

Policy Studio

Ericsson Service-Aware Policy Controller

DESCRIPTION

Copyright

© Ericsson España, S.A. 2017. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.

Abstract

This document introduces the SAPC Policy Studio and provides installation, and operation and maintenance (O&M) information.



Contents

1	Introduction	1
1.1	Document Purpose and Scope	1
1.2	Target Groups	1
1.2.1	Prerequisite Knowledge	1
2	Overview	1
3	Getting Started	3
3.1	System Prerequisites	3
3.2	Software Prerequisites	3
3.2.1	Node.js	3
3.2.2	MongoDB	4
3.3	Downloading the Software and Preparing the Installation	4
3.4	Installation	5
3.5	User Roles	7
3.5.1	Administrator	7
3.5.2	Operator	8
3.5.3	Viewer	8
3.6	Connectivity and Security Management	8
3.6.1	Certificate between the Policy Studio and the SAPC	9
3.6.2	Certificate between the web browser and the Policy Studio	9
4	Working with The Policy Studio	10
4.1	Accessing The Policy Studio	10
4.2	Logon	10
4.2.1	System Administrator Logon	11
4.2.2	Operator and Viewer Logon	13
4.2.3	Improving Policy Studio Performance with NGINX	13
5	Operation And Maintenance	17
5.1	Backup and Data Restore	17
5.1.1	Backup Procedure	17
5.1.2	Restore Procedure	18
5.2	Logging Management	18
5.3	Troubleshooting	19
5.3.1	Administrator Password Recovery	19
5.3.2	Restart the Policy Studio	19
5.4	Upgrade	19
5.5	Rollback	20



6	SAPC Functionality Not Supported	20
6.1	REST resources	20
6.2	Functions	20
6.3	Contents	20
6.4	Dataplans	21
6.5	Subscribers	21
6.6	Rules	21
6.7	Policy Tags	21
6.8	Condition Formula Expressions	22
6.9	Condition Formula Operators	22
7	Limitations	22
7.1	Dataplans	23
7.2	Profiles	23
7.3	Global Policies	23
7.4	Export Functionality	23
7.5	Import Functionality	23



1 Introduction

1.1 Document Purpose and Scope

The SAPC Policy Studio is a web-based user interface (UI) to manage provisioning information for a single SAPC node or a group of SAPC nodes, in an intuitive and easy way. It is the main client of the SAPC Provisioning REST API.

Attention!

The Policy Studio is not installed in the SAPC cluster.

The Policy Studio runs in a separate server. The Policy Studio installation instructions are detailed in Section 3.4 on page 5.

1.2 Target Groups

The intended audience of this document is:

- Solution Architects, Service Engineers, Support Engineers, SAPC System Administrators, and any other SAPC Operations personnel.

1.2.1 Prerequisite Knowledge

Users of this document must be familiar with the following topics:

- SAPC Provisioning REST API.
- General networking knowledge.
- Certificate Management.

2 Overview

The Policy Studio is an intuitive web-based user interface (UI), that acts as the main client of the SAPC Provisioning REST API. It provides the following key features:



- Handle draft or committed element versions.
- Guided provisioning flows.
- Easy rule management through a visual condition builder.
- Organize provisioning data into collections.
- Import and Export provisioning information.
- Multiple SAPC node management.
- User accounts management.
- Configuration Guide to help define basic use cases.

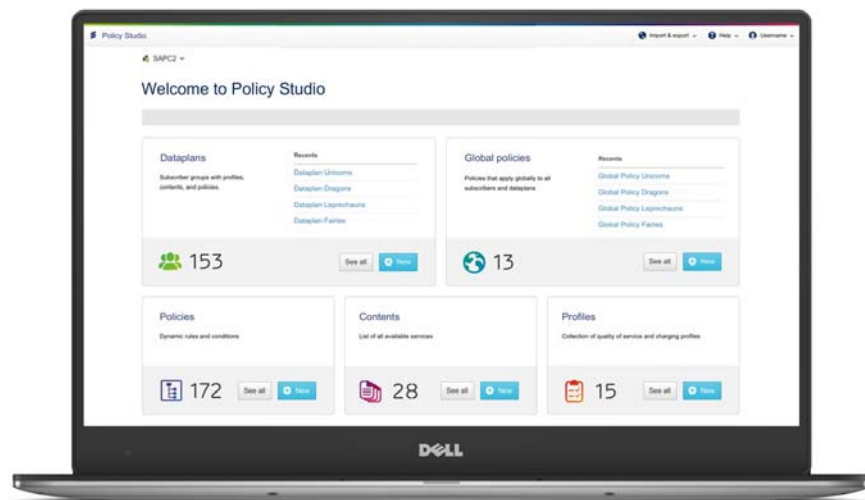



Figure 1 Policy Studio Home Page

The Policy Studio integrates a User Guide directly on its interface. To obtain more information on each of the Policy Studio key features, click the  **Help** icon available at the top-right corner of any page.



