INSTALLATION AND CONFIGURATION MANUAL

Role/Title	Name	Signature	Date
Author	Cristina Arcari		23/05/2022
Verified/Approved	DAFNE Team		23/05/2022

Change Register

Version/Rev.	Date	Change	Reason
1.0	15/11/2021	All	First Release
1.1	10/12/2021	Updated sections: 4.1.1, 4.2, 4.3, 4.3.1, 4.4.1, 5.2.4, Appendix A Change `/data/dafne/back-end/config/appconfig.json` in `/data/dafne/back-end/config/config.json` in Appendix A	Errata corrige Installation procedure improvement Changed the services image link Improved description of FE configuration parameters
1.2	01/02/2022	Removed empty lines from section 1.2 Renamed section 4 Improved sections 4.2, 4.3.2, 4.4.1, 4.4.2, 5.2.2, 5.2.3, 5.2.4, 5.3.1, 6.2.1, 6.3, 6.3.1, Appendix A Updated Appendix B with proper DAFNE roles on IDP	Updated for OSF release
1.3	17/03/2022	Added new configuration parameters in Appendix A	Updated for release 1.1.0
1.4	20/04/2022	Updated section 4.3.1 with additional configuration for no proxy environment. Added new parameters for Publication Latency in Appendix A	Updated for release 2.0.0
1.5	23/05/2022	Updated section 4 and Appendix A with installation details.	Updated for release 2.0.2

Table of Contents

1. Introduction 3
1.1. Purpose and Scope 3
1.2. Applicable and Reference Documents 3
1.3. Acronyms 3
2. Document Overview 5
3. Application Overview 6
4. Installation and Configuration Procedure (Docker) 6
4.1. Pre-requirements 6
4.2. Installation procedure 7
4.3. Configuration procedure 8
4.4. Getting Started 12
5. Installation and Configuration Procedure (*Legacy*) 16
5.1. Pre-requirements 16
5.2. Installation procedure 17
5.3. Configuration procedure 20
5.4. Getting Started 21
6. DB Maintenance procedures 23
6.1. DB Backup 23
6.2. DB Restore 23
6.3. DB password update 25
Appendix A – DAFNE configuration files 26
Appendix B – Keycloak configuration for DAFNE 36

1. Introduction

1.1. Purpose and Scope

The purpose of this document is to describe the installation and configuration procedure of the Data Flow Network Environment (*DAFNE*). This document has been prepared in the frame of the Collaborative Data Hub Software Maintenance and Evolution Services for Digital Twin Earth (*hereinafter referred to as the "Service"*).

1.2. Applicable and Reference Documents

ID	Document Title	Reference	Issue
AD-1.	Collaborative Data Hub Software - Maintenance and Evolution Services - Ready for Digital Twin Earth -System Design Document	COPE-SERCO-TN-21-1171	1.0
AD-2.	Statement of Work: COLLABORATIVE DATA HUB SOFTWARE - MAINTENANCE AND EVOLUTION SERVICES - READY FOR DIGITAL TWIN EARTH	ESA-EOPG-EOPGC-SOW-12	1.0

Table 1 Applicable Documents

ID	Document Title	Reference	Issue
RD-1.	DHS Operational Concept Document	COPE-SERCO-TN-21-1174	
RD-2.	DHS Master ICD	Annex_COPE-SERCO-TN-21-1171 (TBW)	
RD-3.	Data Hub Service (DHuS) GDPR Implementation	GAEL-P286-TCN-031	1.3

Table 2 Reference Documents

1.3. Acronyms

Acronym	Description
API	Application Programming Interface
CPU	Central Processing Unit

DB	DataBase
DAFNE	Data Flow Network Environment
DHuS	Data Hub Service
EO	Earth Observation
ESA	European Space Agency
HTTP	Hyper Text Transfer Protocol
IDP	Identity Provider
JSON	JavaScript Object Notation
RAM	Random Access Memory
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
VM	Virtual Machine

2. Document Overview

The overall structure of the document is described below:

- Chapter 1 is an introduction with reference and applicable documents
- Chapter 2 is the document overview
- Chapter 3 contains the system overview
- Chapter 4 describes the installation and configuration procedure based on docker compose
- Chapter 5 describes the installation and configuration procedure in legacy mode
- Chapter 6 contains the DB maintenance procedures
- Appendix A contains the detail of the configuration files
- Appendix B contains Keycloak configuration instruction for DAFNE

3. Application Overview

4. Installation and Configuration Procedure (Docker)

4.1. Pre-requirements

This procedure assumes that all DAFNE components are deployed on one VM.

The recommended SW and HW requirements for the DAFNE VM are:

- OS: Centos7 64 bit
- RAM: 8 GB
- CPU: 4 Cores
- HD: 150 GB

The following software must be pre-installed on each DAFNE VM:

- Docker Engine v.19.03.5 (*API version 1.40*),
- Docker Compose v.1.24.1

The required version of the Docker Compose can be installed following the steps below (*as root or sudoer*):

```
curl -L https://github.com/docker/compose/releases/download/1.24.1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

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

ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

The docker daemon must be active on each VM. Please find below the command useful to check the docker daemon status (*to be executed as root user*):

```
systemctl status docker
```

If the docker daemon status is “inactive”, it must be started using the command (*to be executed as root user*):

```
systemctl start docker
```

4.1.1. Users and groups

The “**dafne**”[1] user must exist on each DAFNE VM and should belong to the “**docker**” group. This step allows the dafne user to run docker and docker-compose commands.

On each DAFNE VM the `/data` folder must exist, owned by `dafne:docker`

[1] It is possible to use an already existing user as long as it belongs to docker group.

This folder will contain (*by default*) the DAFNE persistent data, the configuration files and the log files

> Please consider that the /data folder could be replaced by any other file system or NAS folder. In this case, the management of this folder will be in charge of the Operation Team

4.1.2. Traffic matrix

The DAFNE VM must be able to reach (*and to be reached by*) the Destination (*or Source*) reported in the connectivity matrix below[2]:

Source	Destination	Protocol/Ports
Internet	Front-End	HTTPS/443
Internet	Back -End	HTTPS/443
Front-End	Back-End	HTTPS/3000
Back-End	DB	TCP/5432
Back-End	Keycloak	HTTPS/443
Back-End	Remote DHuS	HTTPS/443

Table 3 DAFNE VM Connectivity Matrix

4.2. Installation procedure

The DAFNE installation procedure requires the access to the release package, directly downloaded from GitHub at https://github.com/DHS-DataflowNetworkEnvironment/DAFNE-Deployment/releases or provided by DAFNE Development Team.
The docker images of DAFNE components are available in the collaborativedhs/dafne public repository on the official Docker registry at https://hub.docker.com/r/collaborativedhs/dafne/tags.

Log in to the DAFNE VM as dafne user.

Download the last DAFNE release as zip on a local folder (*e.g. /data folder, but generally on any folder owned by dafne user*), unzip it and enter the decompressed folder.

Execute the installation script:

```
sh install.sh <version>[3]
```

Check if the docker images of the DAFNE services have been downloaded locally, executing the following command:

```
docker images
```

You should see the following images among the listed images:

[2] Please note that this traffic matrix refers to operational environment
[3] e.g. `sh install.sh 1.0.5`

```
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
<organization>/dafne        fe-<version>        xxxxxxxx            5 days ago          162MB
<organization>/dafne        be-<version>        yyyyyyyy            6 days ago          941MB
```

where <version> is the tag to be installed.

The installation script creates the directories containing the application configuration and log files and copies the default configuration files in the proper folder.
An example of the folders structure is reported in the table below:

Parameter Name	Description
/data/dafne/front-end/config	Contains the DAFNE Front-End configuration files
/data/dafne/front-end/html	Contains the DAFNE Front-End index.html page, to be mounted in case of no-proxy configuration
/data/dafne/back-end/config	Contains the DAFNE Back-End configuration files
/data/dafne/back-end/logs	Used by the DAFNE Back-End to write the proper log files

The installation script creates the symbolic link /data/dafne which points to the installation folder of the current release of DAFNE, i.e. /data/dafne_<version>. The creation of symbolic link allows to restore the previous version of the software more easily.

Please note that this folder structure is the default one, managed by the installation script. If a different folder structure should be used, the creation of this alternative folder structure should be handled manually.
In addition:

- the default configuration files provided in the release package under the *data* folder should be copied manually in the new destination directory
- the symbolic link /data/dafne which points to the installation folder of the current release of DAFNE, i.e. /data/dafne_<version>, must be created manually using the command:
  `ln -s  /data/dafne_<version> /data/dafne`
- the following command must be executed manually from the same folder location of the installation script, to pull the docker images:
  `docker-compose pull`

4.3. Configuration procedure

4.3.1. Docker-compose configuration

The *docker-compose.yml* file is located in the folder <base_dir>/DAFNE-Deployment, where <base_dir> is the local directory where the release package was dowloaded and unzipped (e.g. /data). The content of this file is reported below:

