ERICSSON

# Management over REST for Resource Configuration

## Ericsson Dynamic Activation 1

## INTERFACE DESCRIPTION

33/155 19-CSH 109 628 Uen M

# Contents

# 1 Introduction

This document describes the Management Representational State Transfer (REST) API for Resource Configuration. The API offers the possibility to:

— Add, remove, update, and get devices in the Device Repository.

— Management of device information like login credentials, Device Type, and the use of Resource Configuration features like Candidate Store and Configuration Validation.

— Trigger Device Discovery and Reconciliation.

— Import and Export of Feature Models.

— Update the relation between a template and Device Types.

For more information about the device-related functionality and Feature Models, refer to Function Specification Resource Configuration, Reference [1] and User Guide for Resource Configuration, Reference [2].

## 1.1 Purpose and Scope

All functionality exposed over the REST API for Resource Configuration is also available in the Resource Configuration GUI. For smaller solutions of Resource Configuration, it is convenient to add and remove devices manually in the system, using this GUI. When the number of devices is increasing, it is recommended to automate the introduction of new devices in the solution. The REST API described in this document is then needed to manage the devices in Resource Configuration from another system. The Import and Export of Feature Models allows to build automatic deployments processes when moving Feature Models from one system to another.

## 1.2 Target Group

The target group for this document is as follows:

— System Integrator

For more information about the different target groups, see Library Overview, Reference [3].

## 1.3 Typographic Conventions

Typographic conventions are described in Library Overview, Reference [3].

## 1.4 Prerequisites

To use this document fully, users must meet the following prerequisites:

— Basic knowledge about the Dynamic Activation product

— Knowledge about REST.

— Knowledge about json.

For information about Resource Configuration in Ericsson Dynamic Activation (EDA), refer to Function Specification Resource Configuration, Reference [1].

## 1.5 Operations

The following HTTP request methods are used:

— GET

The GET method requests a representation of the specified resource. Requests using GET only retrieves data and have no other effect.

— POST

The POST method requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI, or an item to add to a database.

— PUT

The PUT method requests that the enclosed entity is to be stored under the supplied URI. If the URI refers to an existing resource, it is modified.

— DELETE

The DELETE method deletes the specified resource.

See the following table for all available URLs and corresponding operations.

Table 1 URLs and Corresponding Operations

| URL | Operations | | | | Content Type | |
|---|---|---|---|---|---|---|
| | GET | POST | PUT | DELETE | Request | Response |
| /scm-rest/device-repository/admin-states | X | | | | | application/json |
| /scm-rest/device-repository/device-categories | X | X | | | application/json | application/json |
| /scm-rest/device-repository/devices | X | X | | | application/json | application/json |
| /scm-rest/device-repository/devices/{identifier} | X | | X | X | application/json | application/json |

| URL | Operations | | | | Content Type | |
|---|---|---|---|---|---|---|
| | GET | POST | PUT | DELETE | Request | Response |
| /scm-rest/device-repository/devices/{identifier}/config | X | X | | | application/xml | application/xml |
| /scm-rest/device-repository/devices/{identifier}/config-sync | | X | | | application/json | application/json |
| /scm-rest/device-repository/devices/{identifier}/force | | | | X | | application/json |
| /scm-rest/device-repository/devices/{identifier}/config/backups/{timestamp}/restore | | X | | | application/json | |
| /scm-rest/device-repository/devices/{identifier}/config/backups | | X | | | | |
| | X | | | | | application/json |
| /scm-rest/device-repository/devices/config/backups | | X | | | application/json | application/json |
| /scm-rest/device-repository/jobs/{jobId} | X | | | | | application/json |
| /scm-rest/device-repository/devices/{identifier}/config/backups/{timestamp} | X | | | | | application/json |
| /scm-rest/modelcatalog/import | | X | | | multipart/form-data | application/json |
| /scm-rest/modelcatalog/export | | X | | | application/json | applicatio n/zip |
| /scm-rest/modelcatalog/jobs/{jobId} | X | | | | | application/json |
| /scm-rest/modelcatalog/models/import/all | | X | | | multipart/form-data | application/json |
| /scm-rest/modelcatalog/models/export/{model name} | X | | | | | applicatio n/zip |
| /scm-rest/modelcatalog/models/export/all | X | | | | | applicatio n/zip |
| /scm-rest/template-management/import | | X | | | multipart/form-data | application/json |
| /scm-rest/template-management/export | | X | | | application/json | applicatio n/zip |
| /scm-rest/template-management/jobs/{jobId} | X | | | | | application/json |
| /scm-rest/template-management/device-types/{device-type} | X | | X | | application/json | application/json |
| /scm-rest/template-management/device-types | X | X | | | application/json | application/json |

## 1.6 Web Service Interface

Resource Configuration provides Web Application Description Language (WADL) file to use as a base when creating REST interface.