3 Getting Started

3.1 System Prerequisites

The minimum requirements the server must fulfill to support the Policy Studio, are listed in Table 1. The Policy Studio compatible web browsers are listed in Table 2.

Table 1 Policy Studio Host Server Minimum Requirements

System Requirements	
Hardware	CPU: 2 GHz, 2 cores. RAM: 2 GB Hard Disk Space: 20 GB
Operating System	Linux, Windows, macOS

Table 2 Web Browser Compatibility Minimum Requirements

System Requirements	
Browser Compatibility	Chrome 55, Firefox 50

3.2 Software Prerequisites

The minimum software prerequisites to install the Policy Studio, are listed in Table 3:

Table 3 Policy Studio Software Minimum Prerequisites

Software Prerequisites	
3PP	Node.js 6.9, MongoDB 3.x, NGINX 1.12

3.2.1 Node.js

1. Download Node.js® from <https://nodejs.org/en/download/> and install the package. Alternatively use a package manager to install Node.js® on the server specific system <https://nodejs.org/en/download/package-manager/>. The minimum software version supported is detailed in Table 3.
2. Verify that node has been installed properly. From the server CLI execute:

`Server:# node -v`
3. Verify that npm has been installed properly. From the server CLI execute:



```
Server:# npm -v
```

3.2.2 MongoDB

1. Download MongoDB from <https://www.mongodb.com/download-center?jmp=nav#community>. The minimum software version supported is detailed in Table 3.
2. Follow the installation guidelines for your specific system at: <https://docs.mongodb.com/manual/administration/install-community/>

3.3 Downloading the Software and Preparing the Installation

To prepare the installation of the SAPC Policy Studio, do the following:

1. Access [Software Gateway](#) and download the SAPC Policy Studio software available inside the SAPC Software Gateway Ticket Number.

The downloaded file is named `policy_studio_<version>` and has ZIP archive file format.

2. Access the server where the Policy Studio is to be installed and create the installation directory, for example `/home/PolicyStudio/` :

```
Server:# mkdir -p /home/PolicyStudio/
```

```
Server:# cd /home/PolicyStudio/
```

3. Copy the `Policy_Studio_<version>.zip` file into the Policy Studio server, and decompress it:

```
Server:# scp <user>@<host>:<SWGW_DownloadDirectory>/Policy_Studio_<version>.zip .
```

```
Server:# unzip Policy_Studio_<version>.zip
```

The folder contains the following files and subfolders:

- **backup.js**
Database Backup script
- **install.js**
Installation script.
- **node_modules**
npm third party dependencies
- **npm-shrinkwrap.json**
npm package lock file. Defines exact module versions for reproducible deployments
- **package.json**



npm package installation information.

- **public**
Statically served files for the policy studio.
- **README.md**
Summary of the system prerequisites and installation instructions.
- **restore.js**
Database Restore script
- **run.js**
Sever initiation script.
- **server**
Files running the API.
- **server.config.example.json**
Server configuration example file.
- **softwareVersion.json**
software version and product information.

3.4 Installation

To install the SAPC Policy Studio, do the following:

1. Start the MongoDB daemon:

```
Server:# mongod --port 27020
```

MongoDB requires access to a folder to store the database files. By default it tries to access /data/db. To specify a different database directory, use the option `--dbpath`.

2. Add a user to the database:

```
mongo <mongodb url>:<mongodb port>/<mongodb database>  
--eval "db.createUser({user:'<mongodb user>', pwd:'<mongodb  
password>', roles: ['read']})"
```

Typically: `Server:# mongo 127.0.0.1:27020/test --eval "db.createUser({user:'admin', pwd:'Admin-1234', roles: ['read']})"`

The database user and password is needed to connect from Policy Studio to MongoDB, as detailed in Step 4.

