Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.1)
Fine-grained Authorization

Project name	5G Enablers for Network and System Security and Resilience	
Short name	5G-ENSURE	
Grant agreement	671562	
Call	H2020-ICT-2014-2	
Delivery date	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	Thales Alenia Space: Gorka Lendrino Thales Services: Cyrille Martins	

Document Version	**Date**	**Change(s)**	**Author(s)**
0.1	28.06.2016	Created template	Felix Klaedtke
1.0 draft	01.08.2016	First draft	Gorka Lendrino Cyrille Martins
1.0	31.08.2016	First issue	Gorka Lendrino Cyrille Martins
1.1	30.09.2016	Completed enabler installation	Gorka Lendrino
1.2	03.10.2016	Corrected delivery file names in installation	Cyrille Martins

Foreword

5G-ENSURE belongs to the first group of EU-funded projects which collaboratively develop 5G under the umbrella of the 5G Infrastructure Public Private Partnership (5G-PPP) in the Horizon 2020 Programme. The overall goal of 5G-ENSURE is to deliver strategic impact across technology and business enablement, standardisation and vision for a secure, resilient and viable 5G network. The project covers research & innovation - from technical solutions (5G security architecture and testbed with 5G security enablers) to market validation and stakeholders' engagement - spanning various application domains.

5G security requirements collected in WP2 will be meet investigating and developing a number of 5G security and privacy enablers. An early vision of the proposed enablers has been provided in D3.1 together with a technical roadmap for R1. Subsequently, enablers planned to be software released for R1 have provide an open specification in D3.2.

This enabler's manual is part of deliverable D3.4 and will be accompanied by its SW delivery (D3.3).

Disclaimer

Copyright notice

Contents

1 Introduction 6
2 Installation and Administration Guide 6
2.1 System Requirements 6
2.2 Enabler Installation 8
2.2.1 Server-side 8
2.2.2 Client-side 12
2.3 Enabler Configuration 14
2.3.1 Server-side 14
2.3.2 Client-side 17
2.4 Troubleshooting 18
2.4.1 Basic distributed authorization enforcement for distributed RCDs feature 18
3 User and Programmer Guide 18
3.1 User Guide 18
3.1.1 Basic distributed authorization enforcement for distributed RCDs feature 18
3.1.2 Basic authorization in Satellite systems feature 20
3.2 Programmer Guide 20
3.2.1 Basic distributed authorization enforcement for distributed RCDs feature 20
3.2.2 Basic authorization in Satellite systems feature 21
4 Unit Tests 23
4.1 Basic distributed authorization enforcement for distributed RCDs feature 24
4.1.1 Information about Tests 24
4.1.2 Unit Test 1 24
4.1.3 Unit Test 2 24
4.1.4 Unit Test 3 25
4.2 Basic authorization in Satellite systems feature 25
4.2.1 Information about Tests 25
4.2.2 Unit Test 1 26
4.2.3 Unit Test 2 26
4.2.4 Unit Test 3 26
5 Acknowledgements 27
6 Abbreviations 27
7 References 28

1 Introduction

This manual describes the installation and administration guide of the "Fine-grained Authorization" enabler taking. Also, provides user and programmers guides and unit testing plans.

More and more sensitive **devices** are exposed on Internet every day. Controlling access to such devices is therefore critical, especially in industrial control systems, and requires flexible, dynamic and potentially complex access control models like **Role-Based Access Control (RBAC, see [10]) or Attribute-Based Access Control (ABAC, see [11])**. The current trend - especially on the web - is to use central authentication/authorization services from Third Parties online. This approach implies that protected resources/devices (or gateways on the resource side) have to authenticate and authorize (almost) every access requested.

To limit security **vulnerabilities** and address **technical issues** of this approach, the "Fine-grained authorization in resource-constrained devices" feature propose an alternative allowing IoT devices/resources to enforce **full offline** attribute-based **authorization** (no direct/live link to a central third-party) with ABAC capabilities, yet preserving the benefit of **central access management** from the first approach.

The main security goal of the "Fine-grained authorization in Satellite systems" feature is to support different authorization methods (RBAC/ABAC) and policies to provide basic access control to devices and services in 5G integrated satellite and terrestrial systems. This SW release covers one identity-management situation involving satellite networks in which the operator/device attaches to the satellite network to grant access to the satellite resources.

The manual is organized as follows. In Section 2, we describe enabler is installed, configured, and administrated, and in Section 3, we describe how to use the enabler. In Section 4, we provide the tests needed to cover all the features of the enabler. In Section 5, we thank people for contributions. Finally, abbreviations and references are listed in Section 6 and Section 7.

2 Installation and Administration Guide

The "Fine-grained Authorization" enabler is composed of:

- the "Fine-grained authorization in resource-constrained devices" feature, and
- the "Fine-grained authorization in Satellite systems" feature.

This section covers the system requirements, and describes how the enabler is installed and configured.

2.1 System Requirements

The "Fine-grained Authorization" enabler consists of:

- A client-side feature (e.g. resource-constrained device, satellite terminal, UE…), able to grant access control to devices and services based on an access token provided by the server-side feature. This feature shall be deployed on each device.