The `interface.zip` file containing the WADL file can be found in `/home/dveinstaller/ma/`. It is also possible to download the zip file and store it in an appropriate area by following below instruction:

1. Save the zip file, Dynamic Activation WSDL and XSD files.zip, to a local folder.

2. Unpack the zip file.

3. Go to `EDAinterfaces\webservice_provisioning_cai3g1.2\schemas\Applications\SCM`.

# 2 Authentication

All the requests in this document must include the HTTP Basic authentication header to provide username and password. The basic authentication string uses Base64 encoding.

## 2.1 Example

This section contains complete examples of authorization headers that can be reused in operations mentioned later in this document.

```
GET /scm-rest/device-repository/devices/device1/ HTTP/1.1
Host: 10.0.0.2:8383
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
Content-Type: application/json
```

Example 1 Complete Header Example

```
POST /scm-rest/modelcatalog/models/import/all HTTP/1.1
Host: 10.0.0.2:8383
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
Content-Type: multipart/form-data
```

Example 2 Complete Header Example Feature Models Import

**Note:** In both examples, the `dXNlcm5hbWU6cGFzc3dvcmQ=` is encoded from `username:password` by using Base64 format.

# 3 Device Management Data

This chapter describes the data attributes related to the device. For more information about each operation, see Section 4 on page 11.

Table 2 Data Attributes

| Property | Type | Occurrence | | Description |
|---|---|---|---|---|
| | | POST | PUT | |
| masterIdentifier | String | M | N/A | The unique identifier of the device (not possible to update). |
| identifiers | Array | O | O | Additional unique identifiers of each device. |
| adminState | Array | M | M | Specifies the availability of the device, if set to the ACTIVE it is possible to provision the device otherwise the device cannot be provisioned.<br>This parameter can be in OPERATION FAILED if an operation failed, for example a restore, then the user has to set it back to ACTIVE manually once the problem has been resolved. |
| modes | Array | O | O | Device modes. |
| CANDIDATESTORE | - | O | O | Specifies whether the Candidate store function is enabled or not for the specific device. If Candidate Store is enabled, the configuration for the selected device is stored in Candidate store before sent out to the device. |
| CONFIGURATIONVALIDATION | - | O | O | Specifies whether the Configuration validation is enabled or not for the specific device and what action to take if there is an inconsistency between configuration.<br>If enabled, a check is made that the actual configuration on the device matches the configuration last provisioned to the device. It must also be specified what action to take if the configurations do not match. |

| Property | Type | Occurrence | | Description |
|---|---|---|---|---|
| | | POST | PUT | |
| subModes | Array | O | O | Sub-modes property is only applicable for CONFIGURATION_VALIDATION. The following operations are available:<br>• DISPLAY ERROR: The user is notified with details of the discrepancies between the configurations.<br>• UPDATE DEVICE: Dynamic Activation updates the device with the stored device configuration.<br>• UPDATE REPOSITORY): Dynamic Activation updates the Resource Configuration device repository with the device configuration. If the configurations match, the device is provisioned with the new or updated configuration. If the configurations do not match, the action that was previously selected occurs. |
| types | Array | O | O | (Deprecated, see deviceType)<br>For backward compatibility, if several types are specified, these are concatenated, separated with the ‘‘#’’ sign, and stored as the Device Type. This field is not included when exporting devices, but is replaced with the `deviceType` filed. |
| deviceType | String | O | O | Specifies what Device Type the device is associated with. The Device Type is used to decide which template to use when generating the southbound commands to the device. A device can be associated with zero or one Device Type. |
| accessPoints | Array | M | M | Access points for the device. |
| protocol | String | M | M | Specifies what protocol to use for communication with the device.<br>Available protocols are SSH_CLI, SSH_NETCONF, and SNMP. |
| address | String | M | M | The address to the device according to format `<IP_address>:<port>`, for example 10.0.0.0:22. |
| parameters | key/value | O | O | Specifies a list of key value objects to set in the device. For example, providing a default parameter value. |

| Property | Type | Occurrence | | Description |
|---|---|---|---|---|
| | | POST | PUT | |
| protocolParameters | key/value | O | O | Specifies a list of key value objects for the communication protocol. For example, providing a protocol additional parameter value. |
| credentials | key/value | M | M | Specifies the authentication credentials to use for communication with the device. These are provided as key value objects. The credentials must be recognized by the device and the keys “username” and “password” must be present.<br><br>Optional Additional Parameter - Name can be added. For example, providing a default parameter value. |
| deviceTaskSchedules | Array | O | O | Task schedule for the device. |
| schedule | String | M | M | Specifies which time the action should be performed.<br><br>The syntax is:<br><br>`* * * * *`<br>`- - - - -`<br>`\| \| \| \| \|`<br>`\| \| \| \| ----- Day of week (0-7) (Sunday=0 or 7)`<br>`\| \| \| ------- Month (1-12)`<br>`\| \| --------- Day of month (1-31)`<br>`\| ----------- Hour (0-23)`<br>`------------- Minute (0-59)` |
| action | String | M | M | Specifies the action to perform.<br><br>The following action is supported: `BACKUP` |

