Connecting Europe Facility 2014-2020

AQMO

Air Quality and MObility

Grant Agreement Number: INEA/CEF/ICT/A2017/1566962

2017-FR-IA-0176

D7.2

AQMO platform reference manual

Final

Version: 1.0

Author(s): François Bodin (UR1), Laurent Morin (UR1), Benjamin Depardon (UCit), Yiannis Georgiou (Ryax), Sylvie Perdriel (AmpliSIM)

Date: 01 / 12 / 2020

Project and Deliverable Information Sheet

AQMO Project	**Project Ref. №:** INEA/CEF/ICT/A2017/1566962	
	Project Title: Air Quality and mobility	
	Project Web Site: http://aqmo.irisa.fr/	
	Deliverable ID: <D7.2>	
	Dissemination Level: PU *	**Contractual Date of Delivery:** 31 / 12 / 2020
		Actual Date of Delivery: 08 / 01 / 2021
	EC Project Officer: Mark VELLA MUSKAT	
Authorship	**Written by:**	François Bodin (UR1)
	Contributors:	Benjamin Depardon (UCit), Yiannis Georgiou (Ryax), Laurent Morin (UR1), Sylvie Perdriel (AmpliSIM)
	Reviewed by:	Corentin Lefèvre (Neovia Innovation)
	Approved by:	Technical and Management Boards

* - The dissemination level are indicated as follows: **PU** – Public, **CO** – Confidential, only for members of the consortium (including the Commission Services) **CL** – Classified, as referred to in Commission Decision 2991/844/EC.

Document Status Sheet

Version	**Date**	**Status**	**Comments**
0.1	30/11/2020	Draft V1	
1.0	04/12/2020	Final version	

References and Applicable Documents

List all external documents referenced in this document

[1] RUDI http://rudi.datarennes.fr/
[2] AQMO deliverable D5.2 Report on the analysis of the visualization tools
[3] Soulhac, L., Salizzoni, P., Cierco, F.-X. et Perkins, R. J., 2011. The model SIRANE for atmospheric urban pollutant dispersion: PART I: presentation of the model. Atmospheric Environment. Volume 45, Issue 39, 7379-7395, (http://air.ec-lyon.fr/SIRANE/)
[4] IDRIS http://www.idris.fr/eng/info/missions-eng.html
[5] AWS https://aws.amazon.com/

List of Acronyms and Abbreviations

Below is an extensive the List of Acronyms used in previous deliverables. Please add additional ones specific to this deliverable and delete unrelated ones.

API	Application Programming Interface
AQMO	Air Quality and Mobility
AWS	Amazon Web Service
CCME	Cloud Cluster Made Easy
CPU	Central Processing Unit
GPS	Global Positioning System
HPC	High-Performance Computing
K3S	Kubernetes
NUC	Next Unit of Computing
PM	Particulate Matter
SDN	Software Defined Network
VPN	Virtual Private Network
WMS	Workflow Management System

Table of contents

List of Figures

Introduction

AQMO is proposing a platform that integrates mobile sensors with numerical simulations for air quality analysis. This deliverable gathers the set of documentations related to the AQMO platform components.

This manual is organized as follows: the first section recalls the various parts of the platform. The sections afterwards provide pointers to the documentation related to the components.

Co-financed by the Connecting Europe Facility of the European Union

1. Platform components overview

Figure 1 gives an overview of the technical components of the platform.