- A server-side feature (i.e. authorization server), able to configure the client-side features and support different authorization methods to provide access control to devices and services. This feature shall be deployed on a central server.

The basic/minimal system requirements for this enabler are that the server and the network components should be running Linux, preferably Ubuntu Server 16.04.1 LTS or newer on an x64 architecture (you can get help from the Ubuntu server guide [13]), with a network interface.

When possible, encrypt user home directory (the operative system will seamlessly mount the encrypted home directory each time the user login and automatically unmounts when the user log out of all active sessions) and set up encrypted LVM (partitions will not be readable without knowing a special key). This feature is useful to protect sensitive data in case server or hard drive gets stolen. The thief might get physical access to the hard drive, but without knowing the right passphrase, the data on the hard drive will look like random characters.

The client host(s) (i.e. 5g-fga-sat-cli01.5g-ensure.eu) in the test-bed is a vSmall virtual host with the following requirements:

- 1 CPU (x86_64)
- 2 GB RAM
- 20 GB Hard disk
- File system: ext4
- Ethernet network interface

It will act as a resource-constrained-device (e.g. IoT), satellite terminal…

The server host (i.e. 5g-fga-sat-srv01.5g-ensure.eu) in the test-bed is a vMedium virtual host with the following requirements:

- 2 CPUs (x86_64)
- 4 GB RAM
- 40 GB Hard disk
- File system: ext4
- Ethernet network interface

It will act as an authorization server.

The client-server API follows the RESTful Web Services (JAX-RS 2.0), therefore a Java Application Server is needed in both sides (any should work, but only Apache HTTP has been tested):

- Client side: to configure the client-side and to grant access to the resources.
- Server side: to provide authorization access control.

Also, the satellite feature uses:

- A LDAP instance.

Remote service access:

- Host access to Ubuntu repositories (main, universe).
- Host access to NTP server of stratum ≤ 4.
- SSH access to host (public access or restricted to specific public source IP address).
- HTTPS (TCP/443) access to host (public access or restricted to specific public source IP address).

2.2 Enabler Installation

The installation package contains two archived web applications (client-side and server-side).

Download the enabler package release from the software repository of the 5G-ENSURE project (e.g. https://<5g-ensure.git>/enablers/FineGrainedAuthorization/archive/FineGrainedAuthorization.vXX_YY_ZZ.tar.gz) later extract the package on the temporal folder **/tmp** or clone the GitHub project (e.g. https://**<5g-ensure.git>**/enablers/FineGrainedAuthorization.git) on the temporal folder **/tmp**.

As a result, all the installation files of this enabler will be located in **/home/tase/FineGrainedAuthorization.vXX_YY_ZZ**.

2.2.1 Server-side

2.2.1.1 Basic distributed authorization enforcement for distributed RCDs feature

Extract /tmp/FineGrainedAuthorization.vXX_YY_ZZ/FineGrainedAuthorization-RCD.vXX_YY_ZZ.zip content in a random directory.

Execute the following commands starting from this directory.

2.2.1.1.1 XACML PDP Installation

Install AuthZForce CE Server 5.4.1 (see [1]) and its dependencies (JDK and Apache Tomcat).

Deploy authzscope-authzforce-extension vX.Y.Z:

```
$ sudo cp authzforce/authzscope-authzforce-extension-X.Y.Z.jar /opt/authzforce-ce-server/webapp/WEB-INF/lib
```

Deploy configuration:

```
$ sudo cp authzforce/authzforce-ext.xsd /opt/authzforce-ce-server/conf
$ sudo cp authzforce/catalog.xml /opt/authzforce-ce-server/conf
$ sudo cp authzforce/pdp.xml /opt/authzforce-ce-server/data/domains/A0bdIbmGEeWhFwcKrC9gSQ
```

Restart tomcat:

```
$ sudo service tomcat7 restart
```

2.2.1.1.2 OAuth2 Server Installation

2.2.1.1.3 Apache2

Install apache2

```
$ sudo apt-get install apache2
```

Install openssl

```
$ sudo apt-get install openssl
```

Activate Apache2 SSL module

```
$ sudo a2enmod ssl
```

Generate a RSA 2048 key pair and a certificate

```
$ openssl genrsa -out privkey.pem 2048
$ openssl openssl rsa -in privkey.pem -pubout -out pubkey.pem
$ openssl req -key privkey.pem -new -x509 -days 2048 -out server.cr
```

Fill the question with answers relevant in the deployment context. Caution: the Common Name must exactly match the domain name of the server.

Configure Apache2 to enable SSL

- Follow step 3 and step 4 of the following guide for ssl configuration into Apache: [2]

2.2.1.1.4 PHP 7

Install php 7

```
$ sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql php-curl php-xml
```

Create symbolic link from token.php to apache2 www directory

```
$ sudo ln -s `readlink -f oauth2-server/token.php` /var/www/html/token.php
```

Restart Apache2

```
$ sudo service apache2 restart
```

2.2.1.1.5 MySQL 5.7

Install mysql 5.7

```
$ sudo apt install mysql-server mysql-client
$ sudo mysql_secure_installation
```

During the installation, set a root password and answer yes to all questions of mysql_secure_installation process.