| Property | Type | Occurrence | | Description |
|---|---|---|---|---|
| | | POST | PUT | |
| info | key/value | O | O | This is a field used by the system to inform about incidents or progress of long running tasks. For example, a restore that failed, or a reconciliation action that changed status.<br>The valid keys are:<br>• errorMsg: Contains information about a current incident related to this device. The field is cleared if Admin state is changed from OPERATION FAILED to ACTIVE.<br>• RECONCILIATION: Shows the progress of a reconciliation action. Predefined values are: STARTED, ONGOING, FINISHED, and FAILED.<br>• RECONCILIATION_ERROR: Shows a detailed error message related to the FAILED state of the RECONCILIATION. |
| description | String | O | O | Free text where to enter additional information about the specific device. |
| candidateConfigCount | Integer | O | O | Specifies how many candidate configurations the device has. For example, 0 or null. |
| category | String | O | O | Specifies what category the device belongs to. |

# 4 Device Management Operations

## 4.1 /scm-rest/device-repository/admin-states

### 4.1.1 GET

This operation fetches all valid admin states.

**Parameters:**

N/A

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/admin-states
```

**Response body example:**

```
[
  "IDLE",
  "INSTALLED",
  "UNKNOWN",
  "NOT INSTALLED",
  "INACTIVE",
  "ACTIVE"
]
```

## 4.2 /scm-rest/device-repository/device-categories

### 4.2.1 GET

This method gets all valid device categories.

**Parameters:**

This operation contains no parameters.

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/device-categories
```

**Response body example:**

```
[
    "OTHER",
    "CE",
```

```
    "Proxy",
    "Jump Server",
    "PE",
    "FIREWALL"
]
```

### 4.2.2 POST

This method adds new device category.

**Parameters:**

N/A

**Request example:**

The request body `{"category":"MY_CATEGORY"}` is sent to the URL `https://10.0.0.2:8383/scm-rest/device-repository/device-categories`.

**Response body example:**

```
{"category":"MY_CATEGORY"}
```

## 4.3 /scm-rest/device-repository/devices

This operation consists of the methods described below.

### 4.3.1 GET

Get a page of devices by filtering criteria.

**Parameters:**

The following parameters can be used when filtering:

Table 3 Device List Filter Parameters

| Name | Type | Occurrence | Description |
|---|---|---|---|
| pageIndex | query | Mandatory | The page to retrieve. |
| pageSize | query | Mandatory | Specifies how many device entries the page should contain. |

| Name | Type | Occurrence | Description |
|---|---|---|---|
| searchPattern | query | Optional | Devices with matching identifiers will be returned. |
| categories | query | Optional | Devices with matching categories will be returned. |
| device-type | query | Optional | Devices with matching Device Type will be returned. |
| protocols | query | Optional | Devices with matching protocol will be returned. |
| modes | query | Optional | Devices with matching modes will be returned. |
| adminStates | query | Optional | Devices with matching admin states will be returned. |

**Request body:**

N/A

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices?pageInde
x=1&pageSize=50&adminStates[]=ACTIVE&categories[]=CE&device-type
s[]=Junos&protocols[]=SSH_CLI&modes[]=CANDIDATE+STORE&searchPatt
ern=device
```

**Response body example:**

```
{
    "totalNrOfPages": 1,
    "currentPage": 1,
    "listItems": [
        {
            "masterIdentifier": "device1",
            "identifiers": [],
            "adminState": "ACTIVE",
            "modes": [
                "CANDIDATE STORE",
                "CONFIGURATION VALIDATION"
```

```
                ],
                "subModes": [
                    "UPDATE REPOSITORY"
                ],
                "deviceType": "Junos",
                "accessPoints": [
                    {
                        "protocol": "SSH_CLI",
                        "address": "10.0.0.3",
                        "parameters": {
                            "parameterKey1": "parameterValue1"
                        },
                        "credentials": {
                            "username": "admin",
                            "password": "admin"
                        }
                    }
                ],
                "info": {
                    "RECONCILIATION": "FAILED"
                    "RECONCILIATION_ERROR": "Unexpected error - detailed message fo
                },
                "description": "This is the description of the device",
                "candidateConfigCount": 0,
                "category": "CE"
            }
        ]
}
```


### 4.3.2 POST

This method adds a new device to the system.

**Parameters:**

N/A

**Request body example:**


```
{
    "masterIdentifier": "device1",
    "adminState": "ACTIVE",
    "deviceType": "Junos",
    "category": "CE",
    "modes": [
      "CANDIDATE STORE",
      "CONFIGURATION VALIDATION"
    ],
    "subModes": ["UPDATE REPOSITORY"],
    "description": "This is the description of the device",
```