```
version: "3.1"
services:
  fe:
    image: "collaborativedhs/dafne:fe-<version>"
    container_name: fe
    hostname: frontend
    restart: always
```

```
    build: .
    ports:
      - "8080:80"
    networks:
      - dafne
    volumes:
      - /data/dafne/front-end/config:/usr/local/apache2/htdocs/assets/config
      # uncomment the following line in case of deployment without proxy
      #- /data/dafne/front-end/html/index.html:/usr/local/apache2/htdocs/index.html
    depends_on:
      - be
  be:
    image: "collaborativedhs/dafne:be-<version>"
    container_name: be
    hostname: backend
    restart: always
    build: .
    command: bash -c "npx sequelize-cli db:migrate && node index.js"

    ports:
      - "3000:2000"
    environment:
      - NODE_ENV=production
      - CONF_PATH=/usr/src/app/config/
      - LOGS_PATH=/usr/src/app/logs/
    networks:
      - dafne
    volumes:
      # create a 'config' folder in the working dir and bind it to an external folder of the host,
where the config.json file is located
      - /data/dafne/back-end/config:/usr/src/app/config
      # create a 'log' folder in the working dir and binds it to an external folder of the host, to
make the logs persistent
      - /data/dafne/back-end/logs:/usr/src/app/logs
  # uncomment the following lines to use a dockerized image of the DAFNE database
  #  depends_on:
  #    - db
  # db:
  #   container_name: dafne_db
  #   hostname: dafne_db
  #   restart: always
  #   image: "library/postgres:12.5"
  #   env_file:
  #     - /data/dafne/back-end/config/db_credentials.env
  #   networks:
  #     - dafne
  #   ports:
  #     - "5432:5432"
  #   volumes:
  #     - /data/dafne-db/:/var/lib/postgresql/data
networks:
    dafne:
      driver: bridge
      ipam:
        driver: default
        config:
          - subnet: 172.16.0.0/16
```

In the DAFNE default docker-compose file, the section related to the DB service is commented, since it is possible to use an already existing instance of PostgreSQL. In case of need, it is possible to remove the comment in the docker-compose file to use a docker DB service. The docker DB service needs a volume on the hosted VM (/data/dafne-db/). The volume must be created manually.
Furthermore, in case no proxy has been set up in front of DAFNE, please uncomment the line `#-/data/dafne/front-end/html/index.html:/usr/local/apache2/htdocs/index.html` in the fe service section, to grant the proper visualization of the DAFNE web client.

Besides this, there is no need to change the content of this file in the default environment. The coloured and bold parameters could be changed, if necessary, as explained below:

Parameter	Description
8080	This is the Front-End HOST PORT, i.e. the port used to expose the Front-End application. This port can be changed if the default one is used by another service.
3000	This is the Back-End HOST PORT, i.e. the port used to expose the Back-End services. This port can be changed if the default one is used by another service.
5432	This is the DB HOST PORT, i.e. the port used to expose the DB service. This port can be changed if the default one is used by another service (*typically another PostgreSQL service*).
/data/dafne/front-end/config	This folder hosts the Front-End configuration file config.json. It is possible to replace this folder with a different one, if needed. In this case, the config.json provided in the installation kit must be copied in the new folder.
/data/dafne/back-end/config	This folder hosts the Back-End configuration files config.json and db_credentials.env. It is possible to replace this folder with a different one, if needed. In this case, both files provided in the installation kit must be copied in the new folder
/data/dafne/back-end/logs	This folder hosts the Back-End log files. It is possible to replace this folder with a different one, if needed.
/data/dafne-db	This folder hosts the DB PostgreSQL configuration and data files. It is possible to replace this folder with a different one, if needed.
subnet: 172.16.0.0/16	The definition of the subnet parameter grants that the docker swarm will use a specific subnet, instead of any available subnet. The default value 172.16.0.0/16 of the subnet could be changed in conformity with the Operations network environment.

	Please consider that also a default getaway could be defined. In this case the syntax to use should be: subnet: xxx.xxx.xxx.xxx/yy gateway: xxx.xxx.xxx.y

Table 4 docker-compose configuration parameters

The other docker environment variables are described in detail in Appendix A

4.3.2. Application Configuration

Front-End Configuration

On the DAFNE VM , open the Front-End configuration file at `/data/dafne/front-end/config/config.json` and update the following properties as indicated below:

"apiUrl": The value of the properties corresponds to IP address of the Back-End services, in the format `http(s)://<BE_host>:<BE_port>`. The BE_host is the hostname or IP of the BE_VM and the BE_port is the BE HOST port (*the default value is 3000*). In case a proxy has been set up in front of DAFNE, please replace the `apiUrl` with the address used in the proxy to expose the Back-End services. Please also note that, in case a proxy is used, the `/dafne/` endpoint must be used to expose the DAFNE Front-End. Please refer to the proxy documentation for instructions on how to implement the proper rules.

The Front-End configuration file is described in detail in Appendix A

DB Configuration

On the DAFNE VM , open the DB environment file at `/data/dafne/back-end/config/db_credentials.env` and update the POSTGRES_PASSWORD property adding a value for the password of the dafne DB user.
Please consider that the DB user account will be created with these credentials. If you need to change the DB user password, please refer to §6.3

If the docker DB service provided in the DAFNE docker-compose file is used, the PostgreSQL configuration files are located in the folder dedicated to the DB, which, by default is `/data/dafne-db/`.

These files can be accessed and modified only by root or sudoer.

In order to allow PostgreSQL to accept remote connections, the following line should be present in the configuration file *postgresql.conf*[4]*:*

```
listen_address = '*'
```

There should be no need to change the default `pg_hba.conf` file, located in the same directory of the *postgresql.conf* file. The content of the file is reported below:

[4] If the docker DB service provided by DAFNE is used, this file is present in `/data/dafne-db/`, otherwise in the PostgreSQL service config directory

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     trust
host    replication     all             127.0.0.1/32            trust
host    replication     all             ::1/128                 trust

host all all all md5
```

If the docker DB service provided by DAFNE is used, the data and configuration files in the folder `/data/dafne-db` are created only after the first application start.

Back-End Configuration

On the DAFNE_VM , open the Back-End configuration file at `/data/dafne/back-end/config/config.json` and update the following properties as indicated below:

"production.password": You must add the same value of the POSTGRES_PASSWORD property defined in the DB credentials environment file `/data/dafne/back-end/config/db_credentials.env`.
"production.port": You have to change the default value of 5432 only if you modified the DB HOST PORT in the docker-compose file
"keycloakBaseUrl": This is the IDP URL used to obtain the OAuth2 token of DAFNE and/or DHuS users (*a keycloak open-id connect end point*). The procedure useful to configure Keycloak for DAFNE is reported in Appendix B.
"crypto.symmetric.secret": This is the key used by the crypto algorithm to cipher the user password of the configured services accounts. A default value is used if no key is defined[5].
"crypto.symmetric.algorithm": This is the algorithm used to crypt the user password of the configured services accounts. Please use "aes-256-ctr" as value.

The Back-End configuration file is described in detail in Appendix A.

4.4. Getting Started

4.4.1. Start Execution

After executing the instructions provided in 4.2 and 4.3, you are ready to launch DAFNE.

[5] In operational environment a different key should be set. A suitable key is a random sequence of 32 characters.

On the DAFNE VM, from the same directory of the *docker-compose.yml* file, execute the following command:

```
sh start.sh
```

Check if the application is properly started using the command

```
docker ps
```

The expected result should be similar to what is reported below

```
CONTAINER ID        IMAGE                           COMMAND                  CREATED             STATUS              PORTS                    NAMES
524a0a6d2f5c        <organization>/dafne:be-<version>   "docker-entrypoint.s…"   2 days ago          Up 2 days           0.0.0.0:3000->2000/tcp   be
e4217e4b729e        <organization>/dafne:fe-<version>   "httpd-foreground"       2 days ago          Up 2 days           0.0.0.0:80->80/tcp       fe
```

Please note that, in case a docker image of Postgres is used for DAFNE DB, the Postgres image appears in the list as well

Check if there is any error executing the following command:

```
docker logs be -f
docker logs fe -f
```

It may happen that Docker cannot use iptables NAT rules to map a published port to a container service. In this case, you could see a similar error when executing the start.sh script:

Cannot start service xx: driver failed programming external connectivity on endpoint yy: Error starting userland proxy: listen tcp4 0.0.0.0:<port> address already in use.

If such an error occurs, please execute the following steps:

```
sudo ss -lptn 'sport = :<port>'
kill <listed_pids>
```

After performing these commands, please execute again the start.sh script. If the error persists, you need to disable the userland proxy usage on docker. To achieve this, please edit or create the file */etc/docker/daemon.json* as root or sudoer adding the following text:

{"userland-proxy": false}

The restart the docker daemon executing the command *systemctl restart docker* and launch again the start.sh script.
This should fix the issue.

