diversion/cp:ruleset
                  /cp:rule[contains($rulesToBeUpdated, concat('|', @id, '|'))]">
                <task version="1">
                  <xsl:attribute name="id">
                    <xsl:value-of select="concat('update-', @id)"/>
                  </xsl:attribute>
                  <head>
                    <action>set</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='@id'/>"]
                        /cp:conditions/mmt-serv:served-identity
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                        xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
                    </xsl:element>
                  </head>
                  <body>
                    <mmt-serv:served-identity>
                      <!-- Copy the mmt-serv:one elements not including the referred identities. -->
                      <xsl:call-template name="copyNotReferredOnes">
                         <xsl:with-param name="ruleName" select="@id"/>
                         <xsl:with-param name="identities" select="$identities"/>
                      </xsl:call-template>
                     </mmt-serv:served-identity>
                  </body>
                </task>
              </xsl:for-each>
            </xsl:if>
          </task-list>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:template>
  </xsl:stylesheet>
</layout>
```

Example 52 Service Data Update – Tasks are Generated

Reviewing the source document once again, it turns out that the current provisioning and the user input result in two tasks: the simple-msn-cfu-to-bob rule is deleted and the simple-msn-cfu-to-0046222222222 is updated by removing alias 2 from the served-identity condition. Both of the generated tasks are shown in Example 53.

```
<?xml version="1.0"?>
<task-list version="1"
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy">

  <task version="1" id="delete-simple-msn-cfu-to-bob">
    <head>
      <action>delete</action>
      <content-type>application/xcap-el+xml</content-type>
      <ut-path>
        simservs/communication-diversion/cp:ruleset
            /cp:rule[@id="simple-msn-cfu-to-bob"]?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
      </ut-path>
    </head>
  </task>

  <task  version="1" id="update-simple-msn-cfu-to-0046222222222">
    <head>
      <action>set</action>
      <content-type>application/xcap-el+xml</content-type>
      <ut-path>
        simservs/communication-diversion/cp:ruleset
            /cp:rule[@id="simple-msn-cfu-to-0046222222222"]/cp:conditions
            /mmt-serv:served-identity?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
            xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
      </ut-path>
    </head>
    <body>
      <mmt-serv:served-identity>
        <mmt-serv:one id="sip:+46111111111@operator.com"/>
      </mmt-serv:served-identity>
    </body>
  </task>

</task-list>
```

Example 53 Result of the XSLT Transformation – A Task List with Two Tasks

The announcement played to the user is the following: `The referred identities have been removed from the MSN CFU rules. You still have active MSN CFU rules.` (media-id=”1181”)

4.4.3 Support for MSN CFU Get Command

This section contains an example for a composite configuration which analyzes the service data keeping it unchanged and plays an appropriate announcement to the user. The interrogation operation checks if there exists any simple MSN enhanced CFU rule for a given alias. The verification operation checks if there exists any simple MSN enhanced CFU rule for a given alias and if it has a specific target. These rules are members of the CDIV rule set, see Example 35.

4.4.3.1 Subject of the Example

The user story declares a subject of this example.

An IMS user wants to interrogate the state or verify the forward-to target value of the MSN CFU rule referred by the alias, using the SSC command formats: `*11*<alias>#*#21#` (interrogation) and `*11*<alias>#*#21<target>#` (verification).

Example for a related SSC command (interrogation):

```
*11*2#*#21#
```

Example for a related SSC command (verification):

```
*11*2#*#210046222222222#
```

The user wants to be informed of the following:

- If there is any active simple MSN enhanced CFU rule including the referred identity (interrogation and verification)
- If the simple MSN enhanced CFU rule for the referred identity includes the entered target number (verification)
- If an error has occurred

4.4.3.2 Configuration Overview

This section shows an example configuration of an `MtasGenSscCmd` MO instance.

`MtasGenSscCmd: MsnCfuGet` (key)

The "MsnCfuGet" string identifies this `MtasGenSscCmd` MO instance.

```
mtasGenSscCmdHeader: *11*1#*#21|*11*2#*#21|*11*3#*#21
```

The command header attribute lists the variations of leading digits related to this configuration. They differ with the alias ID in the middle.

The command syntax chosen in this example allows for using of three aliases at most, owing to the decoding tree limitations. Using more aliases would be possible with a different SSC syntax, for example,`#*1121*<alias>#` and `#*1121*<alias>*<target>#`.

In that case, the command header attribute would have the value of `#*1121*`.

```
mtasGenSscCmdSyntax:
/^\*11\*([1-3])#\*#21#$/schema=composite; number alias=\1;⇒
 item target=none/~~
/^\*11\*([1-3])#\*#21([0-9]+)#$/schema=composite; ⇒
number alias=\1; item target=tel:\2/
```

For interrogation and verification operations, the `mtasGenSscCmdSyntax` consists of two parts.

First Part

The first part `^\*11\*([1-3])#\*#21#$/schema=composite; number alias=\1; item target=none` defines a syntax for the interrogation operation.

The Command Pattern `^\*11\*([1-3])#\*#21#$` declares the format of the Request URI.

The Command Syntax part `schema=composite; number alias=\1; item target=none` informs the GenSSC engine that a composite configuration is processed. It means that the task or report is extracted from the XSLT style sheet included in the layout.xml document of the `mtasGenSscCmdLayout` attribute. It also declares two variables to be fetched from an SSC command: alias of number type to be taken from the first variable part of the command and target of item type which is set to none.

Second Part

The second part `^\*11\*([1-3])#\*#21([0-9]+)#$/schema=composite; number alias=\1; item target=tel:\2` defines a syntax for the verification operation.

The Command Pattern `^\*11\*([1-3])#\*#21([0-9]+)#$`declares the format of the Request URI.

The Command Syntax part `schema=composite; number alias=\1; item target=tel:\2` informs the GenSSC engine that a composite configuration is processed. It also declares two variables to be fetched from an SSC command alias of number type to be taken from the first variable part of the command and target of item type to be taken from the second variable part.

```
mtasGenSscCmdUtPath:
simservs/communication-diversion~~simservs/mmt-serv:⇒
flexible-identity-presentation
```

The `UtPath` points to the CDIV and FIP services, they are copied from the *<service-data>* to the *<source>* document.

```
mtasGenSscCmdAnnouncements:
```

The announcement table is described in Section 4.4.3.3.3 Announcement Configuration on page 100.

```
mtasGenSscCmdLayout:
```

The layout document is described in Section 4.4.3.3.6 Layout Implementation on page 102.

4.4.3.3 Workflow Overview

This section shows the workflow overview.

4.4.3.3.1 Understanding the Details

The details of the Delete command are similar to the Set command detailed in Section 4.4.1.3.1 Understanding the Details on page 72.

4.4.3.3.2 Clarification of the Use Cases

The user's service profile includes the following information:

- If any simple active MSN enhanced CFU rules for the alias ID given in the SSC command exist in the CDIV rule set
- If any simple active MSN enhanced CFU rules for the alias ID and the target given in the SSC command exist in the CDIV rule set

The type of operation (interrogation or verification) is determined by the SSC command syntax.

The combinations of these CFU rule information categories define the use cases of the user story. They are the input parameters which help to determine an announcement to be played to the user.

Table 10 summarizes the use cases to be considered and the announcements to be played for each use case.

Table 10 Use Cases of SSC Command for Interrogating or Verifying an MSN Enhanced CFU Rule

Use Case	Type of Operation	Type of Operation	A Simple Rule for a Given Alias AND a Given Target Exists	Announcement
UC-0	Irrelevant	No	No	There are no active MSN CFU rules including the referred identity.
UC-1	Interrogation	Yes	Irrelevant	Your MSN CFU rule including the referred identity is active.
UC-2	Verification	Yes	Yes	The MSN CFU rule for the referred identity includes the entered target number.
UC-3	Verification	Yes	No	The MSN CFU rule for the referred identity includes a different target.

A summary of the announcement variations is shown in Table 11.

Table 11 Summary of Announcement Variations - Interrogating or Verifying an MSN Enhanced CFU Rule

Announcement	Media ID	Use Cases
There are no active MSN CFU rules including the referred identity.	1241	0
Your MSN CFU rule including the referred identity is active.	1242	1
The MSN CFU rule for the referred identity includes the entered target number.	1243	2
The MSN CFU rule for the referred identity includes a different target.	1244	3

All the use cases in Table 6 represent dummy tasks. The service data is not changed. It is enough to create a proper report.

The layout creates a positive *<report>* for every use case.

4.4.3.3.3 Announcement Configuration

The layout is constructed in a way which provides the necessary information for the announcement.

In Example 54, the ‘‘id’’ attributes of the reports are given descriptive values, which enhance readability and simplify configuration of the announcement table.

```
<?xml version="1.0" encoding="utf-8"?>
 <announcements xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
   <announcement comment="There are no active MSN CFU rules including the referred identity."
   media-id="1241">
     <play-condition id="deactivated"/>
   </announcement>   <announcement comment="Your MSN CFU rule including the referred identity is active.
    " media-id="1242">
     <play-condition id="active"/>
   </announcement>
   <announcement comment="The MSN CFU rule for the referred identity includes a different target."
      media-id="1244">
     <play-condition id="mismatch"/>
   </announcement>   <announcement comment="The MSN CFU rule for the referred identity includes the
    entered target number."media-id="1243">
     <play-condition id="match"/>
   </announcement>
   <announcement comment="Internal error occurred" media-id="1800"/>
 </announcements>
```

Example 54 Announcement Table for Interrogating or Verifying an MSN Enhanced CFU Rule

4.4.3.3.4 Constructing Source Document Variants

An instance of the possible *<source>* document variants is shown in Example 55. This instance represents the use cases ‘‘1’’ and ‘‘2’’. The comments are not generated by the GenSSC engine, they serve only for better understanding.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This document is passed to the XSL Transform function as the source XML of the transformation.
     The <service-data> element represents a provisioning of the use cases 1 and 2. -->
<source version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">

  <!-- The <user-id> comes from the P-Asserted-Identity header of the SSC-INVITE. -->
  <user-id>sip:abc@ericsson.com</user-id>

  <!-- The <command-data> contains the elements of the extracted command syntax.
       It includes the schema type in this case. -->
  <command-data version="1">
    <schema>composite</schema>
  </command-data>

  <!-- The <service-data> contains a piece of the user's simservs document. The mtasGenSscCmdUtPath
        determines which elements shall be taken over from the user's service data.
       There are two <service-data> elements in this case, including the CDIV and FIP services. -->
  <service-data index="0" valid="true" version="1"
      path="simservs/communication-diversion">
    <ss:communication-diversion active="true">
      <cp:ruleset>
        <!-- The "msn-video-to-alice" rule is a combined 'served-identity' and 'media' rule. It is not a
             simple rule that is why it is out of scope and is filtered out. -->
        <cp:rule id="msn-video-to-alice">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111112@operator.com"/>
            </mmt-serv:served-identity>
            <ss:media>video</ss:media>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:alice@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfu-to-0046222222222" rule is an active simple MSN enhanced CFU rule. The
             'served-identity' condition of the rule contains the identity referred in the SSC command
             of the user story and the forward-to target is also the same as in the SSC command. -->
        <cp:rule id="simple-msn-cfu-to-0046222222222">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111112@operator.com"/>
            </mmt-serv:served-identity>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>tel:0046222222222</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfu-to-bob" rule is an active simple MSN enhanced CFU rule. It has a different
             alias and forward-to target than the alias and target in the SSC command of the user story. -->
        <cp:rule id="simple-msn-cfu-to-bob">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111113@operator.com"/>
            </mmt-serv:served-identity>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:bob@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
      </cp:ruleset>
    </ss:communication-diversion>
  </service-data>

  <service-data index="1" valid="true" version="1"
      path="simservs/mmt-serv:flexible-identity-presentation">
    <mmt-serv:flexible-identity-presentation active="true">
      <!-- The fip-identity element is not analyzed by the Generic SSC according to this example
```

```
          configuration. It can contain either a router address or a main number of the subscriber. In the
           latter case, the main number should also be configured in the msn-fip-identity with id="1". -->
      <mmt-serv:fip-identity>sip:something@operator.com</mmt-serv:fip-identity>
      <!-- Alias ID "1" in an SSC command will be replaced with alias PUI sip:+46111111111@operator.com in
          operations on the CDIV service data. This is the subscriber's main (first alias) number . -->
      <mmt-serv:msn-fip-identity id="1">
        <mmt-serv:identity>sip:+46111111111@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "2" in an SSC command will be replaced with alias PUI sip:+46111111112@operator.com in
          operations on the CDIV service data. This is the subscriber's second alias number. -->
      <mmt-serv:msn-fip-identity id="2">
        <mmt-serv:identity>sip:+46111111112@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "3" in an SSC command will be replaced with alias PUI sip:+46111111113@operator.com in
          operations on the CDIV service data. This is the subscriber's third alias number. -->
      <mmt-serv:msn-fip-identity id="3">
        <mmt-serv:identity>sip:+46111111113@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
    </mmt-serv:flexible-identity-presentation>
  </service-data>
</source>
```

Example 55 Generated <source> Document

4.4.3.3.5 Layout Design

The following internal data is defined in this example:

- The alias ID from the SSC command (alias)
- The target from the SSC command including the 'tel:' prefix (target)
- The identity (alias PUI) addressed by the alias ID (identity)
- The ID of the simple MSN enhanced CFU rule which contains PUI stored in the 'identity' variable (ruleName)
- The target of the rule with an ID stored in the 'ruleName' variable

4.4.3.3.6 Layout Implementation

This chapter includes a concrete layout implementation of the user story in Section 4.4.6.1 Subject of the Example on page 144.

The top part of the layout includes definitions of the *<xsl:variable>* elements, see Example 56.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This layout document contains an XSL style sheet which interrogates the state or
     verifies the forward-to target value of the MSN CFU rule referred by the alias.
     Input parameters coming from the command data:
      - alias: includes the MSN alias implicitly referring to the rule to be interrogated
      - target: forward-to target number including the 'tel:' prefix or 'none'
     Service data elements used by the style sheet:
      - 0: /simservs/communication-diversion
      - 1: /simservs/flexible-identity-presentation
-->
<layout version="1"
    name="Interrogation of the MSN CFU rule referred by the alias"
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
      xmlns:gs="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
      xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
      xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
      xmlns:cp="urn:ietf:params:xml:ns:common-policy"
      exclude-result-prefixes="gs">

    <xsl:output method="xml" indent="yes"/>

    <!-- Main template. -->
    <xsl:template match="/">

      <!-- ****************************************
           * Variable definitions                 *
           **************************************** -->

      <!-- Store the alias in a variable. -->
      <xsl:variable name="alias">
        <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='alias']"/>
      </xsl:variable>

      <!-- Store the target including the 'tel:' prefix in a valiable. -->
      <xsl:variable name="target">
        <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='target']"/>
      </xsl:variable>

      <!-- Store the identity addressed by the alias in a variable. -->
      <xsl:variable name="identity">
        <xsl:value-of select="/gs:source/gs:service-data[@index='1']
            /mmt-serv:flexible-identity-presentation/mmt-serv:msn-fip-identity[@id=$alias]"/>
      </xsl:variable>

      <!-- Store the name of the rule addressed by the alias in a variable. -->
      <xsl:variable name="ruleName">
        <xsl:value-of select="/gs:source/gs:service-data[@index='0']
            /ss:communication-diversion/cp:ruleset/cp:rule[
            count(cp:conditions/*)=1
            and count(cp:conditions/mmt-serv:served-identity/mmt-serv:one[@id=$identity])=1
            ]/@id"/>
      </xsl:variable>

      <!-- Store the rule's target in a valiable. -->
      <xsl:variable name="ruleTarget">
        <xsl:value-of select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion/cp:ruleset
            /cp:rule[@id=$ruleName]/cp:actions/ss:forward-to/ss:target"/>
      </xsl:variable>
```

Example 56 Variable Definitions

The bottom part of the layout document comprises the generation of the output. It is divided into several sections. They refer to the use cases of Table 10, see Example 57.