Copy key files (generated in 2.2.1.1.3) in a directory that mysql will be able to read

```
$ sudo cp pubkey.pem privkey.pem /var/lib/mysql-files
```

Create and define database

```
$ mysql -u root -p < oauth2-server/oauth_install.sql
```

Enter the root password defined during mysql 5.7 installation

2.2.1.2 Basic authorization in Satellite systems feature

First of all disable unattended updates:

$ sudo systemctl disable apt-daily.service

$ sudo systemctl disable apt-daily.timer

$ sudo shutdown -r now

Add tase user to the sudoers file:

$ cd FineGrainedAuthorization.vXX_YY_ZZ/server

$ mkdir cp

$ sudo cp -p /etc/sudoers cp/

$ sudo tee -a /etc/sudoers << EOF

tase ALL=(ALL:ALL) NOPASSWD:ALL

EOF

Update package information:

$ sudo apt-get install -f

$ sudo dpkg --configure -a

$ sudo apt-get update

Configure time zone:

$ sudo timedatectl set-timezone UTC

Configure host name:

$ sudo cp -p /etc/hosts cp/

$ sudo hostnamectl set-hostname '5g-fga-sat-srv01'

$ sudo sed -i s/ubuntu/"5g-fga-sat-srv01.5g-ensure.eu\t5g-fga-sat-srv01"/g /etc/hosts

Add the FGA_SAT_PATH environment variable to tase user:

$ sudo tee -a /home/tase/.bashrc << EOF

export FGA_SAT_PATH=~/FineGrainedAuthorization.v01_00_00/

EOF

Install OpenJDK:

$ sudo apt install -y openjdk-8-jre

Install PostgreSQL:

$ sudo apt install -y postgresql postgresql-contrib

$ sudo -u postgres psql postgres

postgres=# \password postgres

Enter new password:

Enter it again:

postgres=# \q

$ sudo su - postgres

$ psql

postgres=# CREATE USER tase WITH PASSWORD 'tasetase';

postgres=# CREATE DATABASE snm OWNER tase;

postgres=# \q

$ exit

Install OpenLDAP (LDAP administrator password: secret):

$ sudo apt install -y slapd ldap-utils

Install docker:

$ sudo apt-get install -y apt-transport-https ca-certificates

$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D

$ sudo tee /etc/apt/sources.list.d/docker.list << EOF

deb https://apt.dockerproject.org/repo ubuntu-xenial main

EOF

$ sudo apt-get update

$ sudo apt-get purge lxc-docker

$ sudo apt-get install -y linux-image-extra-$(uname -r) linux-image-extra-virtual

$ sudo apt-get install -y docker-engine

$ sudo usermod -aG docker tase

Install docker compose:

$ curl -L https://github.com/docker/compose/releases/download/1.8.0/docker-compose-`uname -s`-`uname -m` > docker-compose

$ sudo mv docker-compose /usr/local/bin/docker-compose

$ sudo chmod +x /usr/local/bin/docker-compose

Create the terminal docker image:

$ cd ..

$ cd common/Dockerfile_satelliteTerminal

$ docker build -t satelliteTerminal_img .

$ docker save satelliteTerminal_img > satelliteTerminal_img.tar

$ cd ..

Alternatively, use the following command:

$ /tmp/FineGrainedAuthorization.vXX_YY_ZZ/FineGrainedAuthorization-Sat.vXX_YY_ZZ/server/install-server.sh

Among other things, this script allows to:

- Set time zone to UTC.
- Configure the server IP address.
- Map hostnames to IP addresses.
- Configure JAVA_HOME and PATH environment variables.
- Create the FGA_SAT_PATH environment variable.
- Install the required COTS.

2.2.2 Client-side

2.2.2.1 Basic authorization in Satellite systems feature

First of all disable unattended updates:

$ sudo systemctl disable apt-daily.service

$ sudo systemctl disable apt-daily.timer

$ sudo shutdown -r now

Add tase user to the sudoers file:

$ cd FineGrainedAuthorization.vXX_YY_ZZ/client

$ mkdir cp

$ sudo cp -p /etc/sudoers cp/

$ sudo tee -a /etc/sudoers << EOF

tase ALL=(ALL:ALL) NOPASSWD:ALL

EOF

Update package information:

$ sudo apt-get install -f

$ sudo dpkg --configure -a

$ sudo apt-get update

Configure time zone:

$ sudo timedatectl set-timezone UTC

Configure host name:

$ sudo cp -p /etc/hosts cp/

$ sudo hostnamectl set-hostname '5g-fga-sat-cli01'

$ sudo sed -i s/ubuntu/"5g-fga-sat-cli01.5g-ensure.eu\t5g-fga-sat-cli01"/g /etc/hosts

Add the FGA_SAT_PATH environment variable to tase user:

$ sudo tee -a /home/tase/.bashrc << EOF

export FGA_SAT_PATH=~/FineGrainedAuthorization.v01_00_00/

EOF

Install OpenJDK:

$ sudo apt install -y openjdk-8-jre

Install OpenLDAP (LDAP administrator password: secret):

$ sudo apt-get install -y apt-transport-https ca-certificates

$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D

$ sudo tee /etc/apt/sources.list.d/docker.list << EOF

deb https://apt.dockerproject.org/repo ubuntu-xenial main

EOF

$ sudo apt-get update

$ sudo apt-get purge lxc-docker

$ sudo apt-get install -y linux-image-extra-$(uname -r) linux-image-extra-virtual

$ sudo apt-get install -y docker-engine

$ sudo usermod -aG docker tase

Install docker compose:

$ curl -L https://github.com/docker/compose/releases/download/1.8.0/docker-compose-`uname -s`-`uname -m` > docker-compose

$ sudo mv docker-compose /usr/local/bin/docker-compose

$ sudo chmod +x /usr/local/bin/docker-compose

Create the terminal docker image:

$ cd ..