3. Copy the `server.config.example.json` to `server.config.json`:

```
Server:# cp server.config.example.json server.config.json
```



4. Edit `server.config.json` file to adapt the defined configuration variables to the deployment needs:
 - **MONGODB_USERNAME**- [admin]
MongoDB username.
 - **MONGODB_PASSWORD**- [Admin-1234]
MongoDB password.
 - **MONGODB_URL**- [localhost]
MongoDB hostname.
 - **MONGODB_PORT**- [27020]
MongoDB port number.
 - **DISABLE_SSL**- [true]
False to enable SSL and connect the browser and the Policy Studio with HTTPS.
 - **KEY_FILEPATH**- [./server.key]
Path to the Policy Studio SSL certificate key (used ONLY if **DISABLE_SSL** variable is set to false. Ignored otherwise).
 - **CERT_FILEPATH**- [./server.crt]
Path to the Policy Studio SSL certificate (used ONLY if **DISABLE_SSL** variable is set to false. Ignored otherwise).
 - **LOG_MAX_FILE_SIZE_BYTES**- [5242880]
Maximum file size for a log file, if it reaches the size a new file is created.
 - **LOG_MAX_FILES**- [100]
Maximum number of files for a specific log type. Once reached, newest log file overwrites oldest log file.
 - **LOG_FILE_ENABLE_ERROR**- [true]
Enable logging to file for error logs.
 - **LOG_FILE_ENABLE_AUTH**- [true]
Enable logging to file for authentication logs.
 - **LOG_FILE_ENABLE_SAPC**- [true]
Enable logging to file for SAPC connections (request options, request body, and user).
 - **PORT**- [8585] for HTTP or [8686] for HTTPS.
Policy Studio listening port number.

In cloud deployments, to dynamically assign the server port using environment variable "PORT", use instead :**"processMap": { "PORT": "env.PORT" }**.
5. Access the folder that contains the uncompressed Policy Studio software and start the Policy Studio installation script:



```
Server:# npm install
```

As part of the installation, the process installs the npm dependencies defined in `package.json` from the `node_modules` folder.

Attention!

The `npm install` installation step requires user interaction. The process prompts the user to decide if database seeding is needed.

Caution!

Reseed the database only if a fresh installation, a rollback or an application reset is desired, as it deletes any previous data stored on the Policy Studio.

The installation process adds an extra entry in `server.config.json` with variable `API_TOKEN_SECRET_KEY`. The value of this variable acts as the key to sign the token used for the user verification. It is used in the encryption/decryption of the keys for the user logon. If it is changed, all ongoing operations are stopped and users are forced to perform a new logon.

The installation process must finish with the positive status report "Installation is complete" and the list of the Policy Studio installed dependencies including name and version.

6. Start the Policy Studio server:

```
Server:# node run.js
```

3.5 User Roles

There are three types of user roles to access the Policy Studio: administrator, operator and viewer.

3.5.1 Administrator

The administrator user account is created at installation time and it cannot be deleted. The administrator user is able to perform the following operations:

- Manage operator/viewer user accounts: create, modify, and delete accounts.
- Manage connectivity properties of the SAPC nodes the Policy Studio is interacting with: add, modify, and delete nodes.



- Edit the 'Legal notice' text.
- Define password change frequency.

The Policy Studio supports a single administrator account.

3.5.2 Operator

The operator accounts are designed for users that interact with the Policy Studio regularly. The operators have access to the complete Policy Studio functionality except for the **Admin Tools** panel.

3.5.3 Viewer

The viewer accounts are designed for users that interact with the Policy Studio in read-only mode. The viewers have read-only access to all Policy Studio objects. Actions edit, delete, import or export are disabled.

3.6 Connectivity and Security Management

The web browser connects with the Policy Studio over HTTP or HTTPS, as described on interface (1) of Figure 2. The use HTTP or HTTPS is configured with variable `DISABLE_SSL` as described in Step 4 . In case the operator connects over HTTPS, a certificate is required.

The Policy Studio connects with the SAPCs over HTTPS using the Provisioning REST interface as described in interface (2) of Figure 2.

Do!

To establish a communication between the Policy Studio and each of the SAPCs, it is mandatory to renew the certificate that comes in the initial configuration of each SAPC. Otherwise, the HTTPS connection is not established.