```
<!-- ****************************************
     * Output generation                    *
     **************************************** -->

      <xsl:choose>
        <xsl:when test="(string-length($ruleName)=0) and ($target='none')">
          <!-- No rule found - the rule for a given MSN identity has been deactivated. -->
          <report version="1" id="deactivated" xmlns="http://uri.etsi.org/ngn/params/xml/simservs
            /xcap/genssc">
            <response>200 OK</response>
            <job>get</job>
            <value type="empty"/>
          </report>
        </xsl:when>
        <xsl:when test="$target='none'">
          <!-- Rule found - the rule for a given MSN identity is active. -->
          <report version="1" id="active" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
            <response>200 OK</response>
            <job>get</job>
            <value type="empty"/>
          </report>
        </xsl:when>
        <xsl:when test="$ruleTarget=$target">
          <!-- The rule's target matches the input. -->
          <report version="1" id="match" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
            <response>200 OK</response>
            <job>verify</job>
            <value type="simple"><xsl:value-of select="$ruleTarget"/></value>
          </report>
        </xsl:when>
        <xsl:otherwise>
          <!-- The rule's target doesn't match the input. -->
          <report version="1" id="mismatch" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
            <response>200 OK</response>
            <job>verify</job>
            <value type="simple"><xsl:value-of select="$ruleTarget"/></value>
          </report>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:template>
  </xsl:stylesheet>
</layout>
```

Example 57 Generation of the Output – Positive Reports are Generated

Reviewing the source document once again, it turns out that the current provisioning and the user input result in positive reports in both cases of interrogation and verification operations of the user story. The simple-msn-cfu-to-0046222222222 rule is an active simple MSN enhanced CFU rule with the ‘served-identity’ condition containing the identity referred in the SSC commands and the same forward-to target. The generated reports for both operations are shown in Example 58 and Example 59.

```
<?xml version="1.0"?>
<report version="1" id="active" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <response>200 OK</response>
  <job>get</job>
  <value type="empty"/>
</report>
```

Example 58 Result of the Interrogation Operation – A Positive Report

```
<?xml version="1.0"?>
<report version="1" id="match" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <response>200 OK</response>
  <job>verify</job>
  <value type="simple">simple-msn-cfu-to-0046222222222</value>
</report>
```

Example 59 Result of the Verification Operation – A Positive Report

If the interrogation operation the announcement played to the user is: `Your MSN CFU rule including the referred identity is active.` (media-id=”1242”)

If the verification operation the announcement played to the user is: `The MSN CFU rule for the referred identity includes the entered target number.` (media-id=”1243”)

4.4.4 Support for MSN Conditional CDIV Set Command

This section contains an example for a complex composite configuration which analyzes the service data in the first step. According to the actual content, it either updates the service data or keeps it unchanged and plays an appropriate announcement to the user at the end. The example sets a new or updates an existing simple MSN enhanced Conditional CDIV rule. These rules are members of the CDIV rule set, see Example 35.

4.4.4.1 Subject of the Example

The user story declares a subject of this example.

An IMS user wants to set or update one or more simple MSN enhanced Conditional CDIV rules of a given type using the SSC command format: `*11*<aliases>#*<service_code><target>#` so that all incoming calls to the first, second, and third (123) alias PUIs are conditionally (67) diverted to the entered target.

Example for a related SSC command (MSN enhanced CFB):

```
*11*23#*67004622222222#
```

The user wants to be informed of the following:

- If a new rule has been created
- If an existing rule has been updated
- If such a rule exists
- If alias IDs in the SSC command are not mapped to any PUIs in FIP or the service data is broken
- If an error occurred

4.4.4.2 Configuration Overview

This section shows an example configuration of an `MtasGenSscCmd` MO instance which is in line with the above description.

```
MtasGenSscCmd:  MsnCDivCondSet (key)
```

The "MsnCDivCondSet" string identifies this `MtasGenSscCmd` MO instance.

```
mtasGenSscCmdHeader:
*11*1#*6|*11*2#*6|*11*3#*6|*11*12#*6|*11*13#*6|*11*23#*6|⇒
*11*123#*6
```

The command header attribute lists the variations of leading digits related to this configuration. They differ with the alias IDs in the middle.

The command syntax chosen in this example allows for using of three aliases at most, owing to the decoding tree limitations. Using more aliases would be possible with a different SSC syntax, for example, `*11<service_code>*<aliases>#*<target>#`. In that case, the command header attribute would have the value of `*11<first_digit_of_the_service_code>`, for example, *116.

```
mtasGenSscCmdSyntax:
/^\*11\*([1-3]{1,3})#\*61([0-9]+)#$/schema=composite; digits⇒
 aliases=\1; item target=tel:\2; item condition=ss:no-answer;⇒
item prefix=msn_cfnr_/~~
/^\*11\*([1-3]{1,3})#\*62([0-9]+)#$/schema=composite; digits⇒
 aliases=\1; item target=tel:\2; item condition=ss:not-reachable;⇒
item prefix=msn_cfnrc_/~~
/^\*11\*([1-3]{1,3})#\*63([0-9]+)#$/schema=composite; digits⇒
aliases=\1; item target=tel:\2; item condition=ss:not-registered;⇒
item prefix=msn_cfnl_/~~
/^\*11\*([1-3]{1,3})#\*67([0-9]+)#$/schema=composite; digits
 aliases=\1;item target=tel:\2; item condition=ss:busy; item⇒
prefix=msn_busy_/
```

The `mtasGenSscCmdSyntax` consists of four parts – one for each kind of conditional CDIV. Every part contains two subparts: a Command Pattern and a Command Syntax.

The Command Pattern, for example, `^\*11\*([1-3]{1,3})#\*67([0-9]+)#$` declares the format of the Request URI.

The Command Syntax part, for example, `schema=composite; digits aliases=\1; item target=tel:\2; item condition=ss:busy; item prefix=msn_busy_` informs the GenSSC engine that a composite configuration is processed. It means that the task or report is extracted from the XSLT style sheet included in the layout.xml document of the `mtasGenSscCmdLayout` attribute. It also declares four variables to be fetched from an SSC command: aliases of digit types to be taken from the first variable

part of the command, target of item type to be taken from the second variable part, condition of item type which contains an appropriate CDIV condition, and prefix of item type which contains an appropriate prefix for a new rule name.

`mtasGenSscCmdUtPath:`

`simservs/communication-diversion~~simservs/mmt-serv:⇒ flexible-identity-presentation`

The UtPath points to the CDIV and FIP services, they are copied from the *<service-data>* to the *<source>* document.

`mtasGenSscCmdAnnouncements:`

The announcement table is described in Section 4.4.4.3.3 Announcement Configuration on page 116.

`mtasGenSscCmdLayout:`

The layout document is described in Section 4.4.4.3.6 Layout Implementation on page 120.

Other attributes:

The attributes `mtasGenSscCmdReplace`, `mtasGenSscCmdContentType`, and `mtasGenSscCmdUriInChargingMessages` of the `MtasGenSscCmd` MO are not used in this configuration, the default values are assigned to them.

4.4.4.3 Workflow Overview

This section shows the workflow overview.

4.4.4.3.1 Understanding the Details

Analyze the service data pieces related to the MSN enhanced Conditional CDIV rules to understand the technical details and to discover the optional difficulties.

A simple MSN enhanced Conditional CDIV rule is a CDIV rule containing exactly two conditions: *<mmt-serv:served-identity>* and one of the following: *<ss:no-answer>*, *<ss:not-reachable>*, *<ss:not-registered>*, or *<ss:busy>*. The *<mmt-serv:served-identity>* condition contains one or more *<mmt-serv:one>* elements with one subscriber's alias PUI each.

A rule can be deactivated by inserting the *<ss:rule-deactivated/>* element into the conditions of the rule or can be deleted by removing the rule from the rule set. These different technical solutions result in the same user experience, that is, the rule is not active. A rule is active if it exists and is not deactivated. Deactivated rules are not analyzed or updated by the GenSSC configured according to this example. Only simple MSN enhanced Conditional CDIV rules, that is, the ones containing exactly two conditions ('served-identity' and one of the above conditions) are processed.

The ‘served-identity’ condition can also be combined with other conditions (like ‘validity’ and ‘identity’) but this example focuses on the simple MSN enhanced CFNR, CFNRc, CFNL, and CFB rules; the other rules are out of the scope of this example. For example, configuration against simple MSN enhanced CFU rules, where the ‘served-identity’ condition is the only condition in a CDIV rule.

A CDIV rule with the same name as the name composed by the GenSSC according to the configured template may already exist in the subscribers rule set. This case can be considered a provisioning error and the user is notified about it.

4.4.4.3.2 Clarification of the Use Cases

The user’s service profile includes the following information:

- If all alias IDs in the SSC command are mapped to the subscriber’s alias PUIs in the FIP service data
- If any simple active MSN enhanced Conditional CDIV rules of a given type for alias IDs given in the SSC command already exist in the CDIV rule set
- If any simple active MSN enhanced Conditional CDIV rules of a given type for the target given in the SSC command already exist in the CDIV rule set
- If any simple active MSN enhanced Conditional CDIV rules of a given type for all alias IDs and the target given in the SSC command already exist in the CDIV rule set

The combinations of these CFU rule information categories define the use cases of the user story. They are the input parameters which help to determine the following:

- The new rule to be created or new entries to be added to an existing rule
- The aliases from the existing rules and whole rules to be deleted
- The announcement to be played to the user

That is, not only the service data must be modified but the user must be provided with a meaningful announcement as well.

Table 12 summarizes the use cases to be considered and the announcements to be played for each use case.

Table 12 Use Cases of SSC Command for Setting or Updating an MSN Enhanced Conditional CDIV Rule

Use Case	All Alias IDs Mapped to Alias PUIs in FIP	Type of Conditional CDIV	Simple Rules of a Given Type for Given Aliases Exist	Simple Rules of a Given Type for Given Target Exist	A Simple Rule of a Given Type for Both Given Aliases AND a Given Target Exists	To Be Created or Updated	To Be Deleted	Announcement
UC-0	Yes	CFNR	No	No	No	A new rule is created	Nothing	A new MSN CFNR rule has been created with the requested identities and the entered target number.
UC-1	Yes	CFNR	Yes	No	No	A new rule is created	Aliases or whole existing rule is removed	A new MSN CFNR rule has been created with the requested identities and the entered target number.
UC-2	Yes	CFNR	No	Yes	No	New entries are added to the existing rule	Nothing	The requested identities have been added to your MSN CFNR rule of the entered target number.
UC-3	Yes	CFNR	Yes	Yes	No	New entries are added to the existing rule	Aliases or whole existing rule is removed	The requested identities have been added to your MSN CFNR rule of the entered target number.

Use Case	All Alias IDs Mapped to Alias PUIs in FIP	Type of Conditional CDIV	Simple Rules of a Given Type for Given Aliases Exist	Simple Rules of a Given Type for Given Target Exist	A Simple Rule of a Given Type for Both Given Aliases AND a Given Target Exists	To Be Created or Updated	To Be Deleted	Announcement
UC-4	Yes	CFNR	Yes	Yes	Yes	Nothing	Nothing	The referred identities are already included in your MSN CFNR rule of the entered target number.
UC-5	No	CFNR	Irrelevant	Irrelevant	Irrelevant	Nothing	Nothing	The MSN CFNR rules in your service data are corrupt or the alias ID is incorrect.
UC-6	Yes	CFNRc	No	No	No	A new rule is created	Nothing	A new MSN CFNRc rule has been created with the requested identities and the entered target number.
UC-7	Yes	CFNRc	Yes	No	No	A new rule is created	Aliases or whole existing rule is removed	A new MSN CFNRc rule has been created with the requested identities and the entered target number.

Use Case	All Alias IDs Mapped to Alias PUIs in FIP	Type of Conditional CDIV	Simple Rules of a Given Type for Given Aliases Exist	Simple Rules of a Given Type for Given Target Exist	A Simple Rule of a Given Type for Both Given Aliases AND a Given Target Exists	To Be Created or Updated	To Be Deleted	Announcement
UC-8	Yes	CFNRc	No	Yes	No	New entries are added to the existing rule	Nothing	The requested identities have been added to your MSN CFNRc rule of the entered target number.
UC-9	Yes	CFNRc	Yes	Yes	No	New entries are added to the existing rule	Aliases or whole existing rule is removed	The requested identities have been added to your MSN CFNRc rule of the entered target number.
UC-10	Yes	CFNRc	Yes	Yes	Yes	Nothing	Nothing	The referred identities are already included in your MSN CFNRc rule of the entered target number.
UC-11	No	CFNRc	Irrelevant	Irrelevant	Irrelevant	Nothing	Nothing	The MSN CFNRc rules in your service data are corrupt or the alias ID is incorrect.

Use Case	All Alias IDs Mapped to Alias PUIs in FIP	Type of Conditional CDIV	Simple Rules of a Given Type for Given Aliases Exist	Simple Rules of a Given Type for Given Target Exist	A Simple Rule of a Given Type for Both Given Aliases AND a Given Target Exists	To Be Created or Updated	To Be Deleted	Announcement
UC-12	Yes	CFNL	No	No	No	A new rule is created	Nothing	A new MSN CFNL rule has been created with the requested identities and the entered target number.
UC-13	Yes	CFNL	Yes	No	No	A new rule is created	Aliases or whole existing rule is removed	A new MSN CFNL rule has been created with the requested identities and the entered target number.
UC-14	Yes	CFNL	No	Yes	No	New entries are added to the existing rule	Nothing	The requested identities have been added to your MSN CFNL rule of the entered target number.
UC-15	Yes	CFNL	Yes	Yes	No	New entries are added to the existing rule	Aliases or whole existing rule is removed	The requested identities have been added to your MSN CFNL rule of the entered target number.

Use Case	All Alias IDs Mapped to Alias PUIs in FIP	Type of Conditional CDIV	Simple Rules of a Given Type for Given Aliases Exist	Simple Rules of a Given Type for Given Target Exist	A Simple Rule of a Given Type for Both Given Aliases AND a Given Target Exists	To Be Created or Updated	To Be Deleted	Announcement
UC-16	Yes	CFNL	Yes	Yes	Yes	Nothing	Nothing	The referred identities are already included in your MSN CFNL rule of the entered target number.
UC-17	No	CFNL	Irrelevant	Irrelevant	Irrelevant	Nothing	Nothing	The MSN CFNL rules in your service data are corrupt or the alias ID is incorrect.
UC-18	Yes	CFB	No	No	No	A new rule is created	Nothing	A new MSN CFB rule has been created with the requested identities and the entered target number.
UC-19	Yes	CFB	No	No	No	A new rule is created	Aliases or whole existing rules are removed	A new MSN CFB rule has been created with the requested identities and the entered target number.

Use Case	All Alias IDs Mapped to Alias PUIs in FIP	Type of Conditional CDIV	Simple Rules of a Given Type for Given Aliases Exist	Simple Rules of a Given Type for Given Target Exist	A Simple Rule of a Given Type for Both Given Aliases AND a Given Target Exists	To Be Created or Updated	To Be Deleted	Announcement
UC-20	Yes	CFB	No	Yes	No	New entries are added to the existing rule	Nothing	The requested identities have been added to your MSN CFB rule of the entered target number.
UC-21	Yes	CFB	Yes	Yes	No	New entries are added to the existing rule	Aliases or whole existing rules are removed	The requested identities have been added to your MSN CFB rule of the entered target number.
UC-22	Yes	CFB	Yes	Yes	Yes	Nothing	Nothing	The referred identities are already included in your MSN CFB rule of the entered target number.
UC-23	No	CFB	Irrelevant	Irrelevant	Irrelevant	Nothing	Nothing	The MSN CFB rules in your service data are corrupt or the alias ID is incorrect.

A summary of the announcement variations is shown in Table 13.

Table 13 Summary of Announcement – Setting or Updating MSN Enhanced Conditional CDIV Rule

Announcement	Media ID	Use Cases
A new MSN CFNR rule has been created with the requested identities and the entered target number.	1322	0, 1
The requested identities have been added to your MSN CFNR rule of the entered target number.	1312	2, 3
The referred identities are already included in your MSN CFNR rule of the entered target number.	1302	4
The MSN CFNR rules in your service data are corrupt or the alias ID is incorrect.	1602	5
A new MSN CFNRc rule has been created with the requested identities and the entered target number.	1323	6, 7
The requested identities have been added to your MSN CFNRc rule of the entered target number.	1313	8, 9
The referred identities are already included in your MSN CFNRc rule of the entered target number.	1303	10
The MSN CFNRc rules in your service data are corrupt or the alias ID is incorrect.	1603	11
A new MSN CFNL rule has been created with the requested identities and the entered target number.	1324	12, 13
The requested identities have been added to your MSN CFNL rule of the entered target number.	1314	14, 15
The referred identities are already included in your MSN CFNL rule of the entered target number.	1304	16
The MSN CFNL rules in your service data are corrupt or the alias ID is incorrect.	1604	17
A new MSN CFB rule has been created with the requested identities and the entered target number.	1325	18, 19
The requested identities have been added to your MSN CFB rule of the entered target number.	1315	20, 21
The requested identities have been added to your MSN CFB rule of the entered target number.	1305	22
The MSN CFB rules in your service data are corrupt or the alias ID is incorrect.	1605	23

Table 13 is divided into three parts, as follows:

- Use cases “0, 1, 2, 3, 6, 7, 8, 9, 12, 13, 14, 15, 18, 19, 20, 21” represent real tasks. The service data are modified according to the use case.

- Use cases “4, 10, 16, 22” represent dummy tasks. There is no need to change service data since the current value and the new value are the same. It is enough to create a proper report.
- Use cases “5, 11, 17, 23”represent provisioning error cases. The service data is invalid and must not be modified. An appropriate report is created which determines the announcement to be played.

The layout creates a negative `<report>` document for use cases “5, 11, 17 and 23”, a positive `<report>` for use cases “4, 10, 16 and 22”, and generates a `<task>` document for use cases “0, 1, 2, 3, 6, 7, 8, 9, 12, 13, 14, 15, 18, 19, 20 and 21”.

4.4.4.3.3 Announcement Configuration

The layout is constructed in a way which:

- Performs the required modifications on the service data
- Provides the necessary information for the announcement

In Example 60, the “id” attributes of the tasks and reports are given descriptive values, which enhance readability and simplify configuration of the announcement table.