4.4.2. **Monitoring**

In order to check that the application is running, execute the following command:

```
docker ps
```

The expected result should be similar to what is reported below

```
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS                    NAMES
524a0a6d2f5c        aliaspace/dafne:be-1.0.0       "docker-entrypoint.s…"   2 days ago          Up 2 days           0.0.0.0:3000->2000/tcp   be
e4217e4b729e        aliaspace/dafne:fe-1.0.0       "httpd-foreground"       2 days ago          Up 2 days           0.0.0.0:80->80/tcp       fe
```

Please note that, in case a docker image of Postgres is used for DAFNE DB, the Postgres image appears in the list as well

You should be able to see the DAFNE login page on a browser, going to http(s)://<dafne_url>/dafne. After the login, the DAFNE main page should appear.

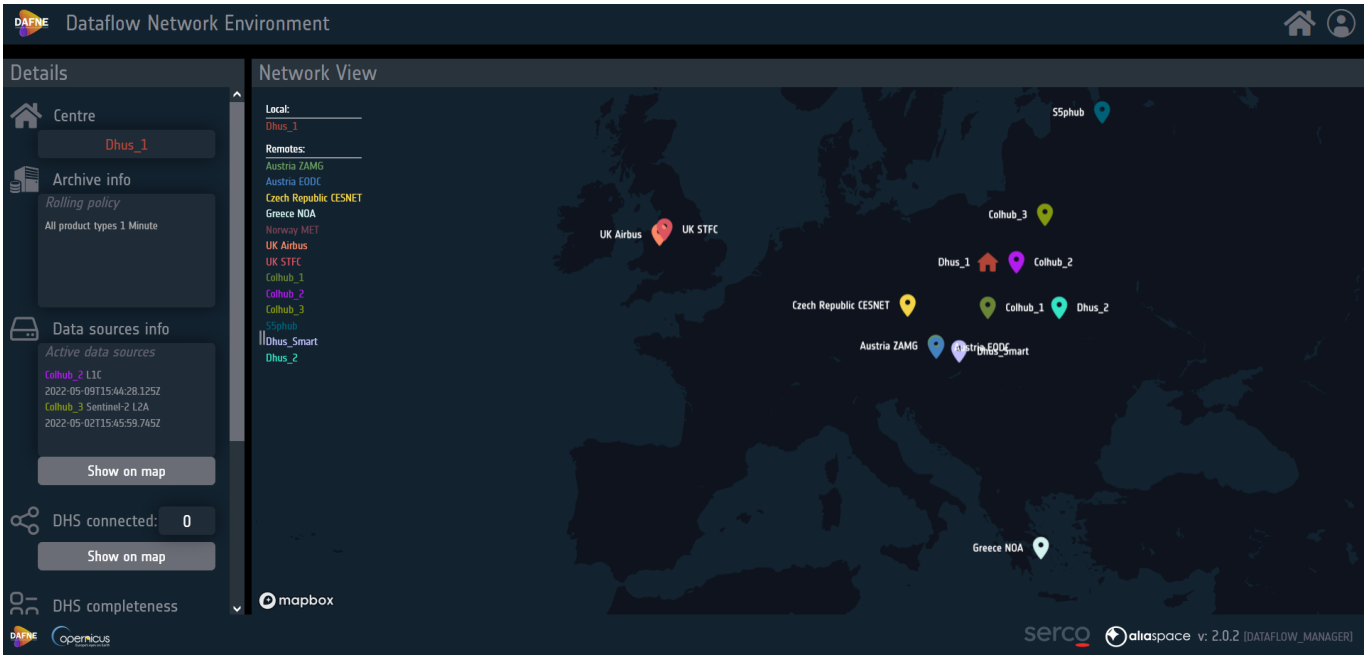

Figure 1 DAFNE Main page

Please note that, in case a previous release of DAFNE was installed, the browser cache should be cleaned to display the proper software version. The same operation should be performed in case of DAFNE logo update, as described in Appendix A.

You can check the BE application logs in 2 different ways:

1. Looking at the log files located, by default, in the folder `data/dafne/back-end/logs`. The default naming convention for log name is:

   `dafne-be-YYYY-MM-DD.log`

   For the current log file, the symbolic link *dafne-be.log* is created under the default log directory.

2. By means of docker service logs, executing the command:

   `docker logs be -f`[6]

[6] Please note that the docker logs of the DAFNE Back-End are not enabled by default. More details about docker logs are available in Appendix A.

If you need to access the psql console of the DAFNE database installed as docker service, execute the following command:

```
docker exec -it <container_id>[7] psql -U dafne
```

4.4.3. Stop Execution

On the DAFNE VM, from the same directory of the *docker-compose.yml* file, execute the following command:

```
sh stop.sh
```

Verify that there is no instance running using the command

```
docker ps
```

The expected result should be like the extract reported below

```
CONTAINER ID        IMAGE                    COMMAND                  CREATED             STATUS              PORTS                    NAMES
```

[7] <container_id> is the identifier of the DB container, retrieved with the command docker ps. It is possible to use the container name as well. The default container name reported in the docker-compose.yml file is dafne-db..

5. Installation and Configuration Procedure (*Legacy*)

5.1. Pre-requirements

This procedure assumes that all DAFNE components are deployed on one VM.

The recommended SW and HW requirements for the DAFNE VM are:

- OS: Centos7 64 bit
- RAM: 8 GB
- CPU: 4 Cores
- HD: 150 GB

The following software must be pre-installed on the DAFNE VM:

- Apache HyperText Transfer Protocol (*HTTP*) server 2.x (*2.4 preferred*)
- Node.js version 12.y.z

The following software must be pre-installed on the DAFNE VM:

- PosgreSQL 12.x (*12.5 preferred*)

More information about how to install the required software can be found at the following links:

- https://www.postgresql.org/download/linux/redhat/ (*for the installation of PostgreSQL*)
- https://nodejs.org/en/download/package-manager/ (*for the installation of Node.js*)
- https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-centos-7 (*for the installation of Apache2*)

The Apache service must be active on the DAFNE VM. Execute the following command to check the status of the service:

```
systemctl status httpd
```

If the Apache service status is “inactive”, it must be started using the command (*to be executed as root user*):

```
systemctl start httpd
```

The PosgreSQL service must be active on the DAFNE VM. Execute the following command to check the status of the service:

```
systemctl status postgresql
```

If the PosgreSQL service status is “inactive”, it must be started using the command (*to be executed as root user*):

```
systemctl start postgresql
```

Check the version of Node.js on the DAFNE VM by executing the following command:

```
node --version
```

The expected result is v12.16.1

5.1.1. Users and groups

The "**dafne**" user must exist on the DAFNE VMs.

On each DAFNE VM the following folders must exist, owned by dafne user:

`/data` folder
`/opt/dafne/back-end` folder

> Please consider that the /data folder could be replaced by any other file system or NAS folder. The management of this folder should be in charge of the Operation Team

5.1.2. Traffic matrix

Please refer to paragraph $4.1.2

5.2. Installation procedure

5.2.1. Database Installation

The following steps must be executed on the DAFNE VM.

After the PostgreSQL service installation, the database must be initialized:

```
postgresql-setup initdb
```

The default directory where PostgreSQL will store data is /var/lib/postgresql/data[8]
Execute the following commands in order to enable the PostgreSQL service:

```
systemctl enable postgresql.service
systemctl start postgresql.service
```

Login to PostgreSQL console with postgres user:

```
psql -U postgres
```

[8] Please note that the path may vary on different OS

Execute the following commands to create the user and database for testing:

```
create user dafne with encrypted password '<dafne_password>';
CREATE DATABASE dafne;
grant all privileges on database dafne to dafne;
ALTER DATABASE dafne OWNER TO dafne;
```

You should be able to access the database using the command:

```
psql -h 127.0.01 -U dafne -d dafne
```

5.2.2. Back-End Installation

The DAFNE Back-End installation procedure requires the access to the release package, directly downloaded from GitHub at https://github.com/DHS-DataflowNetworkEnvironment/DAFNE-Back-End/releases or provided by DAFNE Development Team.

Log in to the DAFNE VM as dafne user.

Download the last DAFNE Back-End release as zip on a local folder (*e.g. /data folder, but generally on any folder owned by dafne user*), unzip it, enter the decompressed folder and copy the content of the src directory into the `/opt/dafne/back-end` folder:

```
cp -R src/* /opt/dafne/back-end/
```

Access the installation directory at `/opt/dafne/back-end/` and open the legacy_start.sh script. The file should look like this:

```
export NODE_ENV=production

export CONF_PATH=/data/dafne/back-end/config/
export LOGS_PATH=/data/dafne/back-end/logs/

bash -c "npx sequelize-cli db:migrate && node index.js"
```

As you can see, the file contains the definition of some environment variables, which are the one also used in the docker based installation, with the same meaning:

Variable Name	Description
NODE_ENV	Back-End environment type. Possible values are: - development - test - production The default value is production
CONF_PATH	Configuration file path: The default value is `/data/dafne/back-end/config/`
LOG_PATH	Log files path. The default value is `/data/dafne/back-end/logs/`

5.2.3. Front-End Installation

The DAFNE Front-End installation procedure requires the access to the release package, directly downloaded from GitHub at https://github.com/DHS-DataflowNetworkEnvironment/DAFNE-Front-End/releases or provided by DAFNE Development Team.