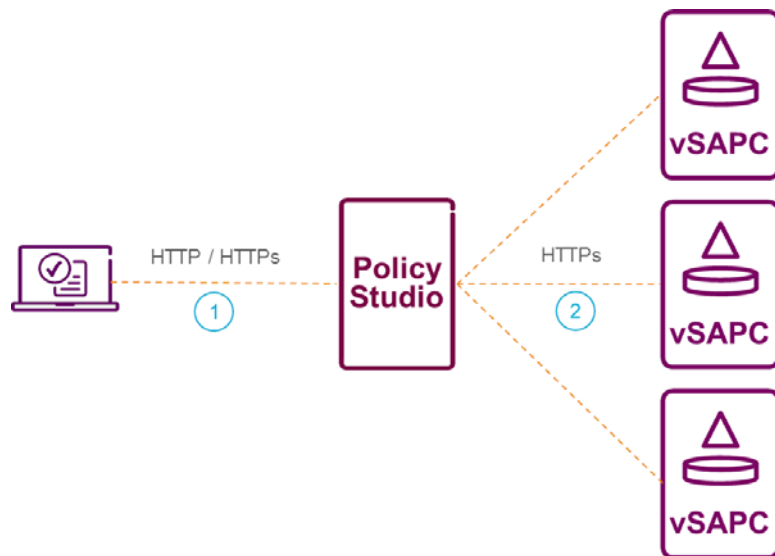


Figure 2 Policy Studio Interfaces

3.6.1 Certificate between the Policy Studio and the SAPC

The protocol HTTPS over TLS is used to secure the protection of the privacy and integrity of the data managed by the Policy Studio and exchanged with the SAPC.

If the SAPC REST service is configured with a certificate obtained from an external trusted Certification Authority (recommended option), no additional configuration is required in the Policy Studio. If a self-signed certificate is configured in the SAPC (intended only for demonstrations or testing environments), the public certificate is required when the SAPC node is added to the Policy Studio (Step 3).

Generating a Self-signed Certificate and Installing It in the SAPC

Refer to the [Security Management Guide](#).

Configuring the Self-signed Certificate in the Policy Studio

The public certificate is required when the SAPC node is added to the Policy Studio, as described in Step 3.

3.6.2 Certificate between the web browser and the Policy Studio

If the Policy Studio connects with the web browser using HTTPS, a certificate is required. It is recommended to use a certificate obtained from an external trusted Certification Authority. A self-signed certificate can be used but intended only for demonstrations or testing environments.



Generating a Self-signed Certificate

The [Security Management Guide](#) provides an example on how to generate a self-signed certificate for the SAPC. This document can be used as a guide to generate an RSA private key and self-signed public certificate for the Policy Studio, using the Policy Studio server hostname or IP in the process.

Adding the Certificate to The Policy Studio

Access the Policy Studio configuration file `server.config.json` and edit:

1. Enable HTTPs setting variable `"DISABLE_SSL": false`
2. Define relative paths to the private key and the public certificate files:

```
"KEY_FILEPATH":  "./server.key"  
"CERT_FILEPATH":  "./server.crt",
```
3. Save the changes and. If the Policy Studio is already running, restart it.

4 Working with The Policy Studio

4.1 Accessing The Policy Studio

To access the Policy Studio, do the following:

1. Open a compatible web browser.
2. Access the Policy Studio server hostname or IP on the required port (the URL hostname could, for example, be localhost).

— `http://<hostname/IP>:<PORT>`

Or for secure connections:

— `https://<hostname/IP>:<PORT>`, and accept the certificate if necessary.

4.2 Logon

The Policy Studio logon and authentication process has the following restrictions:

- On user first-time access, or if the password has expired, users are prompted to reset their password.



- It is performed every 24 hours period for security.
- Passwords must be 8–15 characters long and contain at least one lowercase, one uppercase, one number, and one punctuation mark.
- If a user forgets the access password, the administrator must be contacted for a temporary password reset.

Attention!

If it is the first time logging in to the Policy Studio, it must be done as administrator. The administrator user account and its predefined first access password is created in the installation process.