$ cd common/Dockerfile_satelliteTerminal

$ docker build -t satelliteTerminal_img .

$ docker save satelliteTerminal_img > satelliteTerminal_img.tar

$ cd ..

Alternatively, use the following command:

```
$ /tmp/FineGrainedAuthorization-Sat.vXX_YY_ZZ/client/install-client.sh
```

Among other things, this script allows to:

- Set time zone to UTC.
- Configure the client IP address.
- Map hostnames to IP addresses.
- Configure JAVA_HOME and PATH environment variables.
- Create the FGA_SAT_PATH environment variable.
- Install the required COTS.

2.3 Enabler Configuration

2.3.1 Server-side

2.3.1.1 Basic distributed authorization enforcement for distributed RCDs feature

2.3.1.1.1 User provisioning

Authenticable users must be provisioned in the local “oauth” database, created during the enabler installation.

User credentials and attributes must be provisioned like follows:

```
INSERT INTO oauth_clients (client_id, client_secret) VALUES ("<login>", "<password>");
INSERT INTO oauth_users (username, first_name, last_name, role) VALUES ("<login>", "<First Name>", "<Last Name>", "<Role>");
```

See oauth2-server/oauth_users.sql as an example.

2.3.1.1.2 User attributes definition

User attributes must be defined in the local “oauth” database schema. You can simply add new attributes with the following SQL command:

```
ALTER TABLE oauth_users ADD <column_name> <column_datatype>;
```

or completely recreate the table with:

```
DROP TABLE IF EXISTS oauth_users;
CREATE TABLE oauth_users (username VARCHAR(80) NOT NULL, <attribute1> <datatype>, <attribute2> <datatype>, <attribute3> <datatype>, PRIMARY KEY (username));
```

See oauth2-server/oauth_users.sql as an example.

2.3.1.2 Basic authorization in Satellite system feature (TBD)

For security of communications it is highly recommended to enable SSL/TSL using the following command:

```
$ /tmp/FineGrainedAuthorization.vXX_YY_ZZ/FineGrainedAuthorization-Sat.vXX_YY_ZZ/server/security-server.sh
```

Among other things, this script allows to:

- Generate RSA keys and self-signed certificate

Finally, add entries to the LDAP dictionary information tree using the following command:

```
$ /tmp/FineGrainedAuthorization.vXX_YY_ZZ/FineGrainedAuthorization-Sat.vXX_YY_ZZ/server/loadDIT.sh
```

The LDAP dictionary tree represent a test organization for testing purposes:

- Domain: 5g-ensure.eu
- Root: admin
- Users:
  - Christopher Carroll
  - Mildred Dunn
  - admin5g
- Groups:
  - SNO (Satellite Network Operator). Formed by Christopher Carroll
  - SVNO (Satellite Virtual Network Operator). Formed by Mildred Dunn

The contents of the loadDIT.sh file is:

```
###Initialization
DOMAIN_ROOT=5g-ensure
DOMAIN_EXTENSION=eu
DOMAIN_DC="dc=$DOMAIN_ROOT,dc=$DOMAIN_EXTENSION"
ROOTDN="cn=admin,$DOMAIN_DC"
ROOTDN_PASSWORD=secret
```

```
###Add Organizational Units
ldapadd -w $ROOTDN_PASSWORD -D $ROOTDN <<EOF
dn: ou=groups,$DOMAIN_DC
objectClass: organizationalUnit
ou: groups

dn: ou=users,$DOMAIN_DC
ou: users
objectClass: organizationalUnit
EOF

###Add users
###http://uinames.com/
ldapadd -w $ROOTDN_PASSWORD -D $ROOTDN <<EOF
dn: cn=Christopher Carroll,ou=users,$DOMAIN_DC
objectClass: inetOrgPerson
cn: Christopher Carroll
givenName: Christopher
sn: Carroll
uid: christopher.carroll
userPassword: chCA_5g
mail: christopher.carroll@5g-ensure.eu

dn: cn=Mildred Dunn,ou=users,$DOMAIN_DC
objectClass: inetOrgPerson
cn: Mildred Dunn
givenName: Mildred
sn: Dunn
uid: mildred.dunn
userPassword: miDu_5g
mail: mildred.dunn@5g-ensure.eu

dn: cn=admin5g,ou=users,$DOMAIN_DC
objectClass: inetOrgPerson
objectClass: posixAccount
```

```
cn: 5G-Ensure Administrator
sn: Administrator
uid: admin5g
userPassword: 5g-ensure
mail: admin5g@bg-ensure.eu
uidNumber: 1000
gidNumber: 1000
homeDirectory: /home/admin5g
loginShell: /bin/bash
EOF

###Add groups
ldapadd -w $ROOTDN_PASSWORD -D $ROOTDN <<EOF
dn: ou=sno,ou=groups,$DOMAIN_DC
objectClass: groupOfNames
ou: sno
cn: sno
member: cn=Christopher Carroll,ou=users,$DOMAIN_DC

dn: ou=svno,ou=groups,$DOMAIN_DC
objectClass: groupOfNames
ou: svno
cn: svno
member: cn=Mildred Dunn,ou=users,$DOMAIN_DC
EOF
```