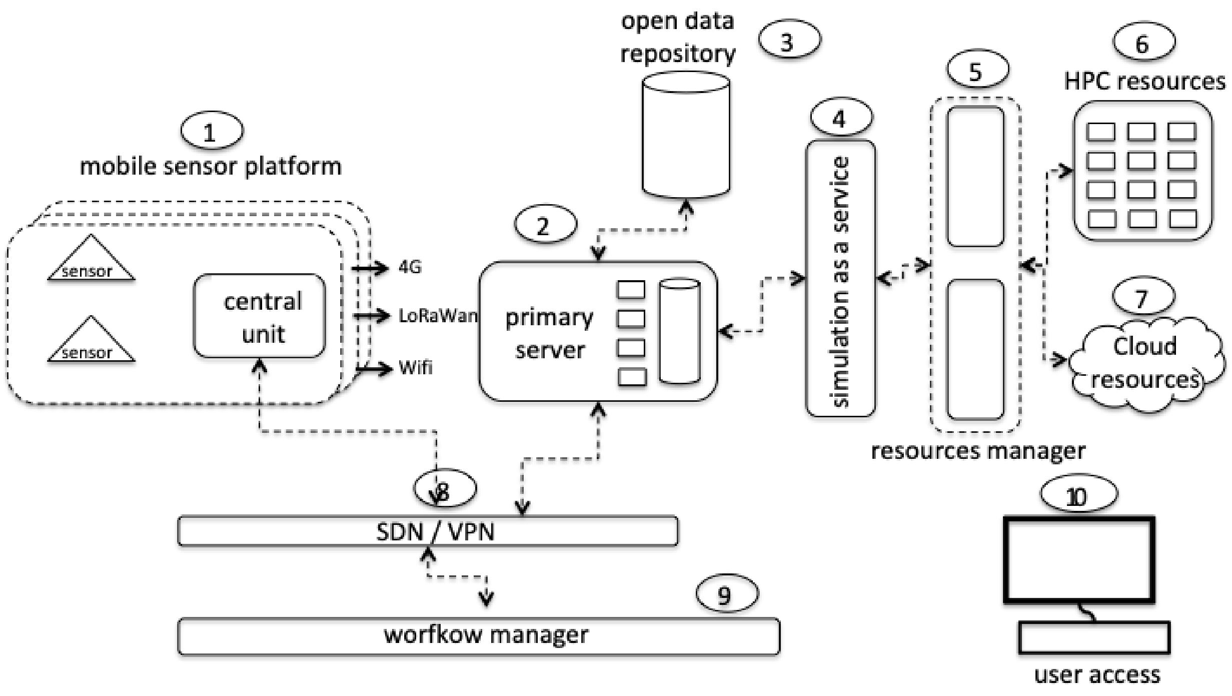

Figure 1: AQMO platform overview

The components in Figure 1 are the following:

1) Mobile sensor platform: this encompasses a set of sensors that are connected wirelessly with a central unit that is in charge of storing, processing and communicating the collected data.
2) Primary server: this component is in charge of storing the data before their long-term storage. It also provides support for that data visualization (see [2]) and platform monitoring capabilities. This part is related to the "Fog" topic.
3) Open data repository: this part is related to the Rennes Métropolis Open Data initiative currently developed by the RUDI project [1].
4) Simulation as a service: This component provides SIRANE [3] simulation access as a service. SIRANE is the numerical model for pollutant dispersion.
5) Resources management: The element of the platform aims at dealing with computing resources.
6) HPC resources: This is the resource provided by the IDRIS supercomputing center [4].
7) Cloud resources: In AQMO we also use HPC clusters deployed on Amazon Web Services [5] (AWS) through UCit's software CCME (Cloud Cluster Made Easy)
8) SDN/VPN: Many devices in AQMO are enrolled in a Software Define Network that provide the interconnection with encryption capabilities between the devices and the primary server.

Co-financed by the Connecting Europe Facility of the European Union

9) Workflow manager: This software platform is in charge of orchestrating the different automations and data analytics workflows (pipelines) taking place at the edge (buses), fog (primary server) or HPC/Cloud resources.

Before going into more details in the next sections, it shall be noted that:

- Component 2 (“primary server”) is a proxy machine and as such does not have a reference manual
- Component 3 (“open data repository“) corresponds to the RUDI platform [1] and being under development (availability mid 2021) it is not yet documented
- Component 10 (“user access”) is explained in AQMO deliverable D7.3 “AQMO user manual”

2. Component 1: Central unit

The central unit (Numbered 1 in Figure 1) is in charge of collecting and uploading the data from the sensors (Particle Matter and GPS). This unit is organized around an Intel NUC running virtual LXC container. The central unit downloads the PM data using the LoRaWan or the 4G connections. For shorter latency purposes, the GPS data are downloaded using the 4G connection. The central unit is enrolled in a software defined network and a VPN is established with the “Primary Server”. Figure 2 gives an overview of the main elements in these components.

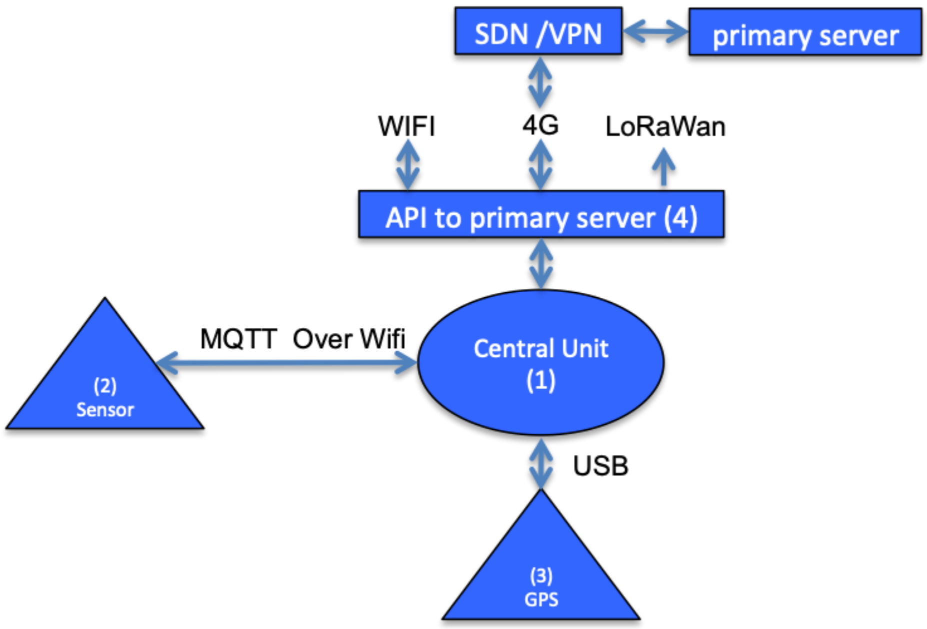

Figure 2: Overview of the central unit component

References

(1) AQMO D2.1 Report on the preliminary sensor platform
(2) http://www.alphasense.com/index.php/products/optical-particle-counter/
(3) https://dustinweb.azureedge.net/media/148300/bu-353s4.pdf