4.2.1 System Administrator Logon

Access the Policy Studio and at the logon screen:

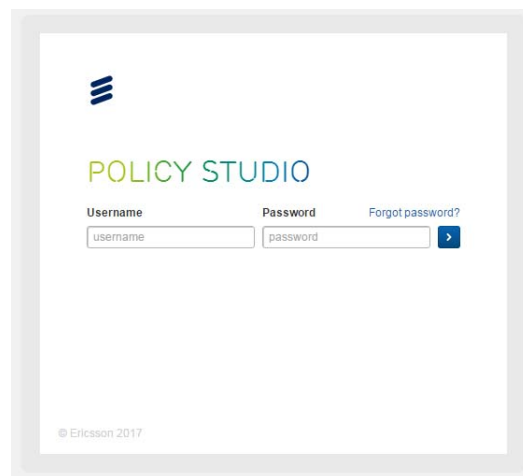


Figure 3 Logon Screen

1. Fill administrator **Username** and **Password**.
First-time access administrator credentials are:
 - Username: **Admin**
 - Password: **Admin-1234**
2. The administrator is invited to change the administrator password in the first Policy Studio access.
3. Define at least one SAPC node for the Policy Studio to connect to. From the **Admin tools** section:



- Click the **New** button from the **SAPC nodes** section.
- Fill in the form shown in Figure 5 with required connectivity details. If the SAPC is configured with a self-signed certificate, include the public certificate file (my_cert.crt described in *Security Management Guide* example) on the respective form field. If the SAPC is configured with CA Trusted Certificate, leave the self-signed certificate field empty.

Once the information is filled, press the **Finish** button.

Note: The Provisioning username is **sapcprov** (Administrator provided to perform REST provisioning operations)

- Repeat the operation for the rest of the SAPC nodes.

4. From the **Admin tools** page, define the user accounts.

Do!

The logic of the Policy Studio directs the administrator to the **New Node** section on the **Admin tools** page on the first-time access, as shown in Figure 5. To be able to use the Policy Studio for provisioning operations, it is mandatory for the administrator to define at least one SAPC node.

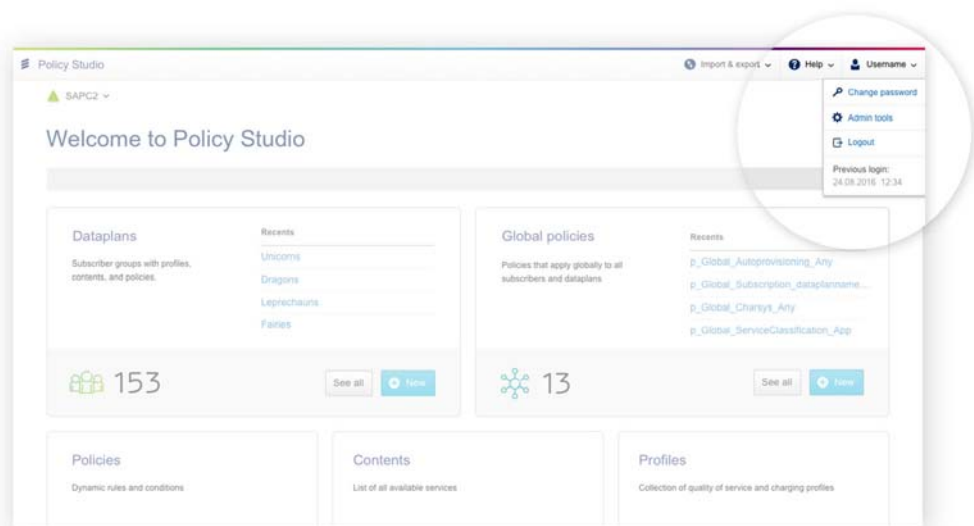


Figure 4 Admin Tools Section



The 'New Node' form contains the following fields:

- Node name ***: Text input field.
- Hostname ***: Text input field.
- Port ***: Text input field with '8443' pre-filled.
- Processing username ***: Text input field.
- Processing password ***: Text input field.
- Self-signed certificate**: Radio button group with 'Choose File' and 'No file chosen' options.

Buttons at the bottom right: Cancel, OK, Finish.

Figure 5 New SAPC Node Form

4.2.2 Operator and Viewer Logon

Access the Policy Studio and at the logon screen:

1. Fill **Username** and **Password** .
2. The operator/viewer is invited to change the password provided by the administrator in the first Policy Studio access.
3. Select the SAPC node to connect to, from the list of available SAPC nodes, as shown in Figure 6.