Log in to the DAFNE VM as dafne user.

Download the last DAFNE Front-End release as zip on a local folder (*e.g. /data folder, but generally on any folder owned by dafne user*), unzip it, enter the decompressed and, as root or sudoer user, copy the content of the directory into the default apache folder `/usr/local/apache2/htdocs`:

```
cp -R * /usr/local/apache2/htdocs
```

Create a file on the apache folder `/usr/local/apache2/htdocs`, named `.htaccess` and copy the following rows in the file:

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} -f [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^ - [L]
RewriteRule ^ index.html [L]
```

Save the file and restart the Apache service to save the changes:

```
systemctl restart httpd
```

5.2.4. Environment Installation

The DAFNE installation procedure requires the access to the release package, directly downloaded from GitHub at https://github.com/DHS-DataflowNetworkEnvironment/DAFNE-Deployment/releases or provided by DAFNE Development Team.

Log in to the DAFNE VM as dafne user.

Download the last DAFNE release as zip on a local folder (*e.g. /data folder, but generally on any folder owned by dafne user*), unzip it and enter the decompressed folder.

Execute the installation script:

```
sh legacy_install.sh <version>[9]
```

The installation script creates the directories containing the application configuration and log files and copies the default configuration files in the proper folder.
An example of the folders structure is reported in the table below:

[9] e.g. `sh install.sh 1.0.0`

Parameter Name	Description
/data/dafne/front-end/config	Contains the DAFNE Front-End configuration files
/data/dafne/back-end/config	Contains the DAFNE Back-End configuration files
/data/dafne/back-end/logs	Used by the DAFNE Back-End to write the proper log files

The installation script creates the symbolic link `/data/dafne` which points to the installation folder of the current release of DAFNE, i.e. `/data/dafne_<version>`. The creation of symbolic link allows to restore the previous version of the software more easily.

Please note that this folder structure is the default one, managed by the installation script. If a different folder structure should be used, the creation of this alternative folder structure should be handled manually.
In addition:

- the default configuration files provided in the release package under the *data* folder should be copied manually in the new destination directory
- the symbolic link `/data/dafne` which points to the installation folder of the current release of DAFNE, i.e. `/data/dafne_<version>`, must be created manually using the command:
  `ln -s  /data/dafne_<version> /data/dafne`

5.3. Configuration procedure

5.3.1. Application Configuration

DB Configuration

On the DAFNE _VM, in order to allow PostgreSQL to accept remote connections, please check that the following line is present in the configuration file `/var/lib/postgresql/data/postgresql.conf` (*you must be root to change this file*):

```
listen_address = '*'
```

There should be no need to change the default pg_hba.conf file, located in `/var/lib/postgresql/data`.

Restart service after making configuration changes, with the command

```
systemctl restart postgresql.service
```

Back-End Configuration

On the DAFNE_VM , open the Back-End configuration file at `/data/dafne/back-end/config/appconfig.json` and update the following properties as indicated below:

"production.password": You must add the same value of the POSTGRES_PASSWORD property defined in the DB credentials environment file `/data/dafne/back-end/config/db_credentials.env`.

"production.port": You have to change the default value of 5432 only if you modified the DB HOST PORT in the docker-compose file
"keycloakBaseUrl": This is the IDP URL used to obtain the OAuth2 token of DAFNE and/or DHuS users (*a keycloak open-id connect end point*). The procedure useful to configure Keycloak for DAFNE is reported in Appendix B.
"crypto.symmetric.secret": This is the key used by the crypto algorithm to cipher the user password of the configured services accounts. A default value is used if no key is defined[10].
"crypto.symmetric.algorithm": This is the algorithm used to crypt the user password of the configured services accounts. Please use "aes-256-ctr" as value.

The Back-End configuration file is described in detail in Appendix A.

Front-End Configuration

On the DAFNE VM , open the Front-End configuration file at `/data/dafne/front-end/config/config.json` and update the following properties as indicated below:

"apiUrl": The value of the properties corresponds to IP address of the Back-End services, in the format `http(s)://<BE_host>:<BE_port>`. The BE_host is the hostname or IP of the BE_VM and the BE_port is the BE HOST port (*the default value is 3000*).
In case a proxy has been set up in front of DAFNE, please replace the `apiUrl` with the address used in the proxy to expose the Back-End services. Please also note that, in case a proxy is used, the /dafne/ endpoint must be used to expose the DAFNE Front-End. Please refer to the proxy documentation for instructions on how to implement the proper rules.

The Front-End configuration file is described in detail in Appendix A

5.4. Getting Started

5.4.1. Start Execution

After executing the instructions provided in 5.2 and 5.3, you are ready to launch the DAFNE application.

On the DAFNE_VM, access the installation directory at `/opt/dafne/back-end/` and execute the following commands:

```
sh legacy_start.sh > nohup.txt 2>&1 &
```

Check if the application is properly started using the command

```
ps -ef | grep "node index.js"
```

The result should be similar to the output reported below:

```
dafne+ 17252 17251  1 17:56 pts/3    00:00:00 node index.js
```

[10] In operational environment a different key should be set. A suitable key is a random sequence of 32 characters.

On the DAFNE_VM, restart the Apache service to finalize the installation (*as root or sudoer*):

```
systemctl restart httpd
```

5.4.2. Monitoring

You can check the BE application logs looking at the log files located, by default, in the folder `/data/dafne/back-end/logs`. The default naming convention for log name is:

```
dafne-be-YYYY-MM-DD.log
```

For the current log file, the symbolic link *dafne-be.log* is created under the default log directory.

5.4.3. Stop Execution

On the DAFNE_VM, stop the Apache service to terminate the application:

```
systemctl stop httpd
```

As dafne user, stop the BE application executing the following command:

```
kill `ps -ef | grep "node index.js" | awk '{print $2}'`
```

Verify that there is no BE instance running using the command

```
ps -ef | grep "node index.js"
```

6. DB Maintenance procedures

6.1. DB Backup

6.1.1. Docker installation

DAFNE must be active to perform the DB backup procedure.

On the DAFNE VM (*or more generally the VM where the container of the DB is running*), execute the command:

```
docker ps
```

The output should be similar to the lines below

```
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS                    NAMES
524a0a6d2f5c        aliaspace/dafne:be-1.0.0       "docker-entrypoint.s…"   2 days ago          Up 2 days           0.0.0.0:3000->2000/tcp   be
e4217e4b729e        aliaspace/dafne:fe-1.0.0       "httpd-foreground"       2 days ago          Up 2 days           0.0.0.0:80->80/tcp       fe
ac77ad51d2c3        postgres:12.5                  "docker-entrypoint.s…"   2 weeks ago         Up 2 days           0.0.0.0:5432->5432/tcp   db
```

Get the CONTAINER ID of the postgres:12.5 image and execute the following command on the DAFNE VM (or more generally the VM where the container of the DB is running):

```
docker exec -it <container_id> pg_dump --column-inserts --data-only -U dafne > <dump_folder>/db_dump_<YYYYMMDD>.sql
```

where:

- `<container_id>` is the identifier of the DB container, retrieved with the command docker ps
- `<dump_folder>` is the folder where to save the dump file
- `<YYYYMMDD>` is the current date

6.1.2. Legacy installation

On the VM where the DB is installed, execute the command:

```
pg_dump --column-inserts -U dafne  > <dump_folder>/db_dump_<YYYYMMDD>.sql
```

where:

- `<dump_folder>` is the folder where to save the dump file
- `<YYYYMMDD>` is the current date

6.2. DB Restore

6.2.1. Docker installation

This procedure requires that the DB volume on the host VM must be empty before starting DAFNE. For instance, if the DB volume is a folder on the host VM, this folder must exist, with no files or subfolder

inside. Please refer to 4.3.1 to know how to configure the DB volume. DAFNE must be running to perform the DB restore procedure. Indeed, the dump procedure described in 6.1 affects only data, while the DB schema and objects are created by DAFNE Back-End at start up, if they do not exist.

On the DAFNE_VM (or more generally the VM where the container of the DB is running), execute the command:

```
docker ps
```

The output should be similar to the lines below

```
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS                      NAMES
524a0a6d2f5c        aliaspace/dafne:be-1.0.0       "docker-entrypoint.s…"   2 days ago          Up 2 days           0.0.0.0:3000->2000/tcp     be
e4217e4b729e        aliaspace/dafne:fe-1.0.0       "httpd-foreground"       2 days ago          Up 2 days           0.0.0.0:80->80/tcp         fe
ac77ad51d2c3        postgres:12.5                  "docker-entrypoint.s…"   2 weeks ago         Up 2 days           0.0.0.0:5432->5432/tcp     db
```

Get the CONTAINER ID of the postgres:12.5 image and execute the following commands on the DAFNE_VM (or more generally the VM where the container of the DB is running):