```
    "accessPoints": [
        {
            "credentials": {
                "username": "admin",
                "password": "admin"
            },
            "parameters": {
                "interfaceType":"ge"
            },
            "protocolParameters": {},
            "address": "10.0.0.3:8000",
            "protocol": "SSH_CLI"
        }
    ]
}
```


**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices
```

**Response body example:**


```
{

    "identifiers": [],
    "modes": [
        "CANDIDATE STORE",
        "CONFIGURATION VALIDATION"
    ],
    "subModes": [
        "UPDATE REPOSITORY"
    ],
    "deviceType": "Junos",
    "accessPoints": [
        {
            "parameters": {
                "interfaceType":"ge"
            },
            "credentials": {
                "password": "admin",
                "username": "admin"
            },
            "protocolParameters": {
                "authenticationTimeout":"10000",
                "subsystem":"",
                "connectionTimeout":"60000",
                "version":"",
                "key":"",
                "responseTimeout":"20000"
            },
```

```
            "protocol": "SSH_CLI",
            "address": "10.0.0.3:8000"
        }
    ],
    "info": {},
    "masterIdentifier": "device1",
    "adminState": "ACTIVE",
    "description": "This is the description of the device",
    "candidateConfigCount": null,
    "category": "CE"
}
```

## 4.4 /scm-rest/device-repository/devices/{identifier}

This operation consists of the methods described below.

### 4.4.1 GET

This method gets the specific device by specifying deviceId.

**Parameters:**

This operation contains a path parameter which sends the request as a string Identifier.

Table 4 Device Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| identifier | path | Mandatory | The identity of the device to retrieve |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device1
```

**Response body example:**

```
{
    "identifiers": [],
    "modes": [
        "CANDIDATE STORE",
        "CONFIGURATION VALIDATION"
    ],
    "subModes": [
        "UPDATE REPOSITORY"
    ],
    "deviceType": "Junos",
    "accessPoints": [
```

```
        {
            "parameters": {
                "parameterKey1": "parameterValue1"
            },
            "credentials": {
                "password": "admin",
                "username": "admin"
            },
            "protocolParameters": {},
            "protocol": "SSH_CLI",
            "address": "10.0.0.3:8000"
        }
    ],
    "info": {
            "RECONCILIATION": "FINISHED"
        },
    "masterIdentifier": "device1",
    "adminState": "ACTIVE",
    "description": "This is the description of the device",
    "candidateConfigCount": null,
    "category": "CE"
}
```


For a more detailed description of the different parameters, see Section 3 on page 7.

### 4.4.2 PUT

This method updates information about a specific device with deviceId.

**Parameters:**

This operation contains a path parameter which sends the request as a string Identifier.

Table 5 Device Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| Identifier | path | Mandatory | The identity of the device to modify |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device1
```

**Request body example:**


```
{
    "identifiers": [],
```

```
    "adminState": "ACTIVE",
    "deviceType": "Junos",
    "category": "CE",
    "modes": ["CONFIGURATION VALIDATION"],
    "subModes": ["UPDATE REPOSITORY"],
    "description": "This is the description of the device",
    "accessPoints": [
        {
            "protocol": "SSH_CLI",
            "address": "10.0.0.3",
            "parameters": {},
            "credentials": {
                "username": "admin",
                "password": "admin"
            }
        }
    ]
}
```

**Response body:**

N/A

### 4.4.3 DELETE

This method removes a specific device and its feature instances if there are any.

**Parameters:**

This operation contains a path parameter which sends the request as a string Identifier.

Table 6 Device Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| Identifier | path | Mandatory | The identity of the device to delete. |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device1
```

**Response body:**

N/A

## 4.5 /scm-rest/device-repository/devices/{identifier}/config

This operation consists of the methods described below.

### 4.5.1 GET

This method retrieves all current device configurations of a specified device. The device configurations is listed in an XML file in the form of MOType and MOId pairs.

**Parameters:**

This operation contains a path parameter which sends the request as a string Identifier.

Table 7 Device Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| identifier | path | Mandatory | The identity of the device to retrieve |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device1
/config
```

**Request header:**

The key and value to be used in request header are:

Table 8 Key in Request Header

| Key | Value |
|---|---|
| Accept | `application/xml` |

**Request body:**

N/A

**Response body example:**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<deviceConfigs>
    <deviceConfig>
        <moType>{http://schemas.ericsson.com/scm/Firewall}/Firewall</moType>
        <moId>
            <ruleNumber>101</ruleNumber>
        </moId>
    </deviceConfig>
    <deviceConfig>
```

```
        <moType>{http://schemas.ericsson.com/scm/EthernetInterface/}EthernetInt
        <moId>
            <interfaceType>FastEthernet</interfaceType>
            <interfaceNumber>1/2.10</interfaceNumber>
        </moId>
    </deviceConfig>
</deviceConfigs>
```

### 4.5.2 POST

This method retrieves a specific device configuration of a device by using an MOType and MOId pair.

**Parameters:**

This operation contains a path parameter which sends the request as a string Identifier.

Table 9 Device Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| identifier | path | Mandatory | The identity of the device to retrieve |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device1
/config
```