Figure 6 First-time Operator Access SAPC Node Selection

4.2.3 Improving Policy Studio Performance with NGINX

NGINX is designed to be a proxy host to cache static content and improve further the performance of Node.js. The installation of NGINX is recommended



to reduce the static content response time and improve the overall Policy Studio user experience.

4.2.3.1 Downloading and Installing NGINX

To download and install NGINX, follow the procedure described in: <https://www.nginx.com/resources/wiki/start/topics/tutorials/install/>. If installing in macOS, use homebrew package manager: `brew install nginx`

4.2.3.2 Configuring NGINX

The way NGINX and its modules work is determined in the configuration file. By default, the configuration file is named `nginx.conf`. The location of the configuration file might be different depending on the server Operating System:

- In Linux and macOS based Operating Systems the `nginx.conf` file is placed in the directory `/usr/local/nginx/conf`, `/etc/nginx`, or `/usr/local/etc/nginx`
- In Windows based Operating Systems, the `nginx.conf` file is placed in the directory `conf` included inside the NGINX installation folder.

4.2.3.2.1 NGINX HTTP configuration

The following configuration example enables cache for static content over a regular HTTP, and offers the service over port 8080. The "TO UPDATE" sections in the config file, require parameter definition based on the server configuration where Policy Studio is installed:

nginx.config non SSL example:

```
events {
    worker_connections 1024;
}

http {
    include             mime.types;
    default_type        application/octet-stream;
    sendfile             on;

    client_body_in_file_only clean;
    client_max_body_size 10M;

    # gzip optional but saves space on transfers
    gzip                on;
    gzip_comp_level     6;
    gzip_vary           on;
    gzip_min_length     100;
    gzip_proxied        any;
    gzip_types          text/plain text/css application/json application/x-javascript;

    server {
```



```
# Port to listen from the browser
listen 8080;
server_name      localhost;
keepalive_timeout 70;

# root connections are being forwarded to the public folder in the d
# Add cache for faster loading of static files

location / {
    add_header Cache-Control "public, max-age=2592000";
    # TO UPDATE with the path to the Policy Studio public folder, for
    root /path/to/distribution/public;
}

# API connections are being forwarded to the node port
location /api {
    proxy_set_header    Host      $host;
    proxy_set_header    Upgrade   $http_upgrade;
    proxy_set_header    Connection "upgrade";
    proxy_http_version  1.1;
    # TO UPDATE with Policy Studio port (8585 by default)
    proxy_pass           http://localhost:8585/api;
    proxy_cache_bypass   $http_upgrade;
}
}
```

4.2.3.2.2 NGINX HTTPs configuration

The following configuration enables cache for static content on an HTTPs configuration, and offers the service over port 8080. The "TO UPDATE" sections in the config file, require parameter definition based on the server configuration where Policy Studio is installed:

nginx.config SSL example:

```
events {
    worker_connections 1024;
}

http {
    include      mime.types;
    default_type application/octet-stream;
    sendfile     on;

    client_body_in_file_only clean;
    client_max_body_size 10M;

    # gzip optional but saves space on transfers
    gzip           on;
    gzip_comp_level 6;
    gzip_vary      on;
```



```
gzip_min_length 100;
gzip_proxied    any;
gzip_types      text/plain text/css application/json application/x-javascript;

server {
    # # Port to listen from the browser, using ssl command to enforce https
    listen 8080 ssl;
    server_name      localhost;
    keepalive_timeout 70;

    # root connections are being forwarded to the public folder in the dist
    # Add cache for faster loading of static files
    location / {
        add_header Cache-Control "public, max-age=2592000";
        # TO UPDATE with the path to the Policy Studio public folder, for example
        root /path/to/distribution/public;
    }

    # API connections are being forwarded to the node port
    location /api {
        proxy_set_header    Host            $host;
        proxy_set_header    Upgrade         $http_upgrade;
        proxy_set_header    Connection     "upgrade";
        proxy_http_version  1.1;
        # TO UPDATE with Policy Studio port (8686 by default)
        proxy_pass           https://localhost:8686/api;
        proxy_cache_bypass   $http_upgrade;
    }

    # TO UPDATE with certificate paths to serve https. Must be the same file
    ssl_certificate         /path/to/server.crt;
    ssl_certificate_key     /path/to/server.key;
}
}
```

4.2.3.3 Starting NGINX

Start the server by running as **root** the NGINX process. The process is located in `/usr/local/nginx/sbin/nginx` for unix based servers, and in the installation folder for windows based servers.