```
docker cp <dump_folder>/<dump_file_name> <container_id>:/<dump_file_name>
```

```
docker exec -it <container_id> psql -U dafne -c 'delete from centres;'[11]
```

```
docker exec -it <container_id> psql -U dafne -f <dump_file_name>
```

where:

- `<container_id>` is the identifier of the DB container, retrieved with the command docker ps
- `<dump_folder>` is the folder where the dump file is located
- `<dump_file_name>` is the name of the dump file

An alternative command is:

```
cat <dump_folder>/<dump_file_name> | docker exec -i <container_id> psql -U dafne
```

Please ignore the possible errors logged during the execution, because they are due to the dictionary tables which are contained in the dump and in the new "empty" DB

6.2.2. **Legacy installation**

On the VM where the new DB is installed, execute the command:

```
psql -U dafne -f <dump_file_name>
```

where:

- `<dump_file_name>` is the full name (with the path) of the dump file

[11] This command allows to replace the pre-configured centres with the ones contained in the dump file, avoiding conflicts

Please note that the new DB must be created, following the steps described in §5.2.1

6.3. DB password update

The following procedure is useful if there is the need of updating the password of the DB user. Please note that this implies to change the DB password also in the DAFNE Back End configuration file accordingly.

6.3.1. Docker installation

DAFNE must be running to perform this procedure. On the DAFNE_VM (or more generally the VM where the container of the DB is running), execute the command:

docker ps

The output should be similar to the lines below

```
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS                    NAMES
524a0a6d2f5c   aliaspace/dafne:be-1.0.0  "docker-entrypoint.s…"   2 days ago       Up 2 days       0.0.0.0:3000->2000/tcp   be
e4217e4b729e   aliaspace/dafne:fe-1.0.0  "httpd-foreground"       2 days ago       Up 2 days       0.0.0.0:80->80/tcp       fe
ac77ad51d2c3   postgres:12.5             "docker-entrypoint.s…"   2 weeks ago      Up 2 days       0.0.0.0:5432->5432/tcp   db
```

Get the CONTAINER ID of the postgres:12.5 image and execute the following commands on the DAFNE_VM (or more generally the VM where the container of the DB is running):

```
docker exec -it <container_id> psql -U dafne
```

where:

- `<container_id>` is the identifier of the DB container, retrieved with the command docker ps

From the psql shell, execute the following command:

```
ALTER USER dafne WITH PASSWORD 'yournewpass';
```

Exit from the psql shell executing `\q`

6.3.2. Legacy installation

On the VM where the new DB is installed, execute the command:

```
psql -U dafne
```

From the psql shell, execute the following command:

```
ALTER USER dafne WITH PASSWORD 'yournewpass';
```

Exit from the psql shell executing `\q`

Appendix A – DAFNE configuration files

Docker environment variables

In the table below are described the main environment variables defined in the DAFNE *docker-compose.yml* file, which is used to configure the DAFNE application's services, i.d. the Back-End the Front-End and the Database.

This file is used by the docker-compose command to start/stop the services defined in it.

In the default environment, there is no need to change the docker-compose file.

Any update to the file requires an application restart.

Parameter Name	Description
NODE_ENV	Back-End environment type. Possible values are: - development - test - production The default value is production. Please use a different value (test or development) to check DAFNE Back-End logs via docker logs.
CONF_PATH	Configuration file path used internally by the Back-End docker container. The default value is `/usr/src/app/config/`
LOG_PATH	Log files path used internally by the Back-End docker container. The default value is `/usr/src/app/logs/`

Front-End configuration file

In the table below are described the configuration parameters of the DAFNE Front-End, contained in the */data/dafne/front-end/config/config.json* file. Please consider that the parameters in bold are those which needs to be updated to allow the correct functioning of the DAFNE Front-End.

Any update to the file requires a refresh of the browser page with the DAFNE Front-End web application.

Section/Parameter Name	Description
"apiUrl"	Back-End base URL which is used as API entry point, in the format `http(s)://<BE_host>:<BE_port>`. The BE_host is the hostname or IP of the BE_VM and the BE_port is the BE HOST port configured in 4.3.1 In case a proxy has been set up in front of DAFNE, please replace the `apiUrl` with the address used in the proxy to expose the Back-End services.

"version"		Front-End version. The default value corresponds to the current version of the application installed.
" centreBackupLogoPath "		The link to the default center logo image in png format. There is no need to change this parameter. The default value is `assets/images/Logo_DAFNE_white.png`
" centreLogoPath"		The link to the center logo image. The link can be an URL or it is possible to put the image in png format in the `/data/dafne/front-end/config/` folder. The value must be in the format `assets/config/<image_name>.png`
" mapSettings.textSize"		The font size used in the map view. The default value is 12.
" mapSettings.iconSize"		The size (in pixels) of the centres icons used in the map view. The default value is 30.
"dataRefreshTime"		The frequency of data refresh on DAFNE home page. The value is in milliseconds. The default value is 300000 (5 minutes)
“availabilityColors”		Array of JSON objects containing the color palette associated to the availability thresholds. The list can be updated to add new thresholds or to change the palette. Please note that, if the changes are not visible after the page refresh, it could be necessary to clean the web browser cache or to access as incognito user. Each JSON object property is detailed below.
“availabilityColors”	“threshold”	Availability threshold, expressed as the percentage of successful requests on total requests performed to check the availability
“availabilityColors”	“color”	Hexadecimal color corresponding to the availability threshold
“averageAvailabilityColor”		Hexadecimal color used to represent the average availability. Please note that, if the changes are not visible after the page refresh, it could be necessary to clean the web browser cache or to access as incognito user.
“latencyColors”		Array of JSON objects containing the color palette associated to the latency thresholds. The list can be updated to add new thresholds or to change the palette. Please note that, if the changes are not visible after the page refresh, it could be necessary to clean the web browser cache or to access as incognito user. Each JSON object property is detailed below.
“latencyColors”	“threshold”	Latency threshold, expressed in milliseconds (translated as hours in the DAFNE FE).
“latencyColors”	“color”	Hexadecimal color corresponding to the latency threshold
“satelliteList”		Array of JSON objects containing the list of satellite missions and product types shown in the completeness

		panel. The list can be updated to manage new missions and/or product types or to exclude missions and/or product types from the computing of the product completeness. Each JSON object property is detailed below
“satelliteList”	“name”	Platform name in extended format. This is the value shown in the *Mission* filter of the completeness panel (e.g., Sentinel-1)
“satelliteList”	“acronym”	Platform name in short format. This is the value used in the OData filter to find the products matching the *Mission* filter selected in the completeness panel. (e.g., S1)
“satelliteList”	“productType”	List of product types of the related platform name. Please note that these values are used in the OData filter to find the products matching the *Product Type* filter selected in the completeness panel. The default list of product types for Sentinel-1 platform is ["RAW", "SLC", "GRD", "OCN"]
“satelliteList”	“platform”	List of platform number of the related platform name. Please note that these values are used in the OData filter, in combination with the acronym, to find the products matching *Mission + Platform Number* filters selected in the completeness panel. In order to search more than one platform number of a platform, the following convention is used: “A + B”. In case of missions with only a platform number (e.g., Sentinel-5P), the value --- is used to indicate that the platform number must be ignored. The default list of platform numbers for Sentinel-1 platform is ["A", "B", "A+B"].

The full content of the configuration file is reported below:

```
{
    "centreBackupLogoPath": "assets/images/Logo_DAFNE_white.png",
    "centreLogoPath": "",
    "mapSettings": {
      "textSize": 12,
      "iconSize": 30
    },
    "apiUrl": "<back-end-url>",
    "version": "1.1.0",
    "dataRefreshTime": 300000,
    "latencyColors": [
        {"threshold": 259200000, "color": "#ff0000"},
        {"threshold": 129600000, "color": "#ffc000"},
        {"threshold": 43200000, "color": "#ffff00"},
        {"threshold": 14400000, "color": "#92d050"},
        {"threshold": 0, "color": "#00b050"}
    ],
    "availabilityColors": [
        {"threshold": 0.0, "color": "#ff0000"},
        {"threshold": 75.0, "color": "#ffc000"},
        {"threshold": 95.0, "color": "#ffff00"},
```