2.3.2 Client-side

2.3.2.1 Basic authorization in Satellite systems feature (TBD)

For security of communications it is highly recommended to enable SSL/TSL using the following command:

```
$ /tmp/FineGrainedAuthorization.vXX_YY_ZZ/FineGrainedAuthorization-Sat.vXX_YY_ZZ/client/security-client.sh
```

This script configures SSL/TSL accordance with the server (see 2.3.1.2)

2.4 Troubleshooting

2.4.1 Basic distributed authorization enforcement for distributed RCDs feature

For troubles during XACML PDP installation or using Policy Management API, please refer to AuthZForce 5.4.1 Diagnosis Procedures (see [3]).

For troubles during OAuth2 server installation, please refer to Apache2, MySQL 5.7 or OpenSSL documentation, according to which step the error occurs.

For troubles using Authentication API, please ensure that the debug mode is set in oauth2-server/server.php by uncommenting the following line

```
ini_set('display_errors',1);error_reporting(E_ALL);
```

Then refer to the error message appearing with PHP7 documentation.

3 User and Programmer Guide

This section describes how to use the enabler and how to “program” the enabler.

3.1 User Guide

3.1.1 Basic distributed authorization enforcement for distributed RCDs feature

3.1.1.1 Policy Management API

This API concerns the RCD owner in order to manage its RCDs access control policy.

3.1.1.1.1 Set policy

Add a new XACML PolicySet:

- Method: POST
- Path: /authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pap/policies
- Headers:
  - Content-Type: application/xml
  - Accept: application/xml
  - Content-Length: <content_length>
- Body: XACML PolicySet as defined in the XACML 3.0 schema

Caution: the top level PolicySet element must respect the following rules:

- The PolicySetId attribute must be “root”
- The PolicyCombiningAlgId must be “urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit”
- The Version attribute must be lexically greater than the one of the last uploaded PolicySet
- It must contain (at least) the following AdviceExpression

```
<AdviceExpression AdviceId="urn:thalesgroup:authzforce:scope:advice" AppliesTo="Deny">
      <AttributeAssignmentExpression
```

```
AttributeId="urn:thalesgroup:authzforce:scope:assignment">
              <AttributeDesignator AttributeId="oauth2-scope"
                     Category="urn:oasis:names:tc:xacml:3.0:attribute-
category:environment"
                     DataType="urn:thalesgroup:authzforce:scope:data-type:xacml-
policyset"
                     MustBePresent="true" />
       </AttributeAssignmentExpression>
</AdviceExpression>
```

See authzforce/policytest.xml as an example

3.1.1.1.2 Get policy

Returns the last deployed XACML PolicySet

- Method: GET
- Path: /authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pap/policies/root/latest
- Headers:
    - Accept: application/xml

3.1.1.1.3 Delete policy

Deletes a deployed XACML PolicySet

- Method: DELETE
- Path: /authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pap/policies/root
- Headers :
    - Accept: application/xml

3.1.1.2 Authentication API

This API concerns clients in order to request for an access token using their own credentials

3.1.1.3 Request token

Returns a JSON payload with the following fields:

- access_token: the effective JWT encoded and signed access token
- expires_in : the lifetime in seconds of the access token
- token_type: the access token type
- scope: a pre-resolved XACML PolicySet (according to user attributes), synthetizing the user access rights, that must be evaluated by the policy enforcement point without requiring external connection (resource, action and environment attributes to be evaluated). If set to '*', it means that the user has all rights.

- Method: POST
- Path: /token.php
- Headers:
    - Authorization: "<login>:<password>" encoded in base64
    - Content-Type: application/x-www-form-urlencoded
    - Content-Length : 29
- Body: "grant_type=client_credentials"

3.1.2 Basic authorization in Satellite systems feature

This section is not applicable as the enabler is a programmer tool; therefore there are no different user and programmer guides.

3.2 Programmer Guide

3.2.1 Basic distributed authorization enforcement for distributed RCDs feature

3.2.1.1 How to integrate your own user directory

This guide concerns those who want to integrate their own user directory with the OAuth 2.0 authentication server provided in the enabler. Please ensure that your user directory is able to provide user credentials as well as user attributes (for access control evaluation).

If starting from a fresh install, follow the 2.2.1.1.1, 2.2.1.1.3 and 2.2.1.1.4 sections of the installation guide only.

Edit oauth2-server/server.php and replace the line

```
$storage = new OAuth2\Storage\Pdo(array('dsn' => $dsn, 'username' => $username, 'password' => $password));
```

in order to instantiate your own storage object.

You can use one of the default provided connectors (PDO, Mongo, Redis, Cassandra, DynamoDB) in following the matching guide at [4] or implement your own custom storage in following the guide at [5].