```
<?xml version="1.0" encoding="utf-8"?>
 <announcements xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
   <announcement comment="The MSN CFNR rules in your service data are corrupt or the alias id
    is incorrect."media-id="1602">
     <play-condition id="msn_cfnr_provisioning_error"/>
   </announcement>
   <announcement comment="The MSN CFNRc rules in your service data are corrupt or the alias id is incorrect."
      media-id="1603">
     <play-condition id="msn_cfnrc_provisioning_error"/>
   </announcement>   <announcement comment="The MSN CFNL rules in your service data are corrupt or the
    alias id is incorrect."media-id="1604">
     <play-condition id="msn_cfnl_provisioning_error"/>
   </announcement>
   <announcement comment="The MSN CFB rules in your service data are corrupt or the alias id is incorrect."
      media-id="1605">
     <play-condition id="msn_busy_provisioning_error"/>
   </announcement>
   <announcement comment="The referred identities are already included in your MSN CFNR rule of the entered
    target number."media-id="1302">
     <play-condition id="msn_cfnr_already_activated"/>
   </announcement>
<announcement comment="The referred identities are already included in your MSN CFNRc rule of the entered
   target number."media-id="1303">
     <play-condition id="msn_cfnrc_already_activated"/>
   </announcement>
   <announcement comment="The referred identities are already included in your MSN CFNL rule of the entered
   target number."media-id="1304">
     <play-condition id="msn_cfnl_already_activated"/>
   </announcement>
   <announcement comment="The referred identities are already included in your MSN CFB rule of the entered
   target number."media-id="1305">
     <play-condition id="msn_busy_already_activated"/>
   </announcement>
   <announcement comment="Internal error occured" media-id="1800">
     <play-condition list-length="0"/>
   </announcement>
   <announcement comment="The requested identities have been added to your MSN CFNR rule of the entered
   target number."media-id="1312">
     <play-condition id="msn_cfnr_extend"/>
   </announcement>
   <announcement comment="The requested identities have been added to your MSN CFNRc rule of the entered
   target number."media-id="1313">
     <play-condition id="msn_cfnrc_extend"/>
   </announcement>
   <announcement comment="The requested identities have been added to your MSN CFNL rule of the entered
    target number."media-id="1314">
     <play-condition id="msn_cfnl_extend"/>
   </announcement>
   <announcement comment="The requested identities have been added to your MSN CFB rule of the entered
    target number."media-id="1315">
     <play-condition id="msn_busy_extend"/>
   </announcement>
   <announcement comment="A new MSN CFNR rule has been created with the requested identities and the entered
   target number."media-id="1322">
    <play-condition id="msn_cfnr_create"/>
   </announcement>   <announcement comment="A new MSN CFNRc rule has been created with the requested
     identities and the entered target number." media-id="1323">
     <play-condition id="msn_cfnrc_create"/>
   </announcement>
   <announcement comment="A new MSN CFNL rule has been created with the requested identities and the entered
      target number." media-id="1324">
     <play-condition id="msn_cfnl_create"/>
   </announcement>
   <announcement comment="A new MSN CFB rule has been created with the requested identities and the entered
      target number." media-id="1325">
     <play-condition id="msn_busy_create"/>
   </announcement>   <announcement comment="Internal error occured" media-id="1800"/>
 </announcements>
```

Example 60 Announcement Table for Setting or Updating an MSN Enhanced Conditional CDIV Rule

4.4.4.3.4 Constructing Source Document Variants

A sample CDIV rule set provisioning is shown in Example 35. It is recommended to create service data variants which represent the use cases. A few instances of the *<source>* document are constructed manually which represent combinations of the user input and the service data variants.

An instance of the possible *<source>* document variants is shown in Example 61. This instance represents the use case “21”. The comments are not generated by the GenSSC engine, they serve only for better understanding.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This document is passed to the XSL Transform function as the source XML of the transformation.
     The <service-data> element represents a provisioning of the use case 21. -->
<source version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">

  <!-- The <user-id> comes from the P-Asserted-Identity header of the SSC-INVITE. -->
  <user-id>sip:abc@ericsson.com</user-id>

  <!-- The <command-data> contains the elements of the extracted command syntax.
       It includes the schema type in this case. -->
  <command-data version="1">
    <schema>composite</schema>
  </command-data>

  <!-- The <service-data> contains a piece of the user's simservs document. The mtasGenSscCmdUtPath
       determines which elements shall be taken over from the user's service data.
       There are two <service-data> elements in this case, including the CDIV and FIP services. -->
  <service-data index="0" valid="true" version="1"
      path="simservs/communication-diversion">
    <ss:communication-diversion active="true">
      <cp:ruleset>
        <!-- The "msn-video-to-alice" rule is a combined 'served-identity' and 'media' rule. It is not a
             simple MSN enhanced conditional CDIV rule in the meaning described in Section 4.4.4.3.1,
             that is why it is out of scope of this configuration example and is filtered out. -->
        <cp:rule id="msn-video-to-alice">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111112@operator.com"/>
            </mmt-serv:served-identity>
            <ss:media>video</ss:media>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:alice@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfb-to-0046222222222" rule is an active simple MSN enhanced CFB rule. It has the
             same forward-to target as requested in the SSC command of the user story but different alias
             (1,that is the main number of the subscriber). Aliases 2 and 3 will be added to the served-identity
             condition of this rule. -->
        <cp:rule id="simple-msn-cfb-to-0046222222222">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111111@operator.com"/>
            </mmt-serv:served-identity>
            </ss:busy>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>tel:0046222222222</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfb-to-bob" rule is an active simple MSN enhanced CFB rule. It has the
             same alias as one of the aliases requested in the SSC command of the user story but different
             forward-to target. Because there are no other aliases in the served-identity condition of the
             rule,the whole rule will be deleted. -->
```

```
        <cp:rule id="simple-msn-cfb-to-bob">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111113@operator.com"/>
            </mmt-serv:served-identity>
            </ss:busy>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:bob@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
       </cp:ruleset>
    </ss:communication-diversion>
  </service-data>

  <service-data index="1" valid="true" version="1"
      path="simservs/mmt-serv:flexible-identity-presentation">
    <mmt-serv:flexible-identity-presentation active="true">
      <!-- The fip-identity element is not analyzed by the Generic SSC according to this example
           configuration. It can contain either a router address or a main number of the subscriber.
            In the latter case, the main number should also be configured in the msn-fip-identity -->
            with id="1".
       <mmt-serv:fip-identity>sip:something@operator.com</mmt-serv:fip-identity>
      <!-- Alias ID "1" in an SSC command will be replaced with alias PUI sip:+46111111111@operator.com in
           operations on the CDIV service data. This is the subscriber's main (first alias) number . -->
      <mmt-serv:msn-fip-identity id="1">
        <mmt-serv:identity>sip:+46111111111@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "2" in an SSC command will be replaced with alias PUI sip:+46111111112@operator.com in
           operations on the CDIV service data. This is the subscriber's second alias number. -->
      <mmt-serv:msn-fip-identity id="2">
        <mmt-serv:identity>sip:+46111111112@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "3" in an SSC command will be replaced with alias PUI sip:+46111111113@operator.com in
           operations on the CDIV service data. This is the subscriber's third alias number. -->
      <mmt-serv:msn-fip-identity id="3">
        <mmt-serv:identity>sip:+46111111113@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
    </mmt-serv:flexible-identity-presentation>
  </service-data>
</source>
```

Example 61 Generated <source> Document

4.4.4.3.5 Layout Design

The following internal data is defined in this example:

- The alias IDs from the SSC command (aliases)
- The target from the SSC command including the 'tel:' prefix (target)
- The condition referring to the rule type (condition)
- The report/task/new rule ID prefix (prefix)
- List of the identities (alias PUIs) addressed by the aliases (alias IDs) separated by the "|" symbol, that is, "|identity-1|identity-2|". "|" means that there are no matching identities (identities).
- List of the IDs of the simple MSN enhanced Conditional CDIV rules of a given type which include a different target and contain PUIs stored in the

‘identities’ variable, separated by the ‘‘|’’ symbol, for example, ‘‘|id-1|id-2|’’. ‘‘|’’ means that there are no matching rules (impactedRules).

- List of the IDs of the simple MSN enhanced Conditional CDIV rules of a given type which have to be updated – a subset of the affected rules to be kept which contain alias PUIs to be removed – separated by the ‘‘|’’ symbol, for example, ‘‘|id-1|id-2|’’. ‘‘|’’ means that there are no such rules (rulesToBeUpdated).

- List of the IDs of the simple MSN enhanced Conditional CDIV rules of a given type which have to be deleted – a subset of the affected rules complementary to the set of rules to be updated – separated by the ‘‘|’’ symbol, for example, ‘‘|id-1|id-2|’’. ‘‘|’’ means that there are no such rules (rulesToBeDeleted).

- ID of the target rule – either an existing one or a new one created with the generated name: *<prefix><target-without-tel-prefix>*. ‘‘|’’ means that the new rule cannot be created because a rule with the generated ID already exists. (targetRule).

- List of the identities in the target rule's served-identity condition separated by the ‘‘|’’ symbol, for example, ‘‘|identity-1|identity-2|’’. ‘‘|’’ means that there are no identities yet, that is, this is a new rule (targetRuleIdentities).

- List of the ‘missing’ identities in the target rule, that is, the identities addressed by the aliases in the SSC command, which are not yet included in the target rule's served-identity condition, separated by the ‘‘|’’ symbol, for example, ‘‘|identity-1|identity-2|’’. ‘‘|’’ means all the identities are already included in the target rule, that is, there is nothing to add (missingIdentities).

- What to do with the target rule: update an existing one, create a new one, nothing or report an error (targetRuleAction)

4.4.4.3.6 Layout Implementation

This chapter includes a concrete layout implementation of the user story in Section 4.4.4.1 Subject of the Example on page 105.

The top part of the layout includes definitions of two helper templates, see Example 62.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This layout document contains an XSL style sheet which adds the referred served identities
     to the corresponding MSN enhanced Conditional CDIV rule (CFNR, CFNRc, CFNL, CFB).
     It extends the cp:condition/mmt-serv:served-identity element if there is an existing rule having
     the same rule type and target number, otherwise a new rule is created.
     The identities are removed from the other rules of the same rule type.
     Input parameters coming from the command data:
      - aliases: includes the MSN aliases referring to the identities to be added
      - target: forward-to target number including the 'tel:' prefix or 'none'
      - condition: includes the name of the CDIV condition element (ss:busy, ss:not-reachable, ...)
      - prefix: includes a name prefix referring to the type of an MSN Conditional CDIV rule (msn_cfb_,
        msn_cfnrc_, ...)
     Service data elements used by the style sheet:
      - 0: /simservs/communication-diversion
      - 1: /simservs/flexible-identity-presentation
-->

<layout version="1"
    name="Update of the corresponding MSN enhanced Conditional CDIV rule with the referred identities"
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:gs="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
      xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
      xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
      xmlns:cp="urn:ietf:params:xml:ns:common-policy"
      exclude-result-prefixes="gs">

    <xsl:output method="xml" indent="yes"/>

    <!-- Copy the mmt-serv:one elements of a given rule not including the referred identities. -->
    <xsl:template name="copyNotReferredOnes" match="/">
        <xsl:param name="ruleName"/>
        <xsl:param name="identities"/>
      <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion/cp:ruleset
          /cp:rule[@id=$ruleName]/cp:conditions/mmt-serv:served-identity
          /mmt-serv:one[not(contains($identities, concat('|', @id, '|')))]">
          <xsl:for-each select="@*">
            <xsl:element name="mmt-serv:one">
              <xsl:attribute name="id">
                <xsl:value-of select="."/>
              </xsl:attribute>
            </xsl:element>
          </xsl:for-each>
      </xsl:for-each>
    </xsl:template>

    <!-- Create mmt-serv:one elements including the referred identities. -->
    <xsl:template name="createReferredOnes" match="/">
        <xsl:param name="aliases"/>
      <!-- Create mmt-serv:one elements including the referred MSN identities. -->
      <xsl:for-each select="/gs:source/gs:service-data[@index='1']/mmt-serv:flexible-identity-presentation/
          mmt-serv:msn-fip-identity[contains($aliases, @id)]/mmt-serv:identity">
        <mmt-serv:one>
          <xsl:attribute name="id">
            <xsl:value-of select="text()"/>
          </xsl:attribute>
        </mmt-serv:one>
      </xsl:for-each>
    </xsl:template>

    <!-- Main template. -->
    <xsl:template match="/">
```

Example 62 Helper Templates

The next part of the layout includes the definitions of the *<xsl:variable>* elements, see Example 63.