```
      {"threshold": 98.0, "color": "#92d050"},
      {"threshold": 99.0, "color": "#00b050"},
      {"threshold": 100.0, "color": "#00b050"}
  ],
  "averageAvailabilityColor": "#00d0ff",
  "satelliteList": [
    {
      "name": "Sentinel-1",
      "acronym": "S1",
      "productType": [
        "RAW",
        "SLC",
        "GRD",
        "OCN"
      ],
      "platform": [
        "A",
        "B",
        "A+B"
      ]
    },
    {
      "name": "Sentinel-2",
      "acronym": "S2",
      "productType": [
        "MSIL1C",
        "MSIL2A"
      ],
      "platform": [
        "A",
        "B",
        "A+B"
      ]
    },
    {
      "name": "Sentinel-3",
      "acronym": "S3",
      "productType": [
        "OL_1_EFR___",
        "OL_1_ERR___",
        "SL_1_RBT___",
        "SR_1_SRA___",
        "SR_1_SRA_A_",
        "SR_1_SRA_BS",
        "OL_2_LFR___",
        "OL_2_LRR___",
        "SL_2_LST___",
        "SL_2_FRP___",
        "SY_2_SYN___",
        "SY_2_AOD___",
        "SY_2_VGP___",
        "SY_2_VG1___",
        "SY_2_V10___",
        "SR_2_LAN___"
      ],
      "platform": [
        "A",
        "B",
        "A+B"
      ]
    },
    {
      "name": "Sentinel-5P",
```

```
        "acronym": "S5P",
        "productType": [
          "L1B_RA_BD1",
          "L1B_RA_BD2",
          "L1B_RA_BD3",
          "L1B_RA_BD4",
          "L1B_RA_BD5",
          "L1B_RA_BD6",
          "L1B_RA_BD7",
          "L1B_RA_BD8",
          "L1B_IR_UVN",
          "L1B_IR_SIR",
          "L2__O3____",
          "L2__O3_TCL",
          "L2__O3__PR",
          "L2__NO2___",
          "L2__SO2___",
          "L2__CO____",
          "L2__CH4___",
          "L2__HCHO__",
          "L2__CLOUD_",
          "L2__AER_AI",
          "L2__AER_LH",
          "L2__NP_BD3",
          "L2__NP_BD6",
          "L2__NP_BD7",
          "AUX_CTMANA",
          "AUX_CTMFCT"
        ],
        "platform": [
          "___"
        ]
      }
    ]
  }
```

Back-End configuration files

In the table below are described the parameters of the files `/data/dafne/back-end/config/db-credentials.env` containing the credentials of the DB user account. Please consider that the parameter in bold are those which needs to be updated to allow the correct functioning of the DAFNE Back-End.

This file must be updated only when the DB is created from scratch. Any subsequent update will be ignored, since the DB user account already exists.

Parameter Name	Description
"POSTGRES_USER"	DB username account. The default value is dafne
"**POSTGRES_PASSWORD**"	DB password account.
"POSTGRES_DB"	DB name. The default value is dafne

In the table below are described the configuration parameters of the DAFNE Back-End, contained in the `/data/dafne/back-end/config/config.json` file. Please consider that the parameters in bold are those which needs to be updated/checked to allow the correct functioning of the DAFNE Back-End.

Any update to the file does not require a restart of the application, with the exception of the purgeSchedule parameters, which requires a restart of the application, since they are sensitive parameters affecting data retention.

Section Name	Parameter Name	Description
"logger"	"zippedArchive"	Set to true to zip the archived log files, false otherwise (default is false)
	"dateFormat"	The format of the date written for each log entry. The default is YYYY-MM-DD HH:mm:ss,SSS
	"maxSize"	Maximum size of the file after which it will rotate. This can be a number of bytes, or units of kb, mb, and gb. If using the units, add 'k', 'm', or 'g' as the suffix. The units need to directly follow the number. The default is 50m
	"maxFiles"	Maximum number of logs to keep. If not set, no logs will be removed. This can be a number of files or number of days. If using days, add 'd' as the suffix. By default, this parameter is not present, in order to avoid the log files rolling.
	"severity"	Minimum log severity written in the log file. The default value is info. The other allowed values (case sensitive) are: error, warn, debug.
	"logname"	Log file name. The default value is dafne-be-%DATE%.log
	"datePattern"	A string representing the date format to be used for rotating. Default is YYYY-MM-DD
	"createSymlink"	Create a tailable symlink to the current active log file. The parameter is optional and not present in the default configuration file, since the creation of the tailable symlink of the active log is managed by default. It is needed only if the creation of the tailable symlink must be disabled. In this case, the value to use for this parameter is false.
	“symlinkName”	The name of the tailable symlink. (default: dafne-be.log). The parameter is optional and not present in the default configuration file. It is needed only if the default name must be changed.
"production[12]"	"username"	Database user in production environment. The default value is dafne
	"password"	Database password in production environment. To be set in production environment only. Please note that the default environment is production.
	"database"	Database name in production environment. The default value is dafne.

[12] The value of this property depends on the value of the NODE_ENV variable set in the docker-compose file (or in the legacy_start.sh script, for legacy installation). For example, if the value of the NODE_ENV variable is set to “test”, the value of this key must be set to “test” accordingly.

	"host"	Database host in production environment Please consider that, in the docker based installation, this is the service name used by the docker container, while in the legacy installation this is the DB hostname or IP.
	"port"	Database port in production environment Please consider that, in the docker based installation, this is the internal port used by the docker container, while in the legacy installation this is the DB port.
	"dialect"	Database dialect in production environment. The default value is postgres.
	"operatorAliases"	Allow the usage of database operator aliases in production environment. The default value is false.
"version"		Back-End version. The default value corresponds to the current version of the application installed
"port"		Back-End port used in the docker container. The default value is 2000. Please note that in legacy deployment, this is the Back-End listening port, which value should match the Back-End HOST PORT of the docker deployment (default: 3000).
"adminRole"		The name of the role defined as administrator in the DAFNE IDP. By default, the administrator role is DATAFLOW_MANAGER. It should be changed only if, for some reason, the customer needs to define a different name for the Administrator role.
"crypto"	"**symmetric.secret**"	This is the key used by the crypto algorithm to cipher the password of the configured service accounts. A default value is used if no key is defined
	"symmetric.algorithm"	This is the algorithm used to crypt the password of the configured service accounts. Please use "aes-256-ctr" as value
"dataSourceStatus"		The list of DHuS Synchronizers status used as filter for active data sources info computing. Please note that only synchronizers having a status included in the list will be considered for the computing of active data sources. The default value of the parameter is ["RUNNING", "PENDING"]
"requestTimeout"		The HTTP request timeout in milliseconds. The default value is 30000
"auth"	"**keycloakBaseUrl**"	This is the IDP URL used to obtain the OAuth2 token of DAFNE and/or DHuS users (A keycloak

		open-id connect end point). The procedure useful to configure Keycloak for DAFNE is reported in Appendix B. The default value is “https://<keycloak_url>/auth/realms/dhus/protocol/openid-connect”
	"clientId"	This is the IDP client id of the realms used for DAFNE users. The default value is “dafne”
	"grantType"	This is the OpenID connect protocol grant type of the realms used to obtain the OAuth2 token of DAFNE and/or DHuS users. The default value is “password”
“availability”	"schedule"	Availability schedule in crontab syntax, used to set the frequency check on the availability of the local centre FE service. The default frequency is 10 minutes ("*/10 * * * *").
	"enablePurge"	This parameter is used to enable or disable the rolling of old availability measures form DAFNE DB. The default value is true (roll enabled).
	"purgeSchedule"	Availability rolling schedule in crontab syntax, used to set the frequency check on the rolling of the old availability measures. The default frequency is once a day at 1 AM ("0 1 * * *").
	"rollingPeriodInDays"	This parameter indicates the rolling window, in days, of the availability measures to keep in the DAFNE DB. The default value is 90.
	"url"	This parameter contains the request URL used to check the availability of the local centre FE service. The default value is "odata/v1/Products?$top=1"
“latency”	"schedule"	Latency schedule in crontab syntax, used to set the frequency check on the latency of the local centre BE service. The default frequency is 1 hour ("0 * * * *").
	"enablePurge"	This parameter is used to enable or disable the rolling of old latency measures form DAFNE DB. The default value is true (roll enabled).
	"purgeSchedule"	Latency rolling schedule in crontab syntax, used to set the frequency check on the rolling of the old latency measures. The default frequency is once a day at 1 AM ("0 1 * * *").
	"rollingPeriodInDays"	This parameter indicates the rolling window, in days, of the latency measures to keep in the DAFNE DB. The default value is 90.
	"feRetrySchedule"	Latency retry schedule, in crontab syntax, used to set the frequency check, on local FE service, of products used to compute the latency and not yet

		found on local FE service. The default frequency is 10 minutes ("*/10 * * * *").
	"feMaxRetry"	This parameter contains the maximum number of retries useful to retrieve a product used to compute the latency on the local FE service. The default value is 10.
	"tolerance"	This parameter contains the tolerance, in seconds, applied to the CreationDate in the odata filter used to retrieve the last product synchronized on the referenced source. The default value is 1 (second). Decimal values are allowed.

The full content of the configuration file is reported below:

```
{
    "logger": {
          "zippedArchive": false,
          "dateFormat": "YYYY-MM-DD HH:mm:ss,SSS",
          "maxSize": "50m",
          "severity": "info",
          "logname": "dafne-be--%DATE%.log",
          "datePattern": "YYYY-MM-DD"
    },
    "production": {
          "username": "dafne",
          "password": "xxx",
          "database": "dafne",
          "host": "localhost",
          "port": "5432",
          "dialect": "postgres",
          "operatorsAliases": false
    },
          "crypto": {
                    "symmetric": {
                              "secret": "",
                              "algorithm": ""
                    }
          },
          "auth": {
                    "keycloakBaseUrl": "https://<keycloak_url>/auth/realms/dhus/protocol/openid-
connect",
                    "clientId": "dafne",
                    "grantType": "password"