If you don't want the OAuth2 server to use your storage to store its working data, you might want to take a look at the guide to use multiple storages at [6].

Then, you should edit the scope module of the OAuth 2.0 server in order to match your user attributes. Please follow the guide at [7] in order to create your own scope module, then integrate it to oauth2-sever/server.php in replaceing the line

```
$server->setScopeUtil(new XACMLScope($storage, $pdpURL));
```

with your own scope module instance.

See 3.2.1.2 section for more information on how your scope module should behave.

3.2.1.2 How to integrate your own OAuth2 authentication service

This guide concerns those who already have their own user directory as well as OAuth 2.0 authentication service and want to integrate it with the rest of the enabler. Please ensure that your OAuth 2.0 supports the Client Credentials authentication flow, is able to provide signed and timestamped access tokens (using JWT format for example), and allows to dynamically define the token scope.

If starting from a fresh install, follow the2.2.1.1.1 section of the installation guide only.

Then, you should customize your OAuth 2.0 service in order to make it perform requests toward the XACML PDP. You might take oauth2-server/XACMLScope.php as an example.

Your scope module should request the XACML PDP at URL http://localhost:8080/authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pdp with an XACML Request containing all user attributes, as defined in the access control policy. The request should have the following form:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
CombinedDecision="false" ReturnPolicyIdList="false">
    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject">
        <Attribute AttributeId="attribute1" IncludeInResult="false">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Value1</AttributeValue>
        </Attribute>
        <Attribute AttributeId="attribute2" IncludeInResult="false">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Value2</AttributeValue>
        </Attribute>
    </Attributes>
</Request>
```

Please ensure that all the user attributes used in the access control policy are present in the request. If there is no value for an attribute for a given user, please put the matching <Attribute> element but without any value inside the <AttributeValue> child, in order to make them disappear from the future OAuth 2.0 token scope.

Then, the PDP should answer with a XACML Response, containing a Decision (which can be 'Permit' or 'Deny'), and in the case of a Deny decision, an advice with AdviceId "urn:thalesgroup:authzforce:scope:advice" containing an AttributeAssignment containing a XACML PolicySet that should be used as token scope. In the case of a Permit decision, set the scope to "*".

3.2.2 Basic authorization in Satellite systems feature

The enabler provides a REST interface, a resource-oriented API accessed via HTTP that uses JSON-based representations for information interchange. This Programmers Guide mainly consists of an overview of these RESTful API calls.

Description	Add a new XACML policy to the PAP policies repository
HTTP Method	POST
URI	https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-srv/api/v01.00.00/pap/policy
URI Parameters	N/A
Request Header	N/A
Request Body Content-type	application/xml
Request Body Content	XACML policy

HTTP status code	201 when success 400 when error
Response Header	N/A
Accept Content-type	N/A
Response Body Content	N/A

Description	Get a XACML policy from the PAP policies repository
HTTP Method	GET
URI	https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-srv/api/v01.00.00/pap/policies/{policyID}
URI Parameters	{policyID}: requested policy id
Request Header	N/A
Request Body Content-type	N/A
Request Body Content	N/A
HTTP status code	200 when success 404 when not found
Response Header	N/A
Accept Content-type	application/xml
Response Body Content	N/A

Description	Request RCD or satellite resource access
HTTP Method	POST
URI	https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-srv/api/v01.00.00/pdp/authorize

URI Parameters	N/A
Request Header	Authorization: Basic <userName>:<userPassword> (encoded in base64)
Request Body Content-type	application/json
Request Body Content	{"userName":<username>,"action":<action>,"resource":<resource>}
HTTP status code	200 when allow access 401 when deny access
Response Header	N/A
Accept Content-type	application/json
Response Body Content	N/A

Description	Start transmit CW Carrier
HTTP Method	PUT
URI	https://5g-fga-sat-cli01.5g-ensure.eu:8080/fga-sat-cli/api/v01.00.00/satelliteModem/mgmt/startCWCarrier where CW means continuous wave
URI Parameters	N/A
Request Header	Authorization: 5GE-HMAC-SHA256 <userName>:<userPassword>
Request Body Content-type	application/json
Request Body Content	{"frequency":" integer_Hz","power":" real_dBm","cwMaximunDuration":" integer_sec"}
HTTP status code	201 when success 404 when not found
Response Header	N/A
Accept Content-type	application/json
Response Body Content	N/A

4 Unit Tests

This section describes the tests needed to cover all the features of the enabler. These tests verify that the enabler is ready to be evaluated in WP4.

Both features are based on RESTful HTTP API, therefore any HTTP client can be used to perform the test. For testing purposes curl command line tool has been used.

4.1 Basic distributed authorization enforcement for distributed RCDs feature

4.1.1 Information about Tests

The aim of the enabler is to provide an OAuth2 access token, signed, using JWT format. This token must contain a scope that is a particular XACML PolicySet corresponding to the current user rights. This token should be evaluated to permit actions the token owner has permission to do, or deny otherwise.

The following tests will ensure that:

- a global access control policy in XACML can be uploaded

and for a given global access control policy and given user credentials:

- the enabler returns an OAuth2 access token if and only if the user credentials are correct
- the token is correctly JWT-encoded, timestamped and signed
- the token scope contains a valid XACML PolicySet corresponding to the current user rights

Important: the following tests must be performed in order on the server host, and using installation directory as current directory.

4.1.2 Unit Test 1

This test aims to initialize the user database with test users.

Load data on the server host:

```
$ mysql -u root -p < oauth2-server/oauth_users.sql
```

Enter the root password defined during mysql 5.7 installation.

Expected result is:

client_id	client_secret	redirect_uri	grant_types	scope	user_id	username
first_name	last_name	role				
admin	adminpwd	NULL	NULL	NULL	NULL	admin	NULL	NULL	Admin
guest	guestpwd	NULL	NULL	NULL	NULL	guest	NULL	NULL	NULL
john	johnpwd	NULL	NULL	NULL	NULL	john	John	Doe	Operator

4.1.3 Unit Test 2

This test aims to verify that the global XACML access control policy is correctly deployed. This XACML access control policy will be used in the following tests.

Deploy the test policy on the server host:

```
$ curl -H "Content-Type: application/xml" --data "@authzforce/policytest.xml"
http://localhost:8080/authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pap/policies
```

Retrieve the last deployed policy from the server host:

```
$ curl -H “Accept: application/xml” http://localhost:8080/authzforce-ce/domains/A0bdIbmGEeWhFwcKrC9gSQ/pap/policies/root/latest
```

Expected result is the PolicySet as defined in authzforce/policytest.xml

4.1.4 Unit Test 3

This test aims to verify that the authentication API is functional.

We assume the following terms:

- <domain_name>: the domain name of the service as defined during 2.2.1.1.3 installation phase.
- <ca_path>: the full path of the directory containing the certificate generated during 2.2.1.1.3 installation phase.

Request for a token as user admin from the server host:

```
$ curl -u admin:adminpwd https://<domain_name>/token.php -d ‘grant_type=client_credentials’ --capath <ca_path>
```

Expected result should be of the following form:

```
{"access_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6IjBhOWVkZTVlMzZjM2JjOTk4ZjZjMWMzZDA2ZWUzNjFkMTQzMTRiYzgiLCJqdGkiOiIwYTllZGU1ZTM2YzNiYzk5OGY2YzFjM2QwNmVlMzYxZDE0MzE0YmM4IiwiaXNzIjoiIiwiYXVkIjoiYWRtaW4iLCJzdWIiOm51bGwsImV4cCI6MTQ3MzE4MTk0NywiaWF0IjoxNDczMTc4MzQ3LCJ0b2tlbl90eXBlIjoiYmVhcmVyIiwic2NvcGUiOiIqIn0.TBPmNARErLi2CxRs5hscRBuz1D79P72t3-sKxKLUJORbfW2fh_3XtSC9T_9SZ9bjMUDQSLTyu2aqYik3VB53Pw2QEgUwGVe5PENwislO0BpYz1tk9WZVrYFlGya1F-qlCBwXE8SLqsmHgE6heZCSuodb7cxZ_IymT3rCwR1pHE2UjZC1i87UlvavTnwHuGHPbI-JO4ANW1CuN8yizfBHlYmqSXvDwjBF-pJjErN3UUgq45c-bDvhr8ZHu_E5mIpsf2MojY53OG16Rfa72NVDTZxQqBRyIjR3udPjXesvfDmTB8ot5ikCSTB_TPhfoJ9D4_K7JoEQW241ZLEt7nLMRA","expires_in":3600,"token_type":"bearer","scope":"*"}
```

access_token field is dependent of the current time so this should be slightly different but *expires_in*, *token_type* and *scope* should be identical.

4.2 Basic authorization in Satellite systems feature

4.2.1 Information about Tests

Important: the following tests must be performed in order on the server host, and using $5G_FGA_SAT directory as current directory.

4.2.2 Unit Test 1

This test aims to initialize the PAP policies repository with test policies.

Deploy the test policy on the server host:

```
$ curl -X GET -H "Content-Type: application/xml" -H "Accept: application/json" --data "@test/UT01/input/UT01_TestPolicy.xml" https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-srv/api/v01.00.00/pap/policy
```

Expected result is HTTP status code 201.

4.2.3 Unit Test 2

This test aims to verify the XACML policies are correctly deployed in the PAP policies repository.

Request policy from the server host:

```
$ curl -X GET -H "Accept: application/xml" https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-srv/api/v01.00.00/pap/policies/UT01_TestPolicy
```

Expected result is HTTP status code 200 and the response body with the XACML policy previously uploaded.

4.2.4 Unit Test 3

This test aims to verify that the authentication/authorization API is functional.

Request for "PUT" action on resource https://5g-fga-sat-cli01.5g-ensure.eu:8080/fga-sat-cli/api/v01.00.00/satelliteModem/mgmt/startCWCarrier.

The contents of the test/UT03/input/UT03_RequestContent.json file is:

```
{"userName":<userName>,"action":"PUT","resource":"https://5g-fga-sat-rcd01.5g-ensure.eu:8080/fga-sat-rcd/api/v01.00.00/satelliteModem/mgmt/startCWCarrier"}
```

Request resource action from the RCD using the server host as an authorization proxy:

```
$ curl -H "Authorization: Basic <userName>:<userPassword>" -H "Content-Type: application/json" -H "Accept: application/json" --data "@UT03/input/UT03_RequestContent.json" https://5g-fga-sat-srv01.5g-ensure.eu:8080/fga-sat-srv/api/v01.00.00/pdp/authorize
```

Where <userName> is "Christopher Carroll" and <userPassword> is "chCA_5g" as defined in the LDAP.

Expected result is the Satellite Terminal SW version:

```
{ "header":{
        "responseCode":"200","bodyContentType":"application/json"
    },"body":{
        "status":"OK"
    }
}
```

5 Acknowledgements

The following partners contributed to this deliverable: Thales Alenia Space and Thales Services.

Thanks Martin Svensson (WPL), 5G-ENSURE peers and 5G-ENSURE Steering Committee for their collaboration and support.

Thanks to AuthZForce Authorization PDP (FIWARE Access Control GE) which provides a XACML 3.0 Authorization engine included in this enabler implementation (see [8]).

Thanks to OAuth 2.0 Server PHP which provides a simple OAuth 2.0 authentication server with support of Client Credentials autentication flow and signed and timestamped JWT access tokens included in this enabler implementation (see [9]). Thanks as well to oauth2-server-php-mysql which helped to write initialization scripts for MySQL PDO storage support (see [10]).

6 Abbreviations

This section comprises a summary of terms and definitions used during the later sections:

5G-PPP	**5G infrastructure Public Private Partnership**
ABAC	**Attribute-Based Access Control:** an access control paradigm whereby access rights are granted according to a combination of attributes of any type (user, resource, environment, etc.)
API	**Application Programming Interface**
CE	**Community Edition**
HTTPS	**HyperText Transfer Protocol Secure**
IP	**Internet Protocol**
JDK	**Java Development Kit**
JSON	**JavaScript Object Notification**
JWT	**Java Web toolkit**
LDAP	**Lightweight Directory Access Protocol**
LTS	**Long Term Support**
LVM	**Logical Volume Manager**
NTP	**Network Time Protocol**
PAP	**Policy Administration Point**
PDO	**PHP Data Objects**
PDP	**Policy Decision Point**
PHP	**PHP: Hypertext Preprocessor**

RBAC	**Role-Based Access Control:** an access control paradigm whereby access rights are granted according to roles and privileges
RCD	**Resource-Constrained Device**
REST	**REpresentational State Transfer**
RSA	**Rivest, Shamir Adleman public-key cryptosystem**
SSH	**Secure SHell**
SSL	**Secure Sockets Layer**
TCP	**Transmission Control Protocol**
TSL	**Transport Layer Security**
UE	**User Equipment**
URI	**Uniform Resource Identifier**
URL	**Uniform Resource Locator**
UT	**Unit Test**
UTC	**Coordinated Universal Time**
XACML	**eXtensible Access Control Markup Language**

7 References

[1] C. Dangerville, "Minimal setup | AuthZForce - Installation and Administration Guide," [Online]. Available: http://authzforce-ce-fiware.readthedocs.io/en/release-5.4.1/InstallationAndAdministrationGuide.html#minimal-setup.

[2] J. Ellingwood, "How To Create a SSL Certificate on Apache for Ubuntu 14.04," 23 April 2014. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-14-04.

[3] C. Dangerville, "Diagnosis Procedures | AuthZForce - Installation and Administration Guide," [Online]. Available: http://authzforce-ce-fiware.readthedocs.io/en/release-5.4.1/InstallationAndAdministrationGuide.html#diagnosis-procedures.

[4] B. Shaffer, "PDO Storage | OAuth2 Server PHP," [Online]. Available: https://bshaffer.github.io/oauth2-server-php-docs/storage/pdo/.

[5] B. Shaffer, "Custom Storage | OAuth2 Server PHP," [Online]. Available: https://bshaffer.github.io/oauth2-server-php-docs/storage/custom/.

[6] B. Shaffer, "Using Multiple Storages | OAuth2 Server PHP," [Online]. Available: https://bshaffer.github.io/oauth2-server-php-docs/storage/multiple/ .

[7] B. Shaffer, "Scope | OAuth2 Server PHP," [Online]. Available: https://bshaffer.github.io/oauth2-server-php-docs/overview/scope/.

[8] C. Dangerville, "Authorization PDP - AuthZForce," 31 05 2016. [Online]. Available: http://catalogue.fiware.org/enablers/authorization-pdp-authzforce.

[9] B. Shaffer, "An OAuth2 Server Library for PHP," 2016. [Online]. Available: https://bshaffer.github.io/oauth2-server-php-docs/.

[10] dsquier, "DDL to create MySQL tables for PDO storage support of oauth2-server-php library," [Online]. Available: https://github.com/dsquier/oauth2-server-php-mysql.

[11] Role-based access control, [Online]. Available: https://en.wikipedia.org/wiki/Role-based_access_control

[12] Attribute-based access control, [Online]. Available: https://en.wikipedia.org/wiki/Attribute-Based_Access_Control

[13] Ubuntu server guide, [Online]. Available: https://help.ubuntu.com/lts/serverguide/index.html