After editing the `nginx.conf` configuration file, for the changes to take effect, stop the NGINX process and start again.

4.2.3.4 Accessing Policy Studio through NGINX

To access the Policy Studio, do the following:

1. Open a compatible web browser.
2. Access the Policy Studio server and port defined in `nginx.conf`.



- `http://<NGINX server_name>:<NGINX listen port>`

Or for secure connections:

- `https://<NGINX server_name>:<NGINX listen port>`, and accept the certificate if necessary.

5 Operation And Maintenance

5.1 Backup and Data Restore

The Policy Studio configuration and data can be persistently stored in the form of a backup. The backup preserves the Policy Studio configuration and minimizes the amount of work needed to perform if the system is disabled and is to be restored.

Do!

The system administrator is responsible for performing regular Policy Studio backups.

The Policy Studio backup contains a copy of the entire MongoDB database and does not include data stored on the SAPC nodes. The Policy Studio backup data includes:

- User accounts and hashed passwords.
- SAPC Node connections and credentials.
- Objects saved as 'Draft' on each of the available the SAPC nodes.
- Collections.
- Legal Notice Message.

5.1.1 Backup Procedure

To create a Policy Studio backup, do the following:

1. Navigate to the installation folder.
2. Execute the backup script:



```
Server:# node backup.js
```

The process finishes with a status report and location of the backup folder, for example:

```
Database successfully backed up to: /home/PolicyStudio/backup/
backup-2017-01-2-13-30-V.1.0/test
```

Note: The Policy Studio stores the backup in a folder named with the date and version information, and the name of the MongoDB database as a subfolder.

5.1.2 Restore Procedure

To restore the Policy Studio, do the following:

1. Navigate to the installation folder.
2. Execute the restore script:

```
node restore.js <backup directory>
```

The backup directory must be the MongoDB database folder. Following previous example:

```
Server:# node restore.js /home/PolicyStudio/backup/backup-2017
-01-2-13-30-V.1.0/test
```

5.2 Logging Management

The Policy Studio logs are stored in the installation folder: /logs folder, for example /home/PolicyStudio/logs/. There are three log types:

- **auth<rotation_index>.log:** access logs.
- **error<rotation_index>.log:** error logs.
- **sapc<rotation_index>.log:** all interactions with SAPC REST API logs.

Active logs do not have a <rotation_index>. The log files rotate increasing a rotation index based on the configured maximum file size defined in `server.config.json`. Also, the maximum number of simultaneous log files per log type, before a new log file overwrites the oldest log file is also configured in `server.config.json`.



5.3 Troubleshooting

5.3.1 Administrator Password Recovery

If the administrator password is lost, a reinstallation of the Policy Studio is required.

5.3.2 Restart the Policy Studio

To restart the Policy Studio, terminate the `node run.js` process and start again.

5.4 Upgrade

This section describes the procedure to upgrade the Policy Studio.

To upgrade the Policy Studio to a new release:

1. Perform the installation preparation steps as described in Section 3.3 on page 4, using the SAPC Software Gateway Ticket Number of the Policy Studio upgrading to.
2. Stop the `node run.js` process.
3. Install the new version of Policy Studio, following Step 3 to Step 6. To simplify the configuration process, it is possible to copy the configuration file `server.config.json` and the Policy Studio SSL certificates files (if SSL is enable) from the old software folder into the new software folder.

Attention!

When running the `npm install` command on an upgrade, do not reseed the database. The installation logic upgrades automatically the database. In that case, a preupgrade and postupgrade backup is performed and stored in the `backup` folder. The automated backups are named: `backup-<yyyy-mm-dd-hh-mm-ss>-V.<x.x.x>-pre-upgrade` and `backup-<yyyy-mm-dd-hh-mm-ss>-V.<x.x.x>-post-upgrade`.

4. To avoid caching issues:
 - Clean the browser cache.
 - Update NGINX configuration with the new Policy Studio version and restart the NGINX process.