```
<!-- ***************************************
           * Variable definitions                *
           *************************************** -->
```

```
<!-- Store the alias IDs from the SSC command in a variable. -->
<xsl:variable name="aliases">
  <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='aliases']"/>
</xsl:variable>

<!-- Store the target from the SSC command including the 'tel:' prefix in a variable. -->
<xsl:variable name="target">
  <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='target']"/>
</xsl:variable>

<!-- Store the condition referring to the rule type in a variable. -->
<xsl:variable name="condition">
  <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='condition']"/>
</xsl:variable>

<!-- Store the report/task/new rule id prefix in a variable. -->
<xsl:variable name="prefix">
  <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='prefix']"/>
</xsl:variable>

<!-- Collect the identities (alias PUIs) addressed by the aliases (alias IDs) in a string and separate
     them by colon '|' symbol.
     Example: |identity-1|identity-2|
     "|" means there are no matching identities -->
<xsl:variable name="identities">
  <xsl:for-each select="/gs:source/gs:service-data[@index='1']
      /mmt-serv:flexible-identity-presentation/mmt-serv:msn-fip-identity[contains($aliases, @id)]
      /mmt-serv:identity">
    <xsl:value-of select="concat('|', text())"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Collect the ids of the simple MSN enhanced Conditional CDIV rules of a given type which include
     a different target and contain PUIs stored in the $identities. -->
<xsl:variable name="impactedRules">
  <xsl:for-each select="/gs:source/gs:service-data[@index='0']
      /ss:communication-diversion/cp:ruleset/cp:rule[
      count(cp:conditions/*)=2
      and count(cp:conditions[$condition])=1
      and (cp:actions/ss:forward-to/ss:target != $target)
      and count(cp:conditions/mmt-serv:served-identity
      /mmt-serv:one[contains($identities, concat('|', @id, '|'))]) != 0
      ]">
    <xsl:value-of select="concat('|', @id)"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Collect the ids of the simple MSN enhanced Conditional CDIV rules of a given type which have to be
     updated: a subset of impacted rules to be kept which contain PUIs to be removed. -->
<xsl:variable name="rulesToBeUpdated">
  <xsl:for-each select="/gs:source/gs:service-data[@index='0']
      /ss:communication-diversion/cp:ruleset/cp:rule[contains($impactedRules, concat('|', @id, '|'))
      and count(cp:conditions/mmt-serv:served-identity
      /mmt-serv:one[not(contains($identities, concat('|', @id, '|')))]) != 0
      ]">
    <xsl:value-of select="concat('|', @id)"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Collect the ids of the simple MSN enhanced Conditional CDIV rules of a given type which have to be
     deleted: a subset of the impacted rules complementary to the set of rules to be updated. -->
<xsl:variable name="rulesToBeDeleted">
  <xsl:for-each select="/gs:source/gs:service-data[@index='0']
      /ss:communication-diversion/cp:ruleset/cp:rule[
      contains($impactedRules, concat('|', @id, '|'))
      and not(contains($rulesToBeUpdated, concat('|', @id, '|')))
      ]">
    <xsl:value-of select="concat('|', @id)"/>
  </xsl:for-each>
  <xsl:value-of select="'|'"/>
</xsl:variable>

<!-- Find out the name (id) of the target rule. -->
```

```
    <xsl:variable name="targetRule">
      <xsl:choose>
        <!-- Try to find an existing rule. -->
        <xsl:when test="count(/gs:source/gs:service-data[@index='0']
            /ss:communication-diversion/cp:ruleset/cp:rule[
            count(cp:conditions/*)=2
            and count(cp:conditions[$condition])=1
            and count(cp:conditions/mmt-serv:served-identity)=1
            and (cp:actions/ss:forward-to/ss:target = $target)
            ]) != 0">
          <xsl:value-of select="/gs:source/gs:service-data[@index='0']
              /ss:communication-diversion/cp:ruleset/cp:rule[
              count(cp:conditions/*)=2
              and count(cp:conditions[$condition])=1
              and count(cp:conditions/mmt-serv:served-identity)=1
              and (cp:actions/ss:forward-to/ss:target = $target)
              ]/@id"/>
        </xsl:when>
        <!-- New rule shall be created with the generated name: <rule-prefix><target-without-tel-prefix>.
             Check if there is an existing rule with the same name. -->
        <xsl:when test="count(/gs:source/gs:service-data[@index='0']
            /ss:communication-diversion/cp:ruleset/cp:rule[
            @id=concat($prefix, substring($target,5))
            ]) = 0">
          <xsl:value-of select="concat($prefix, substring($target,5))"/>
        </xsl:when>
        <!-- Indicate the error case: the new rule cannot be created since the generated rule name is
             reserved. -->
        <xsl:otherwise><xsl:value-of select="'|'"/></xsl:otherwise>
      </xsl:choose>
    </xsl:variable>

    <!-- Collect the identities in the target rule's served-identity condition in a string.
         "|" means there are no identities yet, that is this will be a new rule -->
    <xsl:variable name="targetRuleIdentities">
      <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion/cp:ruleset
          /cp:rule[@id=$targetRule]/cp:conditions/mmt-serv:served-identity/mmt-serv:one">
        <xsl:value-of select="concat('|', @id)"/>
      </xsl:for-each>
      <xsl:value-of select="'|'"/>
    </xsl:variable>

    <!-- Collect the 'missing' identities in the target rule, that is the identities addressed by the⇒
aliases in
         the SSC command, which are not yet included in the target rule's served-identity condition.
         "|" means all the identities are already included in in the target rule, that is there is⇒
nothing to add
    -->
    <xsl:variable name="missingIdentities">
      <xsl:for-each select="/gs:source/gs:service-data[@index='1']
          /mmt-serv:flexible-identity-presentation/mmt-serv:msn-fip-identity[contains($aliases, @id)]
          /mmt-serv:identity">
        <xsl:if test="not(contains($targetRuleIdentities, concat('|', text(), '|')))">
          <xsl:value-of select="concat('|', text())"/>
        </xsl:if>
      </xsl:for-each>
      <xsl:value-of select="'|'"/>
    </xsl:variable>

    <!-- Find out what to do with the target rule:
         update:  update an existing rule
         create:  create a new rule
         nothing: there is nothing to do since all the referred
                  identities are already included in its condition
         error:   error report shall be generated -->
    <xsl:variable name="targetRuleAction">
      <xsl:choose>
        <!-- Check if the target rule is valid. -->
        <xsl:when test="$targetRule = '|'">
          <xsl:value-of select="'error'"/>
        </xsl:when>
        <!-- Check if the target rule exists.
             If not then a new rule shall be created. -->
        <xsl:when test="count(/gs:source/gs:service-data[@index='0']
            /ss:communication-diversion/cp:ruleset/cp:rule[
            @id=$targetRule]) = 0">
```

```
          <xsl:value-of select="'create'"/>
        </xsl:when>
        <!-- Check if there is any missing identity. -->
        <xsl:when test="$missingIdentities = '|'">
          <xsl:value-of select="'nothing'"/>
        </xsl:when>
        <!-- There is at least one missing identity: update. -->
        <xsl:otherwise>
          <xsl:value-of select="'update'"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
```

Example 63 Variable Definitions

The bottom part of the layout document comprises the generation of the output. It is divided into several sections. They refer to the use cases of Table 12.

The first section refers to the use cases “5, 11, 17, 23” of Table 12, see Example 64.

```
    <!-- ***************************************
         * Output generation                   *
         *************************************** -->

    <xsl:choose>
      <xsl:when test="($targetRuleAction='error') or ($identities='|')">
        <!-- The new rule cannot be created since there is an existing rule
with the same nameor the alias IDs in the SSC command are not mapped to
subscriber's PUIs in the FIP service data.
             It is a provisioning error. -->
        <report version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
          <xsl:attribute name="id">
            <xsl:value-of select="concat($prefix, 'provisioning_error')"/>
          </xsl:attribute>
          <response>500 Internal Error</response>
          <job>put</job>
          <value type="empty"/>
        </report>
      </xsl:when>
```

Example 64 Treatment of Provisioning Defects – Negative Report is Generated

Use cases “4, 10, 16, 22” show dummy tasks and are shown in Example 65.

```
      <!-- There is nothing to do: calls to the referred identities are already redirected
to the target number and there are no other simple MSN enhanced Conditional CDIV rules of
a given type including these identities in the served-identity condition. -->
      <xsl:when test="($targetRuleAction='nothing') and ($impactedRules='|')">
        <report version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
          <xsl:attribute name="id">
            <xsl:value-of select="concat($prefix, 'already_activated')"/>
          </xsl:attribute>
          <response>200 OK</response>
          <job>put</job>
          <value type="empty"/>
        </report>
      </xsl:when>
```

Example 65 Dummy Task Use Cases – Positive Report is Generated

The ‘service data update’ section creates *<task>* documents to modify the service data according to the user input. Once the *<task>* has been generated, the further processing is the same as described in Section 4.1.3 Workflow Overview on page 36. An appropriate XCAP request is sent to the XCAP

servlet which applies the service-specific application rules on the request and updates the HSS with the modified service data. The GenSSC engine creates a *<report>* from the XCAP response (the report ID is inherited from the task ID). The report is compared with the announcement table and the matching voice message is sent to the user. The tasks of the service data update are shown in Example 66.

```
<!-- The service data needs to be modified. -->
        <xsl:otherwise>
          <task-list version="1"
              xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
              xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
              xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
              xmlns:cp="urn:ietf:params:xml:ns:common-policy">
            <!-- Remove the rules to be deleted. -->
            <xsl:if test="$rulesToBeDeleted != '|'">
              <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion⇒
/cp:ruleset
                  /cp:rule[contains($rulesToBeDeleted, concat('|', @id, '|'))]">
                <task version="1">
                  <xsl:attribute name="id">
                    <xsl:value-of select="concat('delete-', @id)"/>
                  </xsl:attribute>
                  <head>
                    <action>delete</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='@id'/>"]
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                    </xsl:element>
                  </head>
                </task>
              </xsl:for-each>
            </xsl:if>
            <!-- Remove the referred identities from the rules to be updated. -->
            <xsl:if test="$rulesToBeUpdated != '|'">
              <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion⇒
/cp:ruleset
                  /cp:rule[contains($rulesToBeUpdated, concat('|', @id, '|'))]">
                <task version="1">
                  <xsl:attribute name="id">
                    <xsl:value-of select="concat('update-', @id)"/>
                  </xsl:attribute>
                  <head>
                    <action>set</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='@id'/>"]
                        /cp:conditions/mmt-serv:served-identity
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                        xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
                    </xsl:element>
                  </head>
                  <body>
                    <mmt-serv:served-identity>
                      <!-- Copy the mmt-serv:one elements not including the referred identities. -->
                      <xsl:call-template name="copyNotReferredOnes">
                         <xsl:with-param name="ruleName" select="@id"/>
                         <xsl:with-param name="identities" select="$identities"/>
                      </xsl:call-template>
                    </mmt-serv:served-identity>
                  </body>
                </task>
              </xsl:for-each>
            </xsl:if>
            <!-- Handle the target rule. -->
            <xsl:choose>
              <!-- Extend the target rule with the missing identities. -->
              <xsl:when test="$targetRuleAction = 'update'">
                <task version="1">
                  <xsl:attribute name="id">
                    <xsl:value-of select="concat($prefix, 'extend')"/>
                  </xsl:attribute>
                  <head>
```

```
                    <action>set</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='$targetRule'/>"]
                        /cp:conditions/mmt-serv:served-identity
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                        xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
                    </xsl:element>
                  </head>
                  <body>
                    <mmt-serv:served-identity>
                      <!-- Copy the mmt-serv:one elements not including the referred identities. -->
                      <xsl:call-template name="copyNotReferredOnes">
                         <xsl:with-param name="ruleName" select="$targetRule"/>
                         <xsl:with-param name="identities" select="$identities"/>
                      </xsl:call-template>
                      <!-- Create mmt-serv:one elements including the referred identities. -->
                      <xsl:call-template name="createReferredOnes">
                         <xsl:with-param name="aliases" select="$aliases"/>
                      </xsl:call-template>
                    </mmt-serv:served-identity>
                  </body>
                </task>
              </xsl:when>
              <!-- Create a new target rule including the referred identities. -->
              <xsl:when test="$targetRuleAction='create'">
                <task version="1">
                  <xsl:attribute name="id">
                    <xsl:value-of select="concat($prefix, 'create')"/>
                  </xsl:attribute>
                  <head>
                    <action>set</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='$targetRule'/>"]
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                    </xsl:element>
                  </head>
                  <body>
                    <cp:rule>
                      <xsl:attribute name="id">
                        <xsl:value-of select='$targetRule'/>
                      </xsl:attribute>
                      <cp:conditions>
                        <xsl:element name="{$condition}"/>
                        <mmt-serv:served-identity>
                          <!-- Create mmt-serv:one elements including the referred identities. -->
                          <xsl:call-template name="createReferredOnes">
                             <xsl:with-param name="aliases" select="$aliases"/>
                          </xsl:call-template>
                        </mmt-serv:served-identity>
                      </cp:conditions>
                      <cp:actions>
                          <ss:forward-to>
                            <ss:target><xsl:value-of select="$target"/></ss:target>
                            <ss:notify-caller>true</ss:notify-caller>
                          </ss:forward-to>
                      </cp:actions>
                    </cp:rule>
                  </body>
                </task>
              </xsl:when>
            </xsl:choose>
          </task-list>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:template>
  </xsl:stylesheet>
</layout>
```

Example 66 Service Data Update – Tasks are Generated

Reviewing the source document once again, it turns out that the current provisioning and the user input result in two tasks: the simple-msn-cfb-to-bob

rule is deleted and the simple-msn-cfb-to-0046222222222 is updated by adding aliases 2 and 3 to the served-identity condition. Both of the generated tasks are shown in Example 67.