**Request header:**

The keys and values to be used in request header are:

Table 10 Keys in Request Header

| Key | Value |
|---|---|
| Accept | `application/xml` |
| Content-Type | `application/xml` |

**Request body example:**

```
<deviceConfig>
    <moType>{http://schemas.ericsson.com/scm/EthernetInterface/}EthernetInterfa
    <moId>
        <interfaceType>FastEthernet</interfaceType>
        <interfaceNumber>1/2.10</interfaceNumber>
    </moId>
</deviceConfig>
```

**Response body example:**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<deviceConfig>
    <moType>{http://schemas.ericsson.com/scm/EthernetInterface/}EthernetInte
    <moId>
        <interfaceType>FastEthernet</interfaceType>
        <interfaceNumber>1/2.10</interfaceNumber>
    </moId>
    <deviceId>device1</deviceId>
    <moAttributes>
        <eth:CreateEthernetInterface xmlns:cai3="http://schemas.ericsson.com
             xmlns:eth="http://schemas.ericsson.com/scm/EthernetInterface/"
             xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
             xmlns:types="http://schemas.ericsson.com/scm/Types/"
             interfaceNumber="1/2.10" interfaceType="FastEthernet">
            <eth:interfaceType>FastEthernet</eth:interfaceType>
            <eth:interfaceNumber>1/2.10</eth:interfaceNumber>
        </eth:CreateEthernetInterface>
    </moAttributes>
</deviceConfig>
```

## 4.6 /scm-rest/device-repository/devices/{identifier}/config-sync

This operation consists of the methods described below.

### 4.6.1 POST

This method triggers a device reconciliation to align the Device Repository with the provisioned devices.

**Parameters:**

This following parameters can be used when reconciling a device.

Table 11 Device Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| Identifier | path | Mandatory | The identity of the device to reconcile. |
| syncType | string | Mandatory | |
| UPDATE DEVICE | - | - | Specifies that the Device should be aligned with the Repository |

| Name | Type | Occurrence | Description |
|---|---|---|---|
| UPDATE REPOSITORY | - | - | Specifies that the Repository should be aligned with the Device. |
| cached | boolean | Mandatory | Specifies that the system should rebuild the cache of the network data. Must be “false”. |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device1/config-sync
```

**Request body example:**

```
{
    "syncType": "UPDATE DEVICE",
    "cached": "false"
}
```

**Response body**

N/A

## 4.7 /scm-rest/device-repository/devices/{identifier}/force

### 4.7.1 DELETE

This method removes a specific device and its features instances even if the features belong to a service.

**Parameters:**

This operation contains a path parameter which sends the request as a string Identifier.

Table 12 Device Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| identifier | path | Mandatory | The identity of the device to delete. |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device1/force
```

**Response body:**

N/A

## 4.8 /scm-rest/device-repository/devices/{identifier}/config/backups/{timestamp}/restore

### 4.8.1 POST

This method restores an entire device configuration to a previous created backup.

**Parameters:**

This following parameters can be used when restoring a device.

Table 13 Device Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| Identifier | path | Mandatory | The identity of the device to restore. |
| timestamp | path | Mandatory | The timestamp of the backup to restore, using format of Epoch time in milliseconds. |
| includeDelta | boolean | Optional | Should delta information from Device Repository be restored. Default value is "false".<br>• When it is set to "true", all configuration updates, which are done by using Resource Configuration since the backup was created, are to be restored. This is the recommended settings in most case, because this will restore the most recent information that is known about the device configuration.<br>• If device configurations affecting the modeled features have been done manually on the device, it is recommended to set to "false". |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device1/config/backups/1509016427361/restore
```

**Request body example:**

```
{
    "includeDelta": "true"
}
```

**Response body**

N/A

## 4.9 /scm-rest/device-repository/devices/{identifier}/config/backups

### 4.9.1 POST

This method creates a new backup of the entire device configuration for a device.

**Parameters:**

This following parameters can be used when backing up a device.

Table 14 Device Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| Identifier | path | Mandatory | The identity of the device to restore. |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device1/
config/backups
```

**Request body example:**

N/A

**Response body**

N/A

### 4.9.2 GET

This method retrieve all device configuration backups information for a device. Information includes when a backup is created for which device and the backup status. The content of the backups is not included.

**Parameters:**

This following parameters can be used when backing up a device.

Table 15 Device Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| Identifier | path | Mandatory | The identity of the device to restore. |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device1/
config/backups
```

**Request body example:**

N/A

**Response body**

```
[
    {
        "status": "FAILED",
        "timestamp": 1509016245370
    },
    {
        "status": "SUCCESSFUL",
        "timestamp": 1509016427361
    }
]
```

## 4.10 /scm-rest/device-repository/devices/config/backups

### 4.10.1 POST

This method creates a new backup of the entire device configuration for a list of devices. The method is asynchronous and can be monitored by the method described in Section 4.11 on page 26.