5.5 Rollback

To perform a software rollback:

1. Stop the `node run.js` process.
2. Install previous Policy Studio version following Step 3 to Step 6 on the directory that contains the uncompressed version of Policy Studio rolling back to, reseeding the database when prompt.
3. Restore the backup saved before the upgrade was performed (`backup-<yyyy-mm-dd-hh-mm-ss>-V.<x.x.x>-pre-upgrade`).

6 SAPC Functionality Not Supported

This chapter describes functionality currently not supported by the Policy Studio. If any object with non-supported attributes is edited from the Policy Studio, the non-supported attributes are respected even though the UI does not offer edition capabilities for them.

6.1 REST resources

- Shared Dataplans
- Operator Specific Information

6.2 Functions

- All configuration attributes related to Application Detection and Control (ADC) function.
- All configuration attributes related to Flexible Output Protocol function.

6.3 Contents

Preconfigured PCC Rules:

- PccRuleType:2
- Precedence
- Flows



6.4 Dataplans

- Fair Usage session limits
- Fair Usage time limits

6.5 Subscribers

- CustomerId.
- Policies configured at subscriber level

6.6 Rules

- Values of Output attributes:
 - result: "deny"

6.7 Policy Tags

The following list of tags are recognized as unknown tags by the **Condition Builder**:

- AfData.media.typeAsInt
- AccessData.subscriber.chargingchars
- AccessData.bearer.controlMode
- AccessData.bearer.isAnTrusted
- AccessData.bearer.requestType
- AccessData.subscriber.service["serviceName"].isRunning
- AccessData.requestedQos.classIdentifier
- AccessData.requestedQos.mbrUplink
- AccessData.requestedQos.mbrDownlink
- AccessData.requestedQos.priorityLevel
- AccessData.bearer.isQosNegotiationPossible
- AccessData.subscriber.accumulatedUsage.reportingGroup["total"/"reportingGroupName"].current["type"]
- AccessData.subscriber.accumulatedUsage.reportingGroup["total"/"reportingGroupName"].currentPercentage["type"]



- `AccessData.subscriber.accumulatedUsage.reportingGroup["total"]/"reportingGroupName"].expiryDate["type"]`
- `AccessData.subscriber.receivedUsage.reportingGroup["total"]/"reportingGroupName"].usageType["type"]`
- `AfData.specificActions`
- Session limits related policy tags: `AccessData.subscriber.session.accumulatedUsage.X`

6.8 Condition Formula Expressions

The following list of condition functions are recognized as unknown by the **Condition Builder**:

- `firstCharsOf(string st, integer n)`
- `lastCharsOf(string st, integer n)`
- `strcat(string str1, string str2)`
- `substr(string str, int index, int length)`
- `StrDLeft(string st, string delimiter)`
- `StrDRight(string st, string delimiter)`
- `toString(integer int, string format)`

6.9 Condition Formula Operators

To show a visual representation of a condition, the **Condition Builder** requires the explicit use of parenthesis around all the elements of the condition formula.

7 Limitations

The following limitation must be considered when working with the Policy Studio.



7.1 Dataplans

When policies are added to a dataplan object, if a content is selected as the resource policy locator, it has to be in committed state (draft contents are not selectable from the policy locator resource list).

7.2 Profiles

Presence Reporting Area Profiles:

- Presence Reporting Area profiles counter not available.
- Presence Reporting Area committed profiles are not shown in the profile list unless searched by exact name.
- Presence Reporting Area Global export of committed objects not supported.

7.3 Global Policies

When policies are added to a global policies object, if a content or dataplan is selected as the resource policy locator, it has to be in committed state. Draft contents or draft dataplans are not selectable from the policy locator resource list.

7.4 Export Functionality

- The statuses of the exported objects are not saved on the export file. Because of this, it is not recommended to export two versions (committed and drafted) of the same object simultaneously, as it results in loss of data at import time.
- It is recommended to export objects selecting the option "Export items and dependencies" so that the parent object and its dependencies are treated as a whole at import time.
- Global Policies Locators in draft state cannot be exported.

7.5 Import Functionality

- Is not recommended to import objects as committed when some of its dependencies are existing as draft on the node importing to. Under these circumstances, is preferable to import the parent object as draft, otherwise the dependencies are not linked.
- If an object being imported has existing dependencies on the node importing to, the Policy Studio links the parent object to the committed version of the existing dependency as preferable option.
- Import JSON files must have a maximum size no larger than 1 MB.