```
<?xml version="1.0"?>
<task-list version="1"
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy">

  <task  version="1" id="delete-simple-msn-cfb-to-bob">
    <head>
      <action>delete</action>
      <content-type>application/xcap-el+xml</content-type>
      <ut-path>
        simservs/communication-diversion/cp:ruleset
            /cp:rule[@id="simple-msn-cfb-to-bob"]?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
      </ut-path>
    </head>
  </task>

  <task  version="1" id="msn_busy_extend">
    <head>
      <action>set</action>
      <content-type>application/xcap-el+xml</content-type>
      <ut-path>
        simservs/communication-diversion/cp:ruleset
            /cp:rule[@id="simple-msn-cfb-to-0046222222222"]/cp:conditions
            /mmt-serv:served-identity?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
            xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
      </ut-path>
    </head>
    <body>
      <mmt-serv:served-identity>
        <mmt-serv:one id="sip:+46111111111@operator.com"/>
        <mmt-serv:one id="sip:+46111111112@operator.com"/>
        <mmt-serv:one id="sip:+46111111113@operator.com"/>
      </mmt-serv:served-identity>
    </body>
  </task>

</task-list>
```

Example 67 Result of the XSLT Transformation – A Task List with Two Tasks

The announcement played to the user is: `The requested identities have been added to your MSN CFB rule of the entered target number.` (media-id=”1315”)

4.4.5 Support for MSN Conditional CDIV Delete Command

This section contains an example for a complex composite configuration which analyzes the service data in the first step. According to the actual content, either it updates the service data or keeps it unchanged and plays an appropriate announcement to the user at the end. The example deletes the referred alias identities from the ‘served-identity’ condition of a simple MSN enhanced Conditional CDIV rule. If the referred aliases are the only identities in the ‘served-identity’ condition, the whole rule is removed. These rules are members of the CDIV rule set, see Example 35.

4.4.5.1 Subject of the Example

An IMS user wants to delete one or more simple MSN enhanced Conditional CDIV rules of a given type using the SSC command format: *11*<aliases>##<service_code># so that incoming calls to the first, second, third (123), or both alias PUIs are not conditionally (67) diverted to any target.

Example for a related SSC command:

*11*23##67#

The user wants to be informed of the following:

- If the referred identities have been removed from the MSN enhanced Conditional CDIV rules of a given type
- If whole MSN enhanced Conditional CDIV rules of a given type containing the referred identities have been removed
- If there are no active MSN enhanced Conditional CDIV rules of a given type including the referred identities
- If there are not any active MSN enhanced Conditional CDIV rules of a given type
- If an error occurred

4.4.5.2 Configuration Overview

This section shows an example configuration of an MtasGenSscCmd MO instance which is in line with the above description.

MtasGenSscCmd: MsnCDivCondDel (key). The “MsnCDivCondDel”string identifies this MtasGenSscCmd MO instance.

mtasGenSscCmdHeader:

*11*1##6|*11*2##6|*11*3##6|*11*12##6|*11*13##6|*11*23##6|*11*123##6

The command header attribute lists the variations of leading digits related to this configuration. They differ in the alias IDs in the middle.

The command syntax chosen in this example allows for using of three aliases at most, owing to the decoding tree limitations. Using more aliases would be possible with a different SSC syntax, for example, #11<service_code>*<aliases>#. In that case, the command header attribute would have the value of #11<first_digit_of_the_service_code>, for example, #116.

mtasGenSscCmdSyntax:

```
/\*11\*([1-3]{1,3})##61#$/schema=composite; digits aliases=\1;
item condition=ss:no-answer; item prefix=msn_cfnr_/~~
/^\*11\*([1-3]{1,3})##62#$/schema=composite; digits aliases=\1;
item condition=ss:not-reachable; item prefix=msn_cfnrc_/~~
/^\*11\*([1-3]{1,3})##63#$/schema=composite; digits aliases=\1;
item condition=ss:not-registered; item prefix=msn_cfnl_/~~
/^\*11\*([1-3]{1,3})##67#$/schema=composite; digits aliases=\1;
item condition=ss:busy; item prefix=msn_busy_/
```

The `mtasGenSscCmdSyntax` consists of four parts, one for each kind of conditional CDIV. Every part contains two subparts: a Command Pattern and a Command Syntax.

The Command Pattern, for example, `^\*11\*([1-3]{1,3})##67#$`, declares the format of the Request URI.

The Command Syntax part, for example, `schema=composite; digits aliases=\1; item condition=ss:busy; item prefix=msn_busy_`, informs the GenSSC engine that a composite configuration is processed. It means that the task or report is extracted from the XSLT style sheet included in the layout.xml document of the `mtasGenSscCmdLayout` attribute. It also declares three variables to be fetched from an SSC command: aliases of digits type to be taken from the first variable part of the command, condition of item type which contains an appropriate CDIV condition, and prefix of item type which contains an appropriate prefix to be used as a part of a report or task ID.

`mtasGenSscCmdUtPath`:

```
simservs/communication-diversion~~simservs/mmt-serv:⇒
flexible-identity-presentation
```

The `UtPath` points to the CDIV and FIP services, they are copied from the *<service-data>* to the *<source>* document.

`mtasGenSscCmdAnnouncements`:

The announcement table is described in Section 4.4.2.3.3 Announcement Configuration on page 89.

`mtasGenSscCmdLayout`:

The layout document is described in Section 4.4.2.3.6 Layout Implementation on page 91. Other attributes: the attributes `mtasGenSscCmdReplace`, `mtasGenSscCmdContentType`, and `mtasGenSscCmdUriInChargingMessages` of the `MtasGenSscCmd` MO are not used in this configuration, the default values are assigned to them.

4.4.5.3 Workflow Overview

This section shows the workflow overview.

4.4.5.3.1 Understanding the Details

The details of the Delete command include similar difficulties as are described by the Set command, see Section 4.4.4.3.1 Understanding the Details on page 107.

4.4.5.3.2 Clarification of the Use Cases

The user's service profile includes the following information:

- If any simple MSN enhanced Conditional CDIV rules of a given type exist in the CDIV rule set
- If any simple MSN enhanced Conditional CDIV rules of a given type for alias IDs given in the SSC command exist in the CDIV rule set
- If the existing simple MSN enhanced Conditional CDIV rules of a given type for the alias IDs given in the SSC command contain also other alias identities in the 'served-identity' condition or the referred ones only

The combinations of these CFU rule information categories define the use cases of the user story. They are the input parameters which help to determine the following:

- The aliases from the existing rules to be deleted
- The whole rules to be deleted
- The announcement to be played to the user

That is, not only the service data must be modified but the user must be provided with a meaningful announcement as well.

Table 14 summarizes the use cases to be considered and the announcements to be played for each use case.

Table 14 Use Cases of SSC Command for Deleting MSN Enhanced Conditional CDIV Rules

Use Case	Type of Conditional CDIV	Simple Rules Exist	Simple Rules for Given Aliases Exist	A Simple Rule for Given Aliases Contain Also Other Aliases	To Be Deleted	Announcement
UC-0	CFNR	Yes	Yes	Yes	Aliases from the existing rules and optionally whole rules	The referred identities have been removed from the MSN CFNR rules. There are still active MSN CFNR rules.
UC-1	CFNR	Yes	Yes	No	Whole rules	Whole MSN CFNR rules have been removed.
UC-2	CFNR	Yes	No	No	Nothing	You have no active MSN CFNR rules.
UC-3	CFNR	No	No	No	Nothing	You have no active MSN CFNR rules.

Use Case	Type of Conditional CDIV	Simple Rules Exist	Simple Rules for Given Aliases Exist	A Simple Rule for Given Aliases Contain Also Other Aliases	To Be Deleted	Announcement
UC-4	CFNRc	Yes	Yes	Yes	Aliases from the existing rules and optionally whole rules	The referred identities have been removed from the MSN CFNRc rules. You still have active MSN CFNRc rules.
UC-5	CFNRc	Yes	Yes	No	Whole rules	Whole MSN CFNRc rules have been removed.
UC-6	CFNRc	Yes	No	No	Nothing	There are no active MSN CFNRc rules including the referred identities.
UC-7	CFNRc	No	No	No	Nothing	You have no active MSN CFNRc rules.

Use Case	Type of Conditional CDIV	Simple Rules Exist	Simple Rules for Given Aliases Exist	A Simple Rule for Given Aliases Contain Also Other Aliases	To Be Deleted	Announcement
UC-8	CFNL	Yes	Yes	Yes	Aliases from the existing rules and optionally whole rules	The referred identities have been removed from the MSN CFNL rules. You still have active MSN CFNL rules.
UC-9	CFNL	Yes	Yes	No	Whole rules	Whole MSN CFNL rules have been removed.
UC-10	CFNL	Yes	No	No	Nothing	There are no active MSN CFNL rules including the referred identities.
UC-11	CFNL	No	No	No	Nothing	You have no active MSN CFNL rules.

Use Case	Type of Conditional CDIV	Simple Rules Exist	Simple Rules for Given Aliases Exist	A Simple Rule for Given Aliases Contain Also Other Aliases	To Be Deleted	Announcement
UC-12	CFB	Yes	Yes	Yes	Aliases from the existing rules and optionally whole rules	The referred identities have been removed from the MSN CFB rules. You still have active MSN CFB rules.
UC-13	CFB	Yes	Yes	No	Whole rules	Whole MSN CFB rules have been removed.
UC-14	CFB	Yes	No	No	Nothing	There are no active MSN CFB rules including the referred identities.
UC-15	CFB	No	No	No	Nothing	You have no active MSN CFB rules.

A summary of the announcement variations is shown in Table 15.

Table 15 Summary of Announcement Variations - Deleting MSN Enhanced Conditional CDIV Rules

Announcement	Media ID	Use Cases
The referred identities have been removed from the MSN CFNR rules. You still have active MSN CFNR rules.	1163	0
Whole MSN CFNR rules have been removed.	1164	1
There are no active MSN CFNR rules including the referred identities.	1173	2
You have no active MSN CFNR rules.	1174	3
The referred identities have been removed from the MSN CFNRc rules. You still have active MSN CFNRc rules.	1165	4
Whole MSN CFNRc rules have been removed.	1166	5
There are no active MSN CFNRc rules including the referred identities.	1175	6
You have no active MSN CFNRc rules.	1176	7
The referred identities have been removed from the MSN CFNL rules. You still have active MSN CFNL rules.	1167	8
Whole MSN CFNL rules have been removed.	1168	9
There are no active MSN CFNL rules including the referred identities.	1177	10
You have no active MSN CFNL rules.	1178	11
The referred identities have been removed from the MSN CFB rules. You still have active MSN CFB rules.	1161	12
Whole MSN CFB rules have been removed.	1162	13
There are no active MSN CFB rules including the referred identities.	1171	14
You have no active MSN CFB rules.	1172	15

Table 15 is divided into two parts, as follows:

- Use cases “0, 1, 4, 5, 8, 9, 12, 13” represent real tasks. The service data is modified according to the use case.
- Use cases “2, 3, 6, 7, 10, 11, 14, 15” represent dummy tasks. There is no need to change service data since the current value and the new value are the same. It is enough to create a proper report.

The layout creates a positive `<report>` for the use cases “2, 3, 6, 7, 10, 11, 14” and “15” and generates a `<task>` document for the use cases “0, 1, 4, 5, 8, 9, 12” and “13”.

4.4.5.3.3 Announcement Configuration

The layout is constructed in a way which:

- Performs the required modifications on the service data

AND

- Provides the necessary information for the announcement

In this example, the “id” attributes of the tasks and reports are given descriptive values, which enhance readability and simplify configuration of the announcement table, see Example 68.

```
<?xml version="1.0" encoding="utf-8"?> <announcements xmlns="http://uri.etsi.org/ngn/params/xml/simservs
    /xcap/genssc">
   <announcement comment="There are no active MSN CFNR rules including the referred identities.
    " media-id="1173">
     <play-condition id="msn_cfnr_already_deactivated"/>
   </announcement>   <announcement comment="You have no active MSN CFNR rules." media-id="1174">
       <play-condition id="msn_cfnr_no_active_rule"/>
   </announcement>   <announcement comment="There are no active MSN CFNRc rules including the referred⇒
 identities."
      media-id="1175">
     <play-condition id="msn_cfnrc_already_deactivated"/>
   </announcement>
   <announcement comment="You have no active MSN CFNRc rules." media-id="1176">
     <play-condition id="msn_cfnrc_no_active_rule"/>
   </announcement>
   <announcement comment="There are no active MSN CFNL rules including the referred identities.
   "media-id="1177">
     <play-condition id="msn_cfnl_already_deactivated"/>
   </announcement>
   <announcement comment="You have no active MSN CFNL rules." media-id="1178">
     <play-condition id="msn_cfnl_no_active_rule"/>
   </announcement>
   <announcement comment="There are no active MSN CFB rules including the referred identities.
   "media-id="1171">
     <play-condition id="msn_busy_already_deactivated"/>
   </announcement>   <announcement comment="You have no active MSN CFB rules." media-id="1172">
     <play-condition id="msn_busy_no_active_rule"/>
   </announcement>
   <announcement comment="Internal error occured" media-id="1800">
     <play-condition list-length="0"/>
   </announcement>
   <announcement comment="The referred identities have been removed from the MSN CFNR rules.
    You still have active MSN CFNR rules." media-id="1163">
     <play-condition id="msn_cfnr_update"/>
   </announcement>
   <announcement comment="Whole MSN CFNR rules have been removed." media-id="1164">
    <play-condition id="msn_cfnr_delete"/>
   </announcement>
   <announcement comment="The referred identities have been removed from the MSN CFNRc rules.
    You still have active MSN CFNRc rules." media-id="1165">
     <play-condition id="msn_cfnrc_update"/>
   </announcement>
   <announcement comment="Whole MSN CFNRc rules have been removed." media-id="1166">
    <play-condition id="msn_cfnrc_delete"/>
   </announcement>   <announcement comment="The referred identities have been removed from the⇒
MSN CFNL rules.
     You still have active MSN CFNL rules." media-id="1167">
     <play-condition id="msn_cfnl_update"/>
   </announcement>   <announcement comment="Whole MSN CFNL rules have been removed.
    "media-id="1168"> <play-condition id="msn_cfnl_delete"/>
   </announcement>
   <announcement comment="The referred identities have been removed from the MSN CFB rules.
    You still have active MSN CFB rules." media-id="1161">
     <play-condition id="msn_busy_update"/>
   </announcement>
   <announcement comment="Whole MSN CFB rules have been removed." media-id="1162">
     <play-condition id="msn_busy_delete"/>
   </announcement>   <announcement comment="Internal error occured" media-id="1800"/>
 </announcements>
```

Example 68 Announcement Table for Deleting MSN Enhanced Conditional CDIV Rules

4.4.5.3.4 Constructing Source Document Variants

An instance of the possible <source> document variants is shown in Example 69. This instance represents the use case ‘‘12’’. The comments are not generated by the GenSSC engine, they serve only for better understanding.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This document is passed to the XSL Transform function as the source XML of the transformation.
     The <service-data> element represents a provisioning of the use case 12. -->
```

```
<source version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">

  <!-- The <user-id> comes from the P-Asserted-Identity header of the SSC-INVITE. -->
  <user-id>sip:abc@ericsson.com</user-id>

  <!-- The <command-data> contains the elements of the extracted command syntax.
       It includes the schema type in this case. -->
  <command-data version="1">
    <schema>composite</schema>
  </command-data>

  <!-- The <service-data> contains a piece of the user's simservs document. The mtasGenSscCmdUtPath
  determines which elements shall be taken over from the user's service data.
       There are two <service-data> elements in this case, including the CDIV and FIP services. -->
  <service-data index="0" valid="true" version="1"
      path="simservs/communication-diversion">
    <ss:communication-diversion active="true">
      <cp:ruleset>
        <!-- The "msn-video-to-alice" rule is a combined 'served-identity' and 'media' rule. It is not a
             simple MSN enhanced conditional CDIV rule in the meaning described in Section 4.4.2.1.3.1,
        that is why it is out of scope of this configuration example and is filtered out. -->
        <cp:rule id="msn-video-to-alice">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111112@operator.com"/>
            </mmt-serv:served-identity>
            <ss:media>video</ss:media>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:alice@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfb-to-0046222222222" rule is an active simple MSN enhanced CFB rule.
             Alias 2 will be removed from the served-identity condition of this rule. -->
        <cp:rule id="simple-msn-cfb-to-0046222222222">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111111@operator.com"/>
              <mmt-serv:one id="sip:+46111111112@operator.com"/>
            </mmt-serv:served-identity>
            </ss:busy>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>tel:0046222222222</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfb-to-bob" rule is an active simple MSN enhanced CFB rule. Because there are
             no other aliases in the served-identity condition of this rule, the whole rule will be deleted. -->
        <cp:rule id="simple-msn-cfb-to-bob">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111113@operator.com"/>
            </mmt-serv:served-identity>
            </ss:busy>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:bob@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
       </cp:ruleset>
    </ss:communication-diversion>
  </service-data>

  <service-data index="1" valid="true" version="1"
      path="simservs/mmt-serv:flexible-identity-presentation">
    <mmt-serv:flexible-identity-presentation active="true">
      <!-- The fip-identity element is not analyzed by the Generic SSC according to this example
```

```
          configuration. It can contain either a router address or a main number of the subscriber.
          In the latter case, the main number should also be configured in the msn-fip-identity
           with id="1". -->
      <mmt-serv:fip-identity>sip:something@operator.com</mmt-serv:fip-identity>
      <!-- Alias ID "1" in an SSC command will be replaced with alias PUI sip:+46111111111@operator.com in
          operations on the CDIV service data. This is the subscriber's main (first alias) number . -->
      <mmt-serv:msn-fip-identity id="1">
        <mmt-serv:identity>sip:+46111111111@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "2" in an SSC command will be replaced with alias PUI sip:+46111111112@operator.com in
          operations on the CDIV service data. This is the subscriber's second alias number. -->
      <mmt-serv:msn-fip-identity id="2">
        <mmt-serv:identity>sip:+46111111112@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "3" in an SSC command will be replaced with alias PUI sip:+46111111113@operator.com in
          operations on the CDIV service data. This is the subscriber's third alias number. -->
      <mmt-serv:msn-fip-identity id="3">
        <mmt-serv:identity>sip:+46111111113@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
    </mmt-serv:flexible-identity-presentation>
  </service-data>
</source>
```