**Parameters:**

This method takes a list of devices in JSON format as input. The devices list is contained in the request body.

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/config/
backups
```

**Request body example:**

```
[
```

```
    "device1",
    "device2",
    "device3"
]
```

**Response body**

```
{
          "jobId": "123e4567-e89b-12d3-a456-426655440000"
}
```

## 4.11 /scm-rest/device-repository/jobs/{jobId}

### 4.11.1 GET

This method is used to get the status of the backup method described in Section 4.10 on page 25.

**Parameters:**

This following parameters can be used:

Table 16 Job Status Parameters

| Name | Type | Occurrence | Description |
|---|---|---|---|
| jobId | path | Mandatory | Job identity of whose status information to return. |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/jobs/123e4567-e89
b-12d3-a456-426655440000
```

**Request body example:**

N/A

**Response body examples:**

```
{
    "id": "123e4567-e89b-12d3-a456-426655440000",
    "status": "STARTED",
    "jobDetails": {"action": "BACKUP_DEVICES", "deviceNames ":["device1","devic
}
```

Example 3 The task is started but has not finished yet.

```
{
    "id": "123e4567-e89b-12d3-a456-426655440000",
    "status": "FINISHED",
    "result": "SUCCESSFUL",
    "resultDetails": [],
    "jobDetails": {"action": "BACKUP_DEVICES", "deviceNames ":["device1","de
}
```


Example 4 The task is finished successfully.


```
{
    "id": "123e4567-e89b-12d3-a456-426655440000",
    "status": "FINISHED",
    "result": "FAILED",
    "resultDetails": [
                {"deviceName": "device2",
                "errorMessage": "Failed to backup:  Reason: No downstream co
    "jobDetails": {"action": "BACKUP_DEVICES", "deviceNames ":["device1","de
}
```


Example 5 The task is finished with a failure.

## 4.12 /scm-rest/device-repository/devices/{identifier}/config/backups/{timestamp}

### 4.12.1 GET

This method retrieve the entire device configuration that was backed up on a particular time for a device.

**Parameters:**

This following parameters can be used:

Table 17 Device Parameters

| Name | Type | Occurrence | Description |
|---|---|---|---|
| Identifier | path | Mandatory | The identity of the device to restore. |
| timestamp | path | Mandatory | The timestamp of the backup to restore, using format of Epoch time in milliseconds. |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/device-repository/devices/device
1/config/backups/1509016427361
```

**Request body example:**

N/A

**Response body**


```
{
    "content": "!\nversion 12.4\nservice timestamps debug datetime msec\nservic
    "status": "SUCCESSFUL",
    "deviceId": "123e4567-e89b-12d3-a456-426655440000",
    "timestamp": 1509016427361
}
```

# 5 Feature Model Operations

## 5.1 /scm-rest/modelcatalog/import

### 5.1.1 POST

This operation imports feature models from an zip file that is generated through an export operation.

This operation is asynchronous. Use operation described in Section 5.3 on page 30 to check status.

**Parameters:**

N/A

**Request body:**

The Content-Type in the request is `multipart/form-data`, where the key name to be used is `file`, for example:

```
name="file"; filename="models.zip"
```

The zip file to be imported must be of a format that is generated through an export operation.

**Request example:**

```
https://10.0.0.2:8383/scm-rest/modelcatalog/import
```

**Response body examples**

```
{
          "jobId": "123e4567-e89b-12d3-a456-426655440000"
}
```

## 5.2 /scm-rest/modelcatalog/export

### 5.2.1 POST

This operation exports feature models to a zip file that can be used for import in Section 5.1 on page 29.

**Parameters:**

N/A

**Request example:**

```
https://10.0.0.2:8383/scm-rest/modelcatalog/export
```

**Request body examples:**

```
{
    "type": "ALL"
 }
```

Example 6 Export All Feature Models

```
{
    "models": ["model1","model2"],
    "type": "SELECTED"
 }
```

Example 7 Export Specific Feature Models

**Response body:**

Zipped data with `Content-Type = application/zip`

## 5.3 /scm-rest/modelcatalog/jobs/{jobId}

### 5.3.1 GET

This method is used to get the status of the import operation that is described in Section 5.1 on page 29.

**Parameters:**

This following parameters can be used:

Table 18 Job Status Parameters

| Name | Type | Occurrence | Description |
|---|---|---|---|
| jobId | path | Mandatory | Job identity of whose status information to return. |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/modelcatalog/jobs/123e4567-e89b-12d
3-a456-426655440000
```

**Request body example:**

N/A

**Response body examples:**

```
{
    "id": "123e4567-e89b-12d3-a456-426655440000",
    "status": "STARTED",
    "jobDetails": "IMPORT_MODELS"
}
```