          },
          "availability": {
                    "schedule": "*/10 * * * *",
                    "enablePurge": true,
                    "purgeSchedule": "0 1 * * *",
                    "rollingPeriodInDays": 90,
                    "url": "odata/v1/Products?$top=1"
          },
          "latency": {
                    "schedule": "0 * * * *",
```

```
                "enablePurge": true,
                "purgeSchedule": "0 1 * * *",
                "rollingPeriodInDays": 90,
                "feRetrySchedule": "*/5 * * * *",
                "feMaxRetry": 10,
                "tolerance": 1
            },
            "requestTimeout": 30000,
            "dataSourceStatus": ["RUNNING", "PENDING"],
            "version": "1.0.0-SNAPSHOT",
            "port": 2000,
            "adminRole": "DATAFLOW_MANAGER"---
}
```

Appendix B – Keycloak configuration for DAFNE

The following procedure assumes that both the DHuS and DAFNE will use the same realm in Keycloak. if this is not the case, the configuration of a client for the DHuS can be skipped.

Add a new realm in Keycloak

Note: If Keycloak has been already configured as DHuS IDP and a DHuS realm already exists, please skip this step.

1. Open a browser and go to the Keycloak URL
2. Click on Administration Console and login with the Keycloak admin account
3. Click on the realm selection drop menu located in the upper left corner of the page. Click on Add Realm and name it as 'dhus'.
4. In the “Tokens” tab, add a proper token session duration in the “SSO Session Max” option, if you want to define a proper session duration.

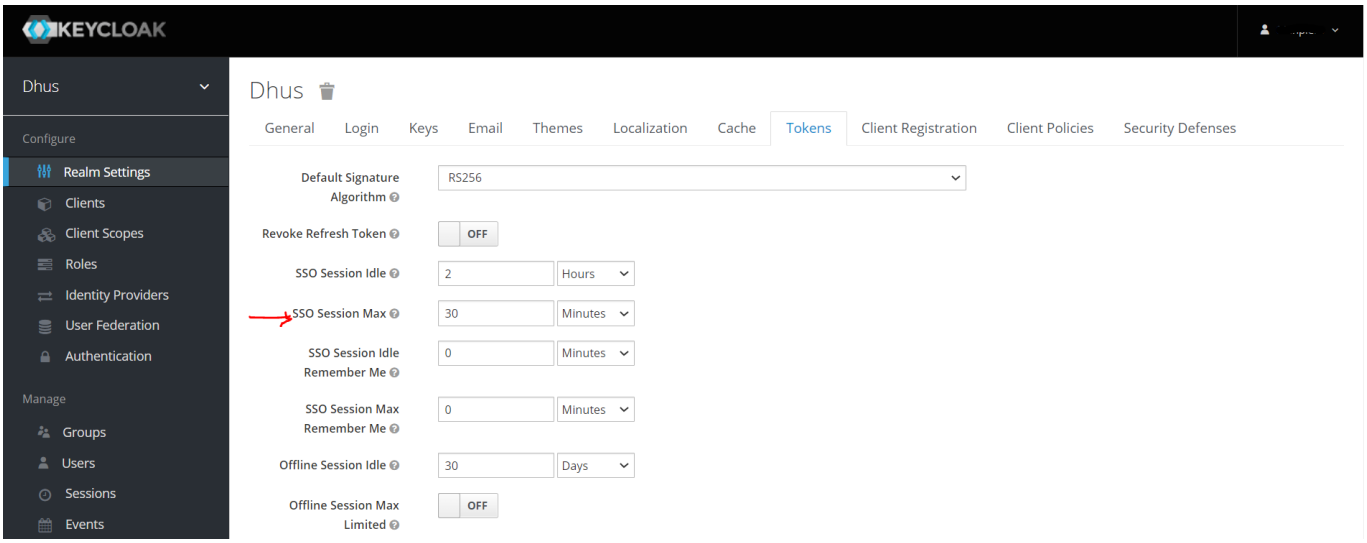

Figure 2 Token duration settings

Create a client for the DHuS

Please follow the instructions reported in the §§4.3, 4.5, 4.6 and 4.7 of RD-3 to configure properly a client for the DHuS. Please note that it is possible to skip this step if a client for the DHuS already exists in the Keycloak realm or in case there is no need to share users between the DHuS and DAFNE

Create a client for DAFNE in Keycloak

Execute the following steps in the dhus realm (or in the proper realm created for DAFNE):

1. In the menu on the left, click on Clients, then, in the “Clients” page, click on Create to add a client. Set ‘dafne as Client ID and ‘openid-connect as Client protocol.
2. In the new client “Settings” tab, verify that the Enabled option is set to “ON”.
3. In the new client “Roles” tab, add the following roles, clicking on the “Add Role” button:

a. DATAFLOW_VIEWER Allows a user to access centre metrics (rolling policies, active datasources info, completeness)
b. DATAFLOW_MANAGER Allows a user to perform the action of USER role and, in addition, allow to configure DHS centres, services and local synchronizers and to access centre metrics (rolling policies, active datasources info, completeness)

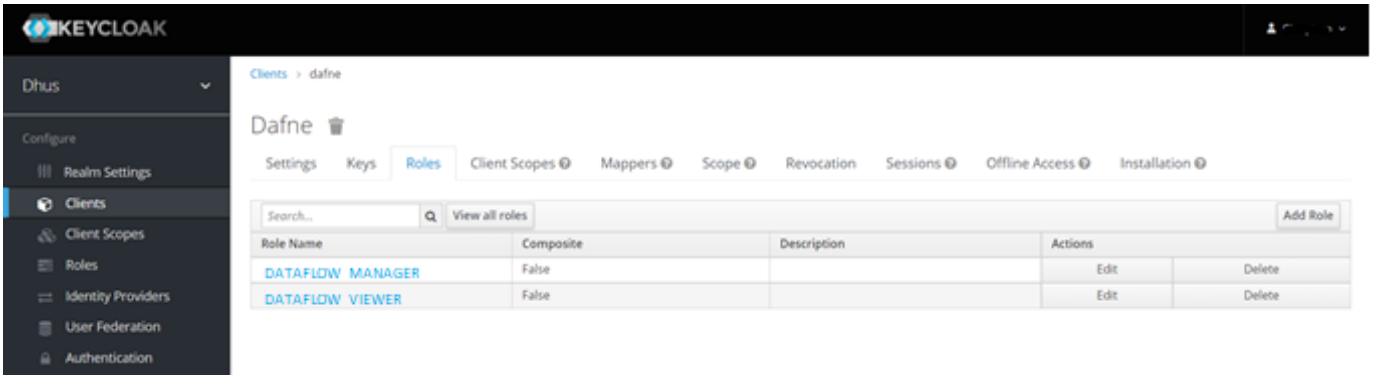

Figure 3 DAFNE Roles

Create a default role for DAFNE in Keycloak

Execute the following steps in the dhus realm (or in the proper realm created for DAFNE):

1. In the menu on the left, click on Roles, then select the "Default Roles" tab.
2. In the "Default Roles" tab, select "dafne" from the "Client Roles" drop down menu, then select the 'DATAFLOW_VIEWER' role and click on "Add selected". This allows to assign "DATAFLOW_VIEWER" as default role for each new user created or already existing in the realm.

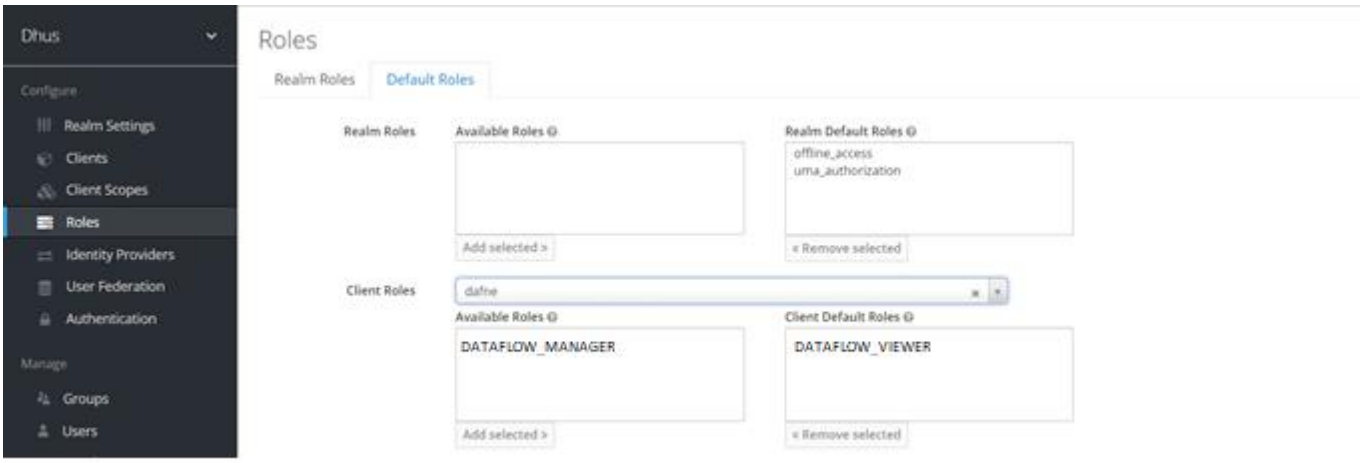

Figure 4 Define DAFNE default role

Create Users for DAFNE in Keycloak

Please note that if you have already created users in the realm, you can skip these steps.

1. In the menu on the left, click on Users, then click on “Add User”.
2. Set a username for the new user and verify that the “User Enabled” option is set to ON, then click on the “Save” button.
3. In the user page, click on “Credentials” tab and set a password for the user. Please verify that the “Temporary” option is set to OFF (change it otherwise). Click on the “Set Password” button to assign credentials to the user.

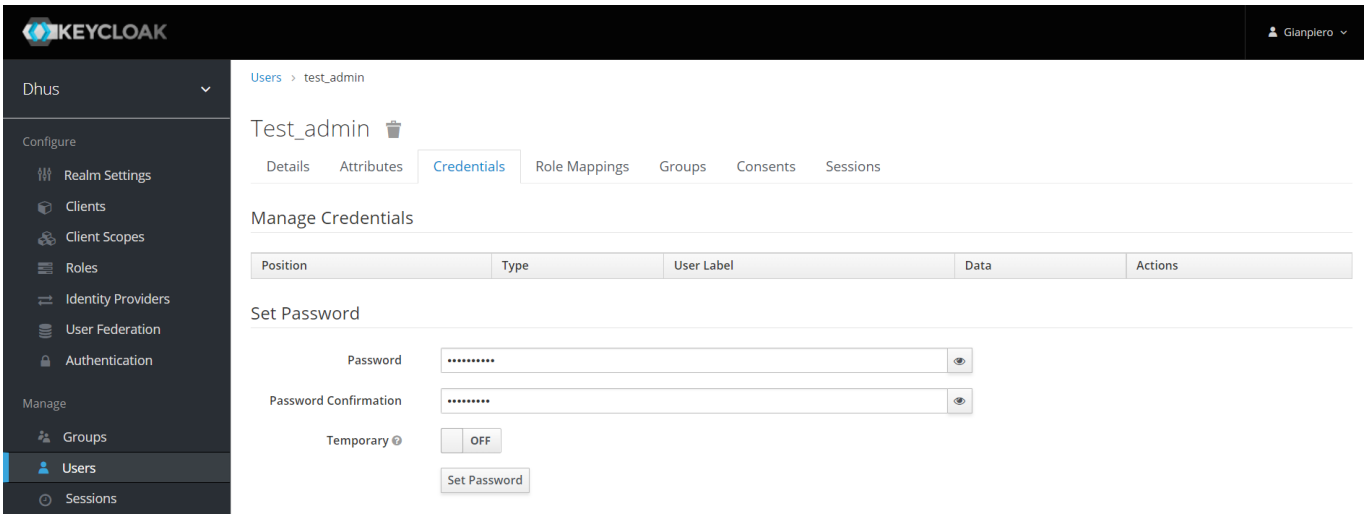

Figure 5 Set DAFNE user credentials

4. In the user page, click on “Role Mappings” tab and select “dafne” from the “Client roles” dropdown menu
5. As you can see, the default “DATAFLOW_VIEWER” role is assigned to the new user. If you want to add the “DATAFLOW_MANAGER” role, please select “DATAFLOW_MANAGER” from the “Available Roles” text area and click on “Add selected”.

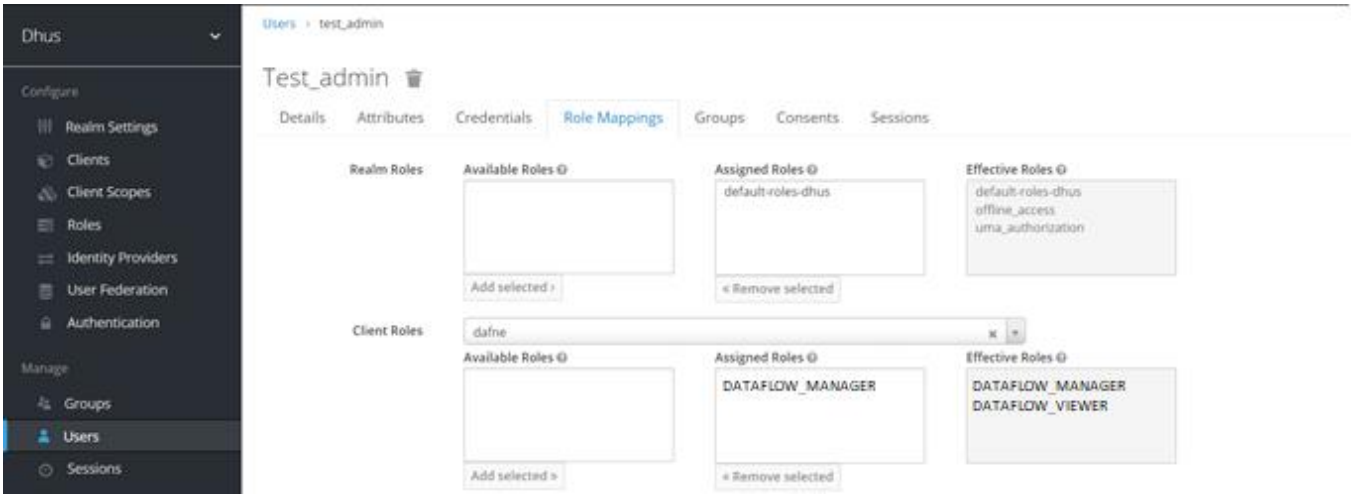

Figure 6 Assign role to DAFNE user

Manage password policy for DAFNE in Keycloak

Keycloak allows to define rules for passwords, clicking on the “Authentication” option in the menu on the left and selecting the “Password Policy” tab.

You can define rule, clicking on the “Add Policy” drop down menu and selecting the policies that feet your need.

An example of password policy is reported in the picture below. This policy is compliant with ESA specifications for password in Copernicus services:

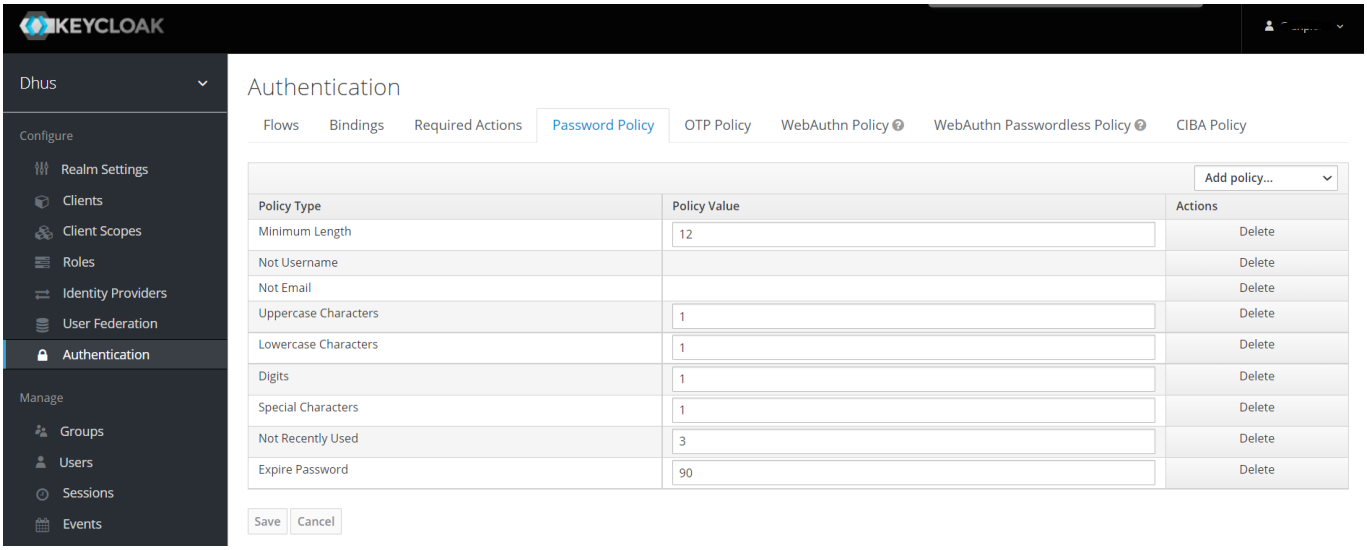

Figure 7 Add password policies for DAFNE users