Example 69 Generated <source> Document

4.4.5.3.5 Layout Design

The following internal data is defined in this example:

- The alias IDs from the SSC command (aliases)
- The condition referring to the rule type (condition)
- The report/task ID prefix (prefix)
- List of the identities (alias PUIs) addressed by the aliases (alias IDs) separated by the ‘‘|’’ symbol, that is, "|identity-1|identity-2|". ‘‘|’’ means that there are no matching identities (identities).
- List of the IDs of the simple MSN enhanced Conditional CDIV rules of a given type which contain PUIs stored in the ‘identities’ variable, separated by the ‘‘ |’’ symbol, for example, ‘‘|id-1|id-2|’’. ‘‘|’’ means that there are no matching rules (impactedRules).
- List of the IDs of the simple MSN enhanced Conditional CDIV rules of a given type which have to be updated, a subset of the affected rules to be kept which contain alias PUIs to be removed, separated by the ‘‘ |’’ symbol, for example, ‘‘|id-1|id-2|’’. ‘‘|’’ means that there are no such rules (rulesToBeUpdated).
- List of the IDs of the simple MSN enhanced Conditional CDIV rules of a given type which have to be deleted – a subset of the affected rules complementary to the set of rules to be updated – separated by the ‘‘ | ’’ symbol, for example, ‘‘|id-1|id-2|’’. ‘‘|’’ means that there are no such rules (rulesToBeDeleted).

4.4.5.3.6 Layout Implementation

This section includes a concrete layout implementation of the user story in Section 4.4.2.1 Subject of the Example on page 85.

The top part of the layout includes a definition of one helper template, see Example 70.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This layout document contains an XSL style sheet which removes the referred served identities
     from the corresponding MSN enhanced Conditional CDIV rules (CFNR, CFNRc, CFNL, CFB).
     Input parameters coming from the command data:
      - aliases: includes the MSN aliases referring to the identities to be removed
      - condition: includes the name of the CDIV condition element (ss:busy, ss:not-reachable, ...)
      - prefix: includes a name prefix referring to the type of an MSN Conditional CDIV rule (msn_cfb_,
        msn_cfnrc_, ...)
     Service data elements used by the style sheet:
      - 0: /simservs/communication-diversion
      - 1: /simservs/flexible-identity-presentation
-->
<layout version="1"
    name="Removal of the referred served identities from the corresponding MSN Conditional CDIV rules"
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:gs="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
      xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
      xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
      xmlns:cp="urn:ietf:params:xml:ns:common-policy"
      exclude-result-prefixes="gs">

    <xsl:output method="xml" indent="yes"/>

    <!-- Copy the mmt-serv:one elements of a given rule not including the referred identities. -->
    <xsl:template name="copyNotReferredOnes"
        match="/">
        <xsl:param name="ruleName"/>
        <xsl:param name="identities"/>
        <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion/cp:ruleset
          /cp:rule[@id=$ruleName]/cp:conditions/mmt-serv:served-identity
          /mmt-serv:one[not(contains($identities, concat('|', @id, '|')))]">
          <xsl:for-each select="@*">
            <xsl:element name="mmt-serv:one">
              <xsl:attribute name="id">
                <xsl:value-of select="."/>
              </xsl:attribute>
            </xsl:element>
          </xsl:for-each>
        </xsl:for-each>
    </xsl:template>

    <!-- Main template. -->
    <xsl:template match="/">
```

Example 70 Helper Template

The next part of the layout includes the definitions of the *<xsl:variable>* elements, see Example 71.

```
<!-- ****************************************
           * Variable definitions                *
           **************************************** -->

      <!-- Store the alias IDs from the SSC command in a variable. -->
      <xsl:variable name="aliases">
        <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='aliases']"/>
      </xsl:variable>

      <!-- Store the condition referring to the rule type in a variable. -->
      <xsl:variable name="condition">
```

```
      <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='condition']"/>
    </xsl:variable>

    <!-- Store the report/task id prefix in a variable. -->
    <xsl:variable name="prefix">
      <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='prefix']"/>
    </xsl:variable>

    <!-- Collect the identities (alias PUIs) addressed by the aliases (alias IDs) in a string and separate
         them by colon '|' symbol.
         Example: |identity-1|identity-2|
         "|" means there are no matching identities -->
    <xsl:variable name="identities">
      <xsl:for-each select="/gs:source/gs:service-data[@index='1']
          /mmt-serv:flexible-identity-presentation/mmt-serv:msn-fip-identity[contains($aliases, @id)]
          /mmt-serv:identity">
        <xsl:value-of select="concat('|', text())"/>
      </xsl:for-each>
      <xsl:value-of select="'|'"/>
    </xsl:variable>

    <!-- Collect the ids of the simple MSN enhanced Conditional CDIV rules of a given type which contain⇒
PUIs
         stored in the $identities.-->
    <xsl:variable name="impactedRules">
      <xsl:for-each select="/gs:source/gs:service-data[@index='0']
          /ss:communication-diversion/cp:ruleset/cp:rule[
          count(cp:conditions/*)=2
          and count(cp:conditions[$condition])=1
          and count(cp:conditions/mmt-serv:served-identity/mmt-serv:one[
          contains($identities, concat('|', @id, '|'))]) != 0
          ]">
        <xsl:value-of select="concat('|', @id)"/>
      </xsl:for-each>
      <xsl:value-of select="'|'"/>
    </xsl:variable>

    <!-- Collect the ids of the simple MSN enhanced Conditional CDIV rules of a given type which have to be
         updated: a subset of impacted rules to be kept which contain PUIs to be removed. -->
    <xsl:variable name="rulesToBeUpdated">
      <xsl:for-each select="/gs:source/gs:service-data[@index='0']
          /ss:communication-diversion/cp:ruleset/cp:rule[contains($impactedRules, concat('|', @id, '|'))
          and count(cp:conditions/mmt-serv:served-identity
          /mmt-serv:one[not(contains($identities, concat('|', @id, '|')))]) != 0
          ]">
        <xsl:value-of select="concat('|', @id)"/>
      </xsl:for-each>
      <xsl:value-of select="'|'"/>
    </xsl:variable>

    <!-- Collect the ids of the simple MSN enhanced Conditional CDIV rules of a given type which have to be
         deleted: a subset of the impacted rules complementary to the set of rules to be updated. -->
    <xsl:variable name="rulesToBeDeleted">
      <xsl:for-each select="/gs:source/gs:service-data[@index='0']
          /ss:communication-diversion/cp:ruleset/cp:rule[
          contains($impactedRules, concat('|', @id, '|'))
          and not(contains($rulesToBeUpdated, concat('|', @id, '|')))
          ]">
        <xsl:value-of select="concat('|', @id)"/>
      </xsl:for-each>
      <xsl:value-of select="'|'"/>
    </xsl:variable>
```

Example 71 Variable Definitions

The bottom part of the layout document comprises the generation of the output. It is divided into several sections. They refer to the use cases of Table 14. The first section refers to the use cases “2, 3, 6, 7, 10, 11, 14, 15” of Table 14, see Example 72.

```
<!-- ****************************************
     * Output generation                    *
     **************************************** -->

<xsl:choose>
  <!-- There is nothing to do: the referred identities are not included in any MSN Conditional
       CDIV rule of a given type. -->
  <xsl:when test="$impactedRules='|'">
    <report version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
      <xsl:attribute name="id">
        <xsl:choose>
          <xsl:when test="count(/gs:source/gs:service-data[@index='0']
              /ss:communication-diversion/cp:ruleset/cp:rule[
              count(cp:conditions/*)=2
              and count(cp:conditions[$condition and mmt-serv:served-identity])=1
              ]) != 0">
            <xsl:value-of select="concat($prefix, 'already_deactivated')"/>
          </xsl:when>
          <xsl:otherwise><xsl:value-of select="concat($prefix, 'no_active_rule')"/></xsl:otherwise>
        </xsl:choose>
      </xsl:attribute>
      <response>200 OK</response>
      <job>delete</job>
      <value type="empty"/>
    </report>
  </xsl:when>
```

Example 72 Dummy Task Use Cases – Positive Reports are Generated

The service data update section creates *<task>* documents to modify the service data according to the user input. Once the *<task>* has been generated, the further processing is the same as described in Section 4.1.3 Workflow Overview on page 36. An appropriate XCAP request is sent to the XCAP servlet which applies the service-specific application rules on the request and updates the HSS with the modified service data. The GenSSC engine creates a *<report>* from the XCAP response (the report ID is inherited from the task ID). The report is compared with the announcement table and the matching voice message is sent to the user. The tasks of the service data update are shown in Example 73.

```
<!-- The service data needs to be modified. -->
        <xsl:otherwise>
          <task-list version="1"
              xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
              xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
              xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
              xmlns:cp="urn:ietf:params:xml:ns:common-policy">
            <!-- Remove the rules to be deleted. -->
            <xsl:if test="$rulesToBeDeleted != '|'">
              <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion⇒
/cp:ruleset
                  /cp:rule[contains($rulesToBeDeleted, concat('|', @id, '|'))]">
                <task version="1">
                  <xsl:attribute name="id">
                    <xsl:value-of select="concat($prefix, 'delete-', @id)"/>
                  </xsl:attribute>
                  <head>
                    <action>delete</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='@id'/>"]
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                    </xsl:element>
                  </head>
                </task>
              </xsl:for-each>
            </xsl:if>
            <!-- Remove the referred identities from the rules to be updated. -->
            <xsl:if test="$rulesToBeUpdated != '|'">
              <xsl:for-each select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion⇒
/cp:ruleset
                  /cp:rule[contains($rulesToBeUpdated, concat('|', @id, '|'))]">
                <task version="1">
                  <xsl:attribute name="id">
                    <xsl:value-of select="concat($prefix, 'update-', @id)"/>
                  </xsl:attribute>
                  <head>
                    <action>set</action>
                    <content-type>application/xcap-el+xml</content-type>
                    <xsl:element name="ut-path">simservs/communication-diversion/cp:ruleset
                        /cp:rule[@id="<xsl:value-of select='@id'/>"]
                        /cp:conditions/mmt-serv:served-identity
                        ?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
                        xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
                    </xsl:element>
                  </head>
                  <body>
                    <mmt-serv:served-identity>
                      <!-- Copy the mmt-serv:one elements not including the referred identities. -->
                      <xsl:call-template name="copyNotReferredOnes">
                         <xsl:with-param name="ruleName" select="@id"/>
                         <xsl:with-param name="identities" select="$identities"/>
                      </xsl:call-template>
                     </mmt-serv:served-identity>
                  </body>
                </task>
              </xsl:for-each>
            </xsl:if>
          </task-list>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:template>
  </xsl:stylesheet>
</layout
```

Example 73 Service Data Update – Tasks are Generated

Taking a look at the source document once again, it turns out that the current provisioning and the user input result in two tasks: the simple-msn-cfb-to-bob rule is deleted and the simple-msn-cfb-to-0046222222222 is updated by removing alias 2 from the served-identity condition. Both of the generated tasks are shown in Example 74.