Example 8 The task is started but has not finished yet.

```
{
    "id": "123e4567-e89b-12d3-a456-426655440000",
    "status": "FINISHED",
    "jobDetails": "IMPORT_MODELS",
    "result": "SUCCESSFUL",
    "resultDetails": [
        "Imported 2 interface models",
        "Imported 2 data models"
    ]
}
```

Example 9 The task is finished successfully.

```
{
    "id": "123e4567-e89b-12d3-a456-426655440000",
    "status": "FINISHED",
    "jobDetails": "IMPORT_MODELS",
    "result": "FAILED",
    "resultDetails": ["Failed to import models. No items imported. The file
}
```

Example 10 The task is finished with a failure.

## 5.4 /scm-rest/modelcatalog/models/import/all

### 5.4.1 POST

This operation imports feature models from an zip file that is generated through an export operation.

**Parameters:**

N/A

**Request body:**

The Content-Type in the request is `multipart/form-data`, where the key name to be used is `file`, for example:

```
name="file"; filename="models.zip"
```

The zip file to be imported must be of a format that is generated through an export operation.

**Request example:**

```
https://10.0.0.2:8383/scm-rest/modelcatalog/models/import/all
```

**Response body:**

N/A

## 5.5 /scm-rest/modelcatalog/models/export/{model name}

### 5.5.1 GET

This operation exports one feature model to a zip file that can be used for import.

**Parameters:**

This operation contains a path parameter which sends the request as a string Identifier to identify the model name.

Table 19 Model Parameter

| Name | Type | Occurrence | Description |
|---|---|---|---|
| model name | path | Mandatory | The identity of the feature model to export. |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/modelcatalog/models/export/EthernetInterface
```

**Response body:**

N/A

## 5.6 /scm-rest/modelcatalog/models/export/all

### 5.6.1 GET

This operation exports all feature models to a zip file that can be used for import.

**Parameters:**

N/A

**Request example:**

```
https://10.0.0.2:8383/scm-rest/modelcatalog/models/export/all
```

**Response body:**

N/A

# 6 Template Management Operations

## 6.1 /scm-rest/template-management/import

### 6.1.1 POST

This operation imports vendor templates from an zip file that is generated through an export operation.

This operation is asynchronous. Use operation described in Section 6.3 on page 36 to check status.

**Parameters:**

N/A

**Request body:**

The Content-Type in the request is `multipart/form-data`, where the key name to be used is `file`, for example:

```
name="file"; filename="templates.zip"
```

The zip file to be imported must be of a format that is generated through an export operation.

**Request example:**

```
https://10.0.0.2:8383/scm-rest/template-management/import
```

**Response body examples**

```
{
          "jobId": "123e4567-e89b-12d3-a456-426655440000"
}
```

## 6.2 /scm-rest/template-management/export

### 6.2.1 POST

This operation exports templates to a zip file that can be used for import in Section 6.1 on page 35. The zip file also includes relevant template definitions (for the XML template) or transformations (for the YANG template).

**Parameters:**

N/A

**Request example:**

```
https://10.0.0.2:8383/scm-rest/template-management/export
```

**Request body:**

The following parameters are used:

Table 20 Template Export Parameters

| Name | Values | Occurrence | Description |
|---|---|---|---|
| type | `XML`, `YANG`, or `ALL` | Mandatory | Specifies which type(s) of templates to be exported, XML, YANG, or both. |
| templateNames | A string list. | Optional | Specifies one or more names of the templates to be exported. Template names are separated by ‘,’. |

**Request body examples:**

```
{
    "type": "ALL"
 }
```

Example 11 Export All Templates

```
{
    "type": "XML"
 }
```

Example 12 Export All XML Templates

```
{
    "templateNames": ["myTemplate1", "myTemplate2"],
    "type": "YANG"
 }
```

Example 13 Export Specified YANG templates

**Response body:**

Zipped data with `Content-Type = application/zip`

# 6.3 /scm-rest/template-management/jobs/{jobId}

## 6.3.1 GET

This method is used to get the status of the import operation that is described in Section 6.1 on page 35.

**Parameters:**

This following parameters can be used:

Table 21 Job Status Parameters

| Name | Type | Occurrence | Description |
|---|---|---|---|
| jobId | path | Mandatory | Job identity of whose status information to return. |

**Request example:**

```
https://10.0.0.2:8383/scm-rest/template-management/jobs/123e4567-e89b-12d3-a456-426655440000
```

**Request body example:**

N/A

**Response body examples:**

```
{
    "id": "123e4567-e89b-12d3-a456-426655440000",
    "status": "STARTED",
    "jobDetails": "IMPORT_TEMPLATES"
}
```

Example 14 The task is started but has not finished yet.

```
{
    "id": "123e4567-e89b-12d3-a456-426655440000",
    "status": "FINISHED",
    "jobDetails": "IMPORT_TEMPLATES",
    "result": "SUCCESSFUL",
    "resultDetails": "Imported 10 templates"
}
```

Example 15 The task is finished successfully.

```
{
    "id": "123e4567-e89b-12d3-a456-426655440000",
    "status": "FINISHED",
    "jobDetails": "IMPORT_TEMPLATES",
    "result": "FAILED",
    "resultDetails": "Failed to import templates. No template items imported. T
}
```

Example 16 The task is finished with a failure.

## 6.4 /scm-rest/template-management/device-types/{device-type}

### 6.4.1 GET

This method is used to retrieve all templates with meta-data, features and operations for the specified device-type.

**Parameters:**

The following parameters are used:

Table 22 Device Type Parameters

| Name | Type | Occurrence | Description |
|---|---|---|---|
| device-type | path | Mandatory | The name of the Device Type. |

**Response body example:**

```
[
    {
        "name": "Template1",
        "type": "XML",
        "protocol": "CLI",
        "features": [
            {
                "name": "Feature1",
                "operations": [
                    "GETALL",
                    "DELETE",
                    "SET",
                    "CREATE",
                    "GET"
                ]
            }
    }
]
```

### 6.4.2 PUT

This method is used to update an existing Device Type to template, feature, operation mapping.

When using this method, the following conditions must be met:

— The Device Type must exist in the system.

— The template(s), feature(s) and operation(s) that will be mapped to the Device Type must exist in the system.

— No overlapping feature operations, within or between templates, are present in the mapping request.

When the method is executed, the included template, feature operations will replace all prior existing mappings for the specified Device Type.

**Parameters:**

The following parameters are used:

Table 23 Device Type Parameters

| Name | Type | Occurrence | Description |
|---|---|---|---|
| device-type | path | Mandatory | The name of the Device Type to update. |

**Request body:**

The following parameters are used:

Table 24 Update Device Type Request Parameters

| Name | | | Type | Occurrence | Description |
|---|---|---|---|---|---|
| templates | | | Array | Mandatory | An array of templates. |
| | name | | String | Mandatory | The name of the template to map. |
| | features | | Array | Mandatory | An array of features. |
| | | name | String | Mandatory | The name of the feature to map |
| | | operations | Array | Mandatory | An array of operations (CREATE, DELETE, GET, SET, GETALL) |

**Request example:**

```
{
   "templates":[
      {
         "name":"Template1",
         "features":[
```

```
            {
               "name":"Feature1",
               "operations":[
                   "CREATE",
                   "DELETE"
               ]
            }
         ]
      }
   ]
}
```

## 6.5 /scm-rest/template-management/device-types

### 6.5.1 GET

This method is used to retrieve all Device Type names.

**Parameters:**

N/A

**Response body example:**

```
[
  "deviceType1",
  "deviceType2",
]
```

### 6.5.2 POST

This method is used to create a new Device Type to template, feature, operation mapping.

When using this method, the following conditions must be met:

— The Device Type must NOT exist in the system.

— The template(s), feature(s) and operation(s) that will be mapped to the Device Type must exist in the system.

— No overlapping feature operations, within or between templates, are present in the mapping request.

**Request body:**

The following parameters are used:

Table 25 Create Device Type Request Parameters

| Name | Type | Occurrence | Description |
|---|---|---|---|
| deviceType | String | Mandatory | The name of the Device Type to create. |
| templates | Array | Mandatory | An array of templates. |
| name | String | Mandatory | The name of the template to map |
| features | Array | Mandatory | An array of features. |
| name | String | Mandatory | The name of the feature to map. |
| operations | Array | Mandatory | An array of operations (CREATE, DELETE, GET, SET, GETALL) |

**Request example:**

```
{
   "deviceType":"testDeviceType1",
   "templates":[
      {
         "name":"Template1",
         "features":[
            {
               "name":"Feature1",
               "operations":[
                   "CREATE",
                   "DELETE"
               ]
            }
         ]
      }
   ]
}
```

# 7 Faults or Errors

A REST error message consists of an error code and a message describing the error.

The following table covers Resource Configuration REST error codes. They can appear in any REST response.

Table 26 Resource Configuration REST Error Codes

| Error Code | Error Code Description |
|---|---|
| 400 | Bad request, the request was invalid. |
| 401 | Unauthorized, due to authentication header is missing, or username or password was invalid. |
| 404 | Not found, the requested resource was not found on the server. |
| 500 | Internal server error, something went wrong when processing the request. |

The following is an example of a REST error message:

```
{
    "errorCode": 404,
    "description": "Not found."
}
```

Example 17 REST Error Message

# Reference List

**Ericsson Documents**

[1] Function Specification Resource Configuration, 19/155 17-CSH 109 628 Uen

[2] User Guide for Resource Configuration, 11/1553-CSH 109 628 Uen

[3] Library Overview, 18/1553-CSH 109 628 Uen