```
<?xml version="1.0"?>
<task-list version="1"
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy">

  <task version="1" id="msn_busy_delete-simple-msn-cfb-to-bob">
    <head>
      <action>delete</action>
      <content-type>application/xcap-el+xml</content-type>
      <ut-path>
        simservs/communication-diversion/cp:ruleset
            /cp:rule[@id="simple-msn-cfb-to-bob"]?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
      </ut-path>
    </head>
  </task>

  <task version="1" id="msn_busy_update-simple-msn-cfb-to-0046222222222">
    <head>
      <action>set</action>
      <content-type>application/xcap-el+xml</content-type>
      <ut-path>
        simservs/communication-diversion/cp:ruleset
            /cp:rule[@id="simple-msn-cfb-to-0046222222222"]/cp:conditions
            /mmt-serv:served-identity?xmlns(cp=urn:ietf:params:xml:ns:common-policy)
            xmlns(mmt-serv=http://schemas.ericsson.com/mmtel/services)
      </ut-path>
    </head>
    <body>
      <mmt-serv:served-identity>
        <mmt-serv:one id="sip:+46111111111@operator.com"/>
      </mmt-serv:served-identity>
    </body>
  </task>

</task-list>
```

Example 74 Result of the XSLT Transformation – A Task List with Two Tasks

The announcement played to the user is:`The referred identities have been removed from the MSN CFB rules. You still have active MSN CFB rules.` (media-id=”1161”)

4.4.6 Support for MSN Conditional CDIV Get Command

This section contains an example for a complex composite configuration which analyzes the service data keeping it unchanged and plays an appropriate announcement to the user. The interrogation operation checks if there exists any simple MSN enhanced Conditional CDIV rule of a given type for a given alias. The verification operation checks if there exists any simple MSN enhanced Conditional CDIV rule of a given type for a given alias and if it has a specific target. These rules are members of the CDIV rule set, see Example 35.

4.4.6.1 Subject of the Example

The user story declares a subject of this example.

An IMS user wants to interrogate the state or verify the forward-to target value of the MSN enhanced CDIV rule of a given type referred by the alias, using the SSC command formats: `*11*<alias>#*#<service_code>#` (interrogation) and `*11*<alias>#*#<service_code><target>#` (verification).

Example for a related SSC command (interrogation):

```
*11*2#*#67#
```

Example for a related SSC command (verification):

```
*11*2#*#670046222222222#
```

The user wants to be informed of the following:

- If there is any active simple MSN enhanced CDIV rule of a given type including the referred identity (interrogation and verification)
- If the simple MSN enhanced CDIV rule of a given type for the referred identity includes the entered target number (verification)
- If an error occurred

4.4.6.2 Configuration Overview

This section shows an example configuration of an `MtasGenSscCmd` MO instance which is in line with the above description.

```
MtasGenSscCmd: MsnCDivCondGet (key)
```

The "MsnCDivCondGet" string identifies this `MtasGenSscCmd` MO instance.

```
mtasGenSscCmdHeader: *11*1#*#6|*11*2#*#6|*11*3#*#6
```

The command header attribute lists the variations of leading digits related to this configuration. They differ with the alias ID in the middle.

The command syntax chosen in this example allows for using of three aliases at most, owing to the decoding tree limitations. Using more aliases would be possible with a different SSC syntax, for example, `#*11<service_code>*<alias>#` and `#*11<service_code>*<alias>*<target>#`. In that case, the command header attribute would have the value of `#*11<first_digit_of_the_service_code>`, for example, `#*116`.

```
mtasGenSscCmdSyntax:
/^\*11\*([1-3])#\*#61#$/schema=composite; number alias=\1;
item target=none; item condition=ss:no-answer;
item prefix=msn_cfnr_/~~
/^\*11\*([1-3])#\*#61([0-9]+)#$/schema=composite; number alias=\1;
item target=tel:\2; item condition=ss:no-answer;
item prefix=msn_cfnr_/~~
/^\*11\*([1-3])#\*#62#$/schema=composite; number alias=\1;
item target=none; item condition=ss:not-reachable;
item prefix=msn_cfnrc_/~~
/^\*11\*([1-3])#\*#62([0-9]+)#$/schema=composite; number alias=\1;
item target=tel:\2; item condition=ss:not-reachable;
item prefix=msn_cfnrc_/~~
```

```
/^\*11\*([1-3])#\*#63#$/schema=composite; number alias=\1;
item target=none; item condition=ss:not-registered;
item prefix=msn_cfnl_/~~
/^\*11\*([1-3])#\*#63([0-9]+)#$/schema=composite; number alias=\1;
item target=tel:\2; item condition=ss:not-registered;
item prefix=msn_cfnl_/~~
/^\*11\*([1-3])#\*#67#$/schema=composite; number alias=\1;
item target=none; item condition=ss:busy;
 item prefix=msn_busy_/~~
/^\*11\*([1-3])#\*#67([0-9]+)#$/schema=composite; number alias=\1;
item target=tel:\2; item condition=ss:busy; item prefix=msn_busy_/
```

The mtasGenSscCmdSyntax consists of eight parts – one for interrogation and one for verification operations for each kind of conditional CDIV. Every part contains two subparts: a Command Pattern and a Command Syntax.

The example part defining a syntax for the interrogation operation (MSN CFB here) looks like the following: `^\*11\*([1-3])#\*#67#$/schema=composite; number alias=\1; item target=none; item condition=ss:busy; item prefix=msn_busy_`.

The Command Pattern `^\*11\*([1-3])#\*#67([0-9]+)#$` declares the format of the Request URI.

The Command Syntax part `schema=composite; number alias=\1; item target=tel:\2; item condition=ss:busy; item prefix=msn_busy_` informs the GenSSC engine that a composite configuration is processed. It also declares four variables to be fetched from an SSC command: alias of number type to be taken from the first variable part of the command, target of item type to be taken from the second variable part, condition of item type which contains an appropriate CDIV condition, and prefix of item type which contains an appropriate prefix to be used as a part of a report ID.

`mtasGenSscCmdUtPath:`

```
simservs/communication-diversion~~simservs/mmt-serv:⇒
flexible-identity-presentation
```

The UtPath points to the CDIV and FIP services, they are copied from the *<service-data>* to the *<source>* document.

`mtasGenSscCmdAnnouncements:`

The announcement table is described in Section 4.4.6.3.3 Announcement Configuration on page 152.

`mtasGenSscCmdLayout:`

The layout document is described in Section 4.4.6.3.6 Layout Implementation on page 155.

Other attributes:

The attributes `mtasGenSscCmdReplace`, `mtasGenSscCmdContentType`, and `mtasGenSscCmdUriInChargingMessages` of the `MtasGenSscCmd` MO are not used in this configuration, the default values are assigned to them.

4.4.6.3 Workflow Overview

This section shows the workflow overview.

4.4.6.3.1 Understanding the Details

The details of the Delete command include similar difficulties as are described by the Set command, refer to Section 4.4.4.3.1 Understanding the Details on page 107.

4.4.6.3.2 Clarification of the Use Cases

The user's service profile includes the following information:

- If any simple active MSN enhanced Conditional CDIV rules of a given type for the alias ID given in the SSC command exist in the CDIV rule set
- If any simple active MSN enhanced Conditional CDIV rules of a given type for the alias ID and the target given in the SSC command exist in the CDIV rule set

The type of operation (interrogation or verification) is determined by the SSC command syntax.

The combinations of these CFU rule information categories define the use cases of the user story. They are the input parameters which help to determine an announcement to be played to the user.

Table 16 summarizes the use cases to be considered and the announcements to be played for each use case.

Table 16 Use Cases of SSC Command for Interrogating or Verifying an MSN Enhanced Conditional CDIV Rule

Use Case	Type of Conditional CDIV	Type of Operation	Simple Rules of a Given Type for a Given Alias Exist	A Simple Rule of a Given Type for a Given Alias AND a Given Target Exists	Announcement
UC-0	CFNR	Irrelevant	No	No	There are no active MSN CFNR rules including the referred identity.
UC-1	CFNR	Interrogation	Yes	Irrelevant	Your MSN CFNR rule including the referred identity is active.
UC-2	CFNR	Verification	Yes	Yes	The MSN CFNR rule for the referred identity includes the entered target number.
UC-3	CFNR	Verification	Yes	No	The MSN CFNR rule for the referred identity includes a different target.

Use Case	Type of Conditional CDIV	Type of Operation	Simple Rules of a Given Type for a Given Alias Exist	A Simple Rule of a Given Type for a Given Alias AND a Given Target Exists	Announcement
UC-4	CFNRc	Verification	No	No	There are no active MSN CFNRc rules including the referred identity.
UC-5	CFNRc	Interrogation	Yes	Irrelevant	Your MSN CFNRc rule including the referred identity is active.
UC-6	CFNRc	Verification	Yes	Yes	The MSN CFNRc rule for the referred identity includes the entered target number.
UC-7	CFNRc	Verification	Yes	No	The MSN CFNRc rule for the referred identity includes a different target.

Use Case	Type of Conditional CDIV	Type of Operation	Simple Rules of a Given Type for a Given Alias Exist	A Simple Rule of a Given Type for a Given Alias AND a Given Target Exists	Announcement
UC-8	CFNL	Irrelevant	No	No	There are no active MSN CFNL rules including the referred identity.
UC-9	CFNL	Interrogation	Yes	Irrelevant	Your MSN CFNL rule including the referred identity is active.
UC-10	CFNL	Verification	Yes	Yes	The MSN CFNL rule for the referred identity includes the entered target number.
UC-11	CFNL	Verification	Yes	No	The MSN CFNL rule for the referred identity includes a different target.
UC-12	CFB	Irrelevant	No	No	There are no active MSN CFB rules including the referred identity.

Use Case	Type of Conditional CDIV	Type of Operation	Simple Rules of a Given Type for a Given Alias Exist	A Simple Rule of a Given Type for a Given Alias AND a Given Target Exists	Announcement
UC-13	CFB	Interrogation	Yes	Irrelevant	Your MSN CFB rule including the referred identity is active.
UC-14	CFB	Verification	Yes	Yes	The MSN CFB rule for the referred identity includes the entered target number.
UC-15	CFB	Verification	Yes	No	The MSN CFB rule for the referred identity includes a different target.

A summary of the announcement variations is shown in Table 17.

Table 17 Summary of Announcement Variations - Interrogating or Verifying an MSN Enhanced Conditional CDIV Rule

Announcement	Media ID	Use cases
There are no active MSN CFNR rules including the referred identity.	1211	0
Your MSN CFNR rule including the referred identity is active.	1212	1
The MSN CFNR rule for the referred identity includes the entered target number.	1213	2

The MSN CFNR rule for the referred identity includes a different target.	1214	3
There are no active MSN CFNRc rules including the referred identity.	1221	4
Your MSN CFNRc rule including the referred identity is active.	1222	5
The MSN CFNRc rule for the referred identity includes the entered target number.	1223	6
The MSN CFNRc rule for the referred identity includes a different target.	1224	7
There are no active MSN CFNL rules including the referred identity.	1231	8
Your MSN CFNL rule including the referred identity is active.	1232	9
The MSN CFNL rule for the referred identity includes the entered target number.	1233	10
The MSN CFNL rule for the referred identity includes a different target.	1234	11
There are no active MSN CFB rules including the referred identity.	1201	12
Your MSN CFB rule including the referred identity is active.	1202	13
The MSN CFB rule for the referred identity includes the entered target number.	1203	14
The MSN CFB rule for the referred identity includes the entered target number.	1204	15

All the use cases in Table 17 represent dummy tasks. The service data is not changed. It is enough to create a proper report.

The layout creates a positive *<report>* for every use case.

4.4.6.3.3 Announcement Configuration

The layout is constructed in a way which provides the necessary information for the announcement.

In this example, the “id” attributes of the reports are given descriptive values, which enhance readability and simplify configuration of the announcement table, see Example 75.

```
<?xml version="1.0" encoding="utf-8"?>
 <announcements xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
   <announcement comment="There are no active MSN CFNR rules including the referred identity.⇒
" media-id="1211">
     <play-condition id="msn_cfnr_deactivated"/>
   </announcement>
   <announcement comment="Your MSN CFNR rule including the referred identity is active." media-id="1212">
     <play-condition id="msn_cfnr_active"/>
   </announcement>
   <announcement comment="The MSN CFNR rule for the referred identity includes the entered target number."
      media-id="1213">
     <play-condition id="msn_cfnr_match"/>
   </announcement>
   <announcement comment="The MSN CFNR rule for the referred identity includes a different target."
      media-id="1214">
     <play-condition id="msn_cfnr_mismatch"/>
   </announcement>
   <announcement comment="There are no active MSN CFNRc rules including the referred identity."⇒
media-id="1221">
     <play-condition id="msn_cfnrc_deactivated"/>
   </announcement>
   <announcement comment="Your MSN CFNRc rule including the referred identity is active." media-id="1222">
     <play-condition id="msn_cfnrc_active"/>
   </announcement>
   <announcement comment="The MSN CFNRc rule for the referred identity includes the entered target number."
      media-id="1223">
     <play-condition id="msn_cfnrc_match"/>
   </announcement>
   <announcement comment="The MSN CFNRc rule for the referred identity includes a different target."
      media-id="1224">
     <play-condition id="msn_cfnrc_mismatch"/>
   </announcement>
   <announcement comment="There are no active MSN CFNL rules including the referred identity."⇒
media-id="1231">
     <play-condition id="msn_cfnl_deactivated"/>
   </announcement>
   <announcement comment="Your MSN CFNL rule including the referred identity is active." media-id="1232">
     <play-condition id="msn_cfnl_active"/>
   </announcement>
   <announcement comment="The MSN CFNL rule for the referred identity includes the entered target number."
      media-id="1233">
     <play-condition id="msn_cfnl_match"/>
   </announcement>
   <announcement comment="The MSN CFNL rule for the referred identity includes a different target."
      media-id="1234">
     <play-condition id="msn_cfnl_mismatch"/>
   </announcement>   <announcement comment="There are no active MSN CFB rules including the referred⇒
identity." media-id="1201">
     <play-condition id="msn_busy_deactivated"/>
   </announcement>
   <announcement comment="Your MSN CFB rule including the referred identity is active." media-id="1202">
     <play-condition id="msn_busy_active"/>
   </announcement>
   <announcement comment="The MSN CFB rule for the referred identity includes the entered target number."
      media-id="1203">
     <play-condition id="msn_busy_match"/>
   </announcement>   <announcement comment="The MSN CFB rule for the referred identity includes a different⇒
target."
      media-id="1204">
     <play-condition id="msn_busy_mismatch"/>
   </announcement>
   <announcement comment="Internal error occured" media-id="1800"/>
 </announcements>
```

Example 75 Announcement Table for Interrogating or Verifying an MSN Enhanced Conditional CDIV Rule

4.4.6.3.4 Constructing Source Document Variants

An instance of the possible `<source>` document variants is shown in Example 76. This instance represents the use cases 13 and 14. The comments are not generated by the GenSSC engine, they serve only for better understanding.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This document is passed to the XSL Transform function as the source XML of the transformation.
     The <service-data> element represents a provisioning of the use cases 13 and 14. -->
<source version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
    xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy"
    xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services">

  <!-- The <user-id> comes from the P-Asserted-Identity header of the SSC-INVITE. -->
  <user-id>sip:abc@ericsson.com</user-id>

  <!-- The <command-data> contains the elements of the extracted command syntax.
       It includes the schema type in this case. -->
  <command-data version="1">
    <schema>composite</schema>
  </command-data>

  <!-- The <service-data> contains a piece of the user's simservs document. The mtasGenSscCmdUtPath determines
       which elements shall be taken over from the user's service data.
       There are two <service-data> elements in this case, including the CDIV and FIP services. -->
  <service-data index="0" valid="true" version="1"
      path="simservs/communication-diversion">
    <ss:communication-diversion active="true">
      <cp:ruleset>
        <!-- The "msn-video-to-alice" rule is a combined 'served-identity' and 'media' rule. It is not a
             simple MSN enhanced conditional CDIV rule in the meaning described in Section 4.4.2.1.3.1, that is
             why it is out of scope of this configuration example and is filtered out. -->
        <cp:rule id="msn-video-to-alice">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111112@operator.com"/>
            </mmt-serv:served-identity>
            <ss:media>video</ss:media>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:alice@example.com</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfb-to-0046222222222" rule is an active simple MSN enhanced CFB rule. The
             'served-identity' condition of the rule contains the identity referred in the SSC command of the
             user story and the forward-to target is also the same as in the SSC command. -->
        <cp:rule id="simple-msn-cfb-to-0046222222222">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111112@operator.com"/>
            </mmt-serv:served-identity>
            <ss:busy/>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>tel:0046222222222</ss:target>
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
        <!-- The "simple-msn-cfb-to-bob" rule is an active simple MSN enhanced CFB rule. It has a different
             alias and forward-to target than the alias and target in the SSC command of the user story. -->
        <cp:rule id="simple-msn-cfb-to-bob">
          <cp:conditions>
            <mmt-serv:served-identity>
              <mmt-serv:one id="sip:+46111111113@operator.com"/>
            </mmt-serv:served-identity>
            <ss:busy/>
          </cp:conditions>
          <cp:actions>
            <ss:forward-to>
              <ss:target>sip:bob@example.com</ss:target>
```

```
            </ss:forward-to>
          </cp:actions>
        </cp:rule>
      </cp:ruleset>
    </ss:communication-diversion>
  </service-data>

  <service-data index="1" valid="true" version="1"
      path="simservs/mmt-serv:flexible-identity-presentation">
    <mmt-serv:flexible-identity-presentation active="true">
      <!-- The fip-identity element is not analyzed by the Generic SSC according to this example⇒
 configuration.
           It can contain either a router address or a main number of the subscriber. In the latter case,⇒
 the
           main number should also be configured in the msn-fip-identity with id="1". -->
      <mmt-serv:fip-identity>sip:something@operator.com</mmt-serv:fip-identity>
      <!-- Alias ID "1" in an SSC command will be replaced with alias PUI sip:+46111111111@operator.com in
           operations on the CDIV service data. This is the subscriber's main (first alias) number . -->
      <mmt-serv:msn-fip-identity id="1">
        <mmt-serv:identity>sip:+46111111111@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "2" in an SSC command will be replaced with alias PUI sip:+46111111112@operator.com in
           operations on the CDIV service data. This is the subscriber's second alias number. -->
      <mmt-serv:msn-fip-identity id="2">
        <mmt-serv:identity>sip:+46111111112@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
      <!-- Alias ID "3" in an SSC command will be replaced with alias PUI sip:+46111111113@operator.com in
           operations on the CDIV service data. This is the subscriber's third alias number. -->
      <mmt-serv:msn-fip-identity id="3">
        <mmt-serv:identity>sip:+46111111113@operator.com</mmt-serv:identity>
      </mmt-serv:msn-fip-identity>
    </mmt-serv:flexible-identity-presentation>
  </service-data>
</source >
```

Example 76 Generated <source> Document

4.4.6.3.5 Layout Design

The following internal data is defined in this example:

- The alias ID from the SSC command (alias)
- The target from the SSC command including the 'tel:' prefix (target)
- The condition referring to the rule type (condition)
- The report ID prefix (prefix)
- The identity (alias PUI) addressed by the alias ID (identity)
- The ID of the simple MSN enhanced Conditional CDIV rule of a given type which contains PUI stored in the 'identity' variable (ruleName)
- The target of the rule with an ID stored in the 'ruleName' variable

4.4.6.3.6 Layout Implementation

This section includes a concrete layout implementation of the user story in Section 4.4.4.3.6 Layout Implementation on page 120.

The top part of the layout includes definitions of the *<xsl:variable>* elements, see Example 77.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This layout document contains an XSL style sheet which interrogates the state or verifies the⇒
forward-to
     target value of the MSN enhanced Conditional CDIV rule of a given type (CFB, CFNR, CFNRc or CFNL)⇒
referred by
     the alias.
     Input parameters coming from the command data:
      - alias: includes the MSN alias implicitly referring to the rule to be interrogated
      - target: forward-to target number including the 'tel:' prefix or 'none'
      - condition: includes the name of the CDIV condition element (ss:busy, ss:not-reachable, ...)
      - prefix: includes a name prefix referring to the type of an MSN conditional CDIV rule (msn_cfb_,
        msn_cfnrc_, ...)
     Service data elements used by the style sheet:
      - 0: /simservs/communication-diversion
      - 1: /simservs/flexible-identity-presentation
-->
<layout version="1"
    name="Interrogation of the MSN enhanced Conditional CDIV rule referred by the alias"
    xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
      xmlns:gs="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc"
      xmlns:ss="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
      xmlns:mmt-serv="http://schemas.ericsson.com/mmtel/services"
      xmlns:cp="urn:ietf:params:xml:ns:common-policy"
      exclude-result-prefixes="gs">

    <xsl:output method="xml" indent="yes"/>

    <!-- Main template. -->
    <xsl:template match="/">

      <!-- ****************************************
           * Variable definitions                 *
           **************************************** -->

      <!-- Store the alias in a variable. -->
      <xsl:variable name="alias">
        <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='alias']"/>
      </xsl:variable>

      <!-- Store the target including the 'tel:' prefix in a valiable. -->
      <xsl:variable name="target">
        <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='target']"/>
      </xsl:variable>

      <!-- Store the condition referring to the rule type in a variable. -->
      <xsl:variable name="condition">
        <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='condition']"/>
      </xsl:variable>

      <!-- Store the report id prefix in a variable. -->
      <xsl:variable name="prefix">
        <xsl:value-of select="/gs:source/gs:command-data/gs:data[@name='prefix']"/>
      </xsl:variable>

      <!-- Store the identity addressed by the alias in a variable. -->
      <xsl:variable name="identity">
        <xsl:value-of select="/gs:source/gs:service-data[@index='1']
            /mmt-serv:flexible-identity-presentation/mmt-serv:msn-fip-identity[@id=$alias]"/>
      </xsl:variable>

      <!-- Store the name of the rule addressed by the alias in a variable. -->
      <xsl:variable name="ruleName">
        <xsl:value-of select="/gs:source/gs:service-data[@index='0']
            /ss:communication-diversion/cp:ruleset/cp:rule[
            count(cp:conditions/*)=2
            and count(cp:conditions[$condition])=1
            and count(cp:conditions/mmt-serv:served-identity/mmt-serv:one[@id=$identity])=1
            ]/@id"/>
      </xsl:variable>
```

```
      <!-- Store the rule's target in a valiable. -->
      <xsl:variable name="ruleTarget">

        <xsl:value-of select="/gs:source/gs:service-data[@index='0']/ss:communication-diversion/cp:ruleset
            /cp:rule[@id=$ruleName]/cp:actions/ss:forward-to/ss:target"/>
      </xsl:variable>
```

Example 77 Variable Definitions

The bottom part of the layout document comprises the generation of the output. It is divided into several sections. They refer to the use cases of Table 16, see Example 78.

```
      <!-- ***************************************
           * Output generation                   *
           *************************************** -->

      <xsl:choose>
        <xsl:when test="(string-length($ruleName)=0) and ($target='none')">
          <!-- No rule found - the rule for a given MSN identity has been deactivated. -->
          <report version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
            <xsl:attribute name="id">
              <xsl:value-of select="concat($prefix, 'deactivated')"/>
            </xsl:attribute>
            <response>200 OK</response>
            <job>get</job>
            <value type="empty"/>
          </report>
        </xsl:when>
        <xsl:when test="$target='none'">
          <!-- Rule found - the rule for a given MSN identity is active. -->
          <report version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
            <xsl:attribute name="id">
              <xsl:value-of select="concat($prefix, 'active')"/>
            </xsl:attribute>
            <response>200 OK</response>
            <job>get</job>
            <value type="empty"/>
          </report>
        </xsl:when>
        <xsl:when test="$ruleTarget=$target">
          <!-- The rule's target matches the input. -->
          <report version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
            <xsl:attribute name="id">
              <xsl:value-of select="concat($prefix, 'match')"/>
            </xsl:attribute>
            <response>200 OK</response>
            <job>verify</job>
            <value type="simple"><xsl:value-of select="$ruleTarget"/></value>
          </report>
        </xsl:when>
        <xsl:otherwise>
          <!-- The rule's target doesn't match the input. -->
          <report version="1" xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
            <xsl:attribute name="id">
              <xsl:value-of select="concat($prefix, 'mismatch')"/>
            </xsl:attribute>
            <response>200 OK</response>
            <job>verify</job>
            <value type="simple"><xsl:value-of select="$ruleTarget"/></value>
          </report>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:template>
  </xsl:stylesheet>
</layout>
```

Example 78 Generation of the Output – Positive Reports are Generated

Reviewing the source document once again, it turns out that the current provisioning and the user input result in positive reports in both cases of interrogation and verification operations of the user story. The simple-msn-cfb-to-0046222222222 rule is an active simple MSN enhanced CFB rule with the 'served-identity' condition containing the identity referred in the SSC commands and the same forward-to target. The generated reports for both operations are shown in Example 79 and Example 80.

```
<?xml version="1.0"?>
<report version="1" id="msn_busy_active"
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <response>200 OK</response>
  <job>get</job>
  <value type="empty"/>
</report>
```

Example 79 Result of the Interrogation Operation – A Positive Report

```
<?xml version="1.0"?>
<report version="1" id="msn_busy_match"
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap/genssc">
  <response>200 OK</response>
  <job>verify</job>
  <value type="simple">simple-msn-cfb-to-0046222222222</value>
</report
```

Example 80 Result of the Verification Operation – A Positive Report

In case of the interrogation operation, the announcement played to the user is: `Your MSN CFB rule including the referred identity is active.` (media-id="1202")

In case of the verification operation the announcement played to the user is: `The MSN CFB rule for the referred identity includes the entered target number.` (media-id="1203")

4.5 PIN Employment

The `pinCheck()` function can be applied in both simple and the composite schemas, see Section 3.2.4.2 pinCheck() Function on page 31.

This section includes an example of how to employ the PIN in the user story, see Section 4.1.1 Subject of Example on page 33. The only extension is that the user has to enter the PIN next to the service code. For example, the SSC command activating the Hotline service is `*111*1234#`.

The digits `1234` represent the user PIN in the SSC command.

The following attributes must be adapted for this extension:

- mtasGenSscCmdHeader
- mtasGenSscCmdSyntax
- mtasGenSscCmdUriInChargingMsgs

The mtasGenSscCmdHeader and the mtasGenSscCmdSyntax attributes are related to the included PIN. It means that the URI never ends with the service code but always contains a PIN next to the service code.

The modified mtasGenSscCmdHeader attribute is *111*|#111*|*#111*.

The modified mtasGenSscCmdSyntax attribute:

```
/^([*#]#?)111\*([0-9]{4-8})#$/schema=simple;
function pinCheck(\2); function serviceState(\1)/~~

/^\*111\*([0-9]{4-8})\*([0-9]{4-14})#$/schema=simple;
function pinCheck(\1); action=update; digits st-num=\2;
function serviceState(activate)/~~

/^\*#111\*([0-9]{4-8})\*([0-9]{4-14})#$/schema=simple;
function pinCheck(\1); action=verify; digits st-num=\2/
```

The mtasGenSscCmdUriInChargingMsgs attribute defines the rule how to mask the PIN in the URI included in the charging messages:

```
/([*#]#?)111\*[0-9]{4-8}/\1111\*0000/
```

The rule copies the prefix and the service code and replaces the PIN with four zeros.

The invocation of the pinCheck() function triggers the built-in PIN check function of the SSC service. The PIN check is a common function of the system-defined and the GenSSC features. The announcements to be played on a PIN-related fault and the other behaviors like the default PIN handling are configured in the system defined MtasSsc MO. For more information, refer to the following documents:

- *Managed Object Model (MOM)*
- *MTAS Wholesale Support Management Guide*

The only exception is that the PIN length is defined in the command pattern/translation pattern. That is, the mtasSscLengthOfPin attribute is not used by the GenSSC.

If the PIN check fails, then an appropriate announcement is played to the user and the SSC call is released, for the procedures description, see Section 4.1.3 Workflow Overview on page 36.

5 Troubleshooting Command Configuration

The troubleshooting of the command configuration is based on the log information produced by the GenSSC engine. The engine is divided into two parts; GenSSC Adapter and GenSSC Servlet.

5.1 GenSSC Adapter

The GenSCC adapter processes, the incoming `SSC-INVITE` determines which `mtasGenSscCmd` MO is referred by the `INVITE` request, performs the substitutions in the command syntax, and sends the extracted command data to the GenSSC servlet in the XML Document Management Server (XDMS) side in form of a `<command>` document.

In turn, the GenSSC adapter receives a `<report>` document, increases the PM counters, generates charging messages, and plays an announcement to the user.

5.2 GenSSC Servlet

The GenSSC servlet receives a `<command>` document from the GenSSC adapter, generates, and processes all the intermediate GenSSC XML documents (source, task), generates the XCAP requests, processes the XCAP responses, and generates a `<report>` at the end.

Since all GenSSC-related XML documents are processed by the GenSSC servlet, the servlet log contains more details about the possible faults than the adapter printouts. The servlet log cannot be used only if the processing of the `SSC-INVITE` fails at the beginning and the GenSSC adapter does not send a `<command>` to the servlet. The adapter log can be used in this case.

Both the GenSSC servlet and the GenSSC adapter use the processor logs for tracing purposes. The Apptrace feature can be used to get more information about the problem. The trace level must be set to `Debug-trace (55)` to have the printouts in the processor log. For more information on the Apptrace feature, refer to *AppTrace User Guide*.

The GenSSC servlet log is part of one of the processor logs. The servlet is bounded to the trace domain `ims.mtas.xdms.genssc`. The servlet generates log printouts if the trace level of the trace domain `ims.mtas.xdms.genssc` is set to `Debug-trace`.

The GenSSC adapter log is part of one of the processor logs. The adapter is bounded to the trace domain `ims.mtas.services.ssc`. The adapter generates log printouts if the trace level of the trace domain `ims.mtas.services.ssc` is set to `Debug-trace`.

6 Performance Management

The GenSSC always steps the invocation-related counters (`MtasGenSscInv[Ok/Err/NOkE/NOkI/NOkP]`) in case of composite schema since it cannot be determined what operations are configured in the layout (`mtasGenSscCmdLayout`).

For measurements related to the GenSSC service, refer to *Managed Object Model (MOM)*.

The granularity of PM is configurable for the GenSSC. The counter types listed in this section can be created on group level or command level. The operator can select the PM level for each GenSSC command group (`MtasGenSscGroup` MO).

There are two groups of performance counters according to the started subscriber data use cases, as follows:

- Service Interrogation-related PM counters

  The appropriate counter is incremented if the generic service code command started only a single ‘‘Get data’’ use case.

- Service Invocation-related PM counters

  The appropriate counter is incremented if the generic service code command starts at least one ‘‘Update data’’ use case or more than one ‘‘Get data’’ use cases.

The following types of counters are considered for the generic SSCs in the MTAS:

- Number of successful interrogations
- Number of incorrect interrogations
- Number of unsuccessful interrogations owing to an MTAS internal fault
- Number of unsuccessful interrogations owing to an MTAS external fault
- Number of unsuccessful interrogations owing to a subscriber data configuration fault
- Number of successful invocations
- Number of incorrect invocations
- Number of unsuccessful invocations owing to an MTAS internal fault
- Number of unsuccessful invocations owing to an MTAS external fault

- Number of unsuccessful invocations owing to a subscriber data configuration fault

7 Fault Management

The GenSSC service has no alarms.