Co-financed by the Connecting Europe Facility of the European Union

3. Components 4 to 7: HPC as a Service

The HPC as a service solution is built upon 3 main elements (Numbered 4-5-6-7 in Figure 1):

1. The Primary Server that holds the core functionalities receiving the requests for HPC Computations ((2) authenticate user and execute requested workflow), select the relevant target computing platform ((3) Predict-IT)
2. An HPC Centre, accessed through web services APIs ((5) and (6))
3. A Cloud Platform in which HPC Clusters can be dynamically deployed (4) and then accessed through the same web services APIs ((5) and (6))

Figure 3 gives an overview of the main elements in these components.

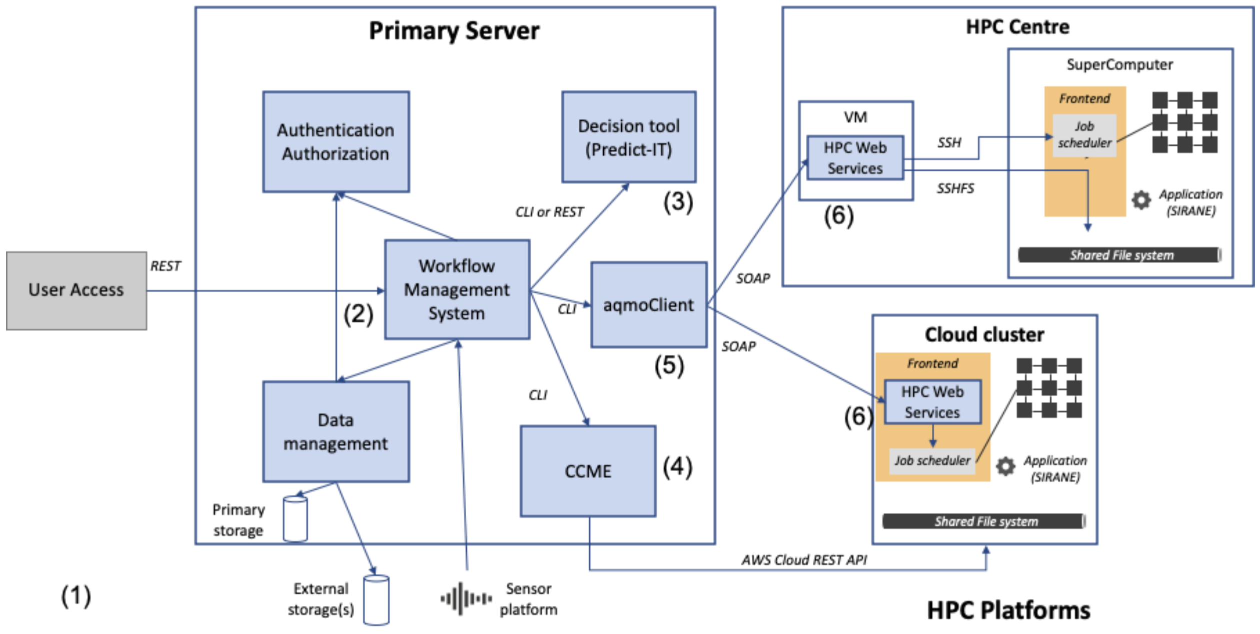

Figure 3: HPC as a Service components

References

(1) AQMO D4.1 - Architecture solution for HPC as a service
(2) Ryax documentation https://docs.ryax.tech/
(3) Predict-IT documentation: https://ucit.fr/product-publish/predictit/doc/v1.5/predictit_v1.5.pdf
(4) Annex A - CCME CLI documentation
(5) Annex B - aqmoClient documentation
(6) EnginFrame documentation: https://download.enginframe.com/

4. Components 8-9: SDN & Workflow Manager

The Software Defined Networking (SDN) provides security, enabling access to devices either on the edge or between the edge and the Primary Server infrastructure. The Workflow Management System (WMS) implements data analytics automations such as pollution contextualization which take place within the bus (edge side) or HPC job dispatching through the HPC as a Service functionalities (fog side). WMS runs on top of a Kubernetes installation. Due to edge devices running on the bus having low CPU and memory resources, Ryax Technologies makes use of the lightweight Kubernetes distribution K3S which can run on both the edge and the primary server ensuring low memory and CPU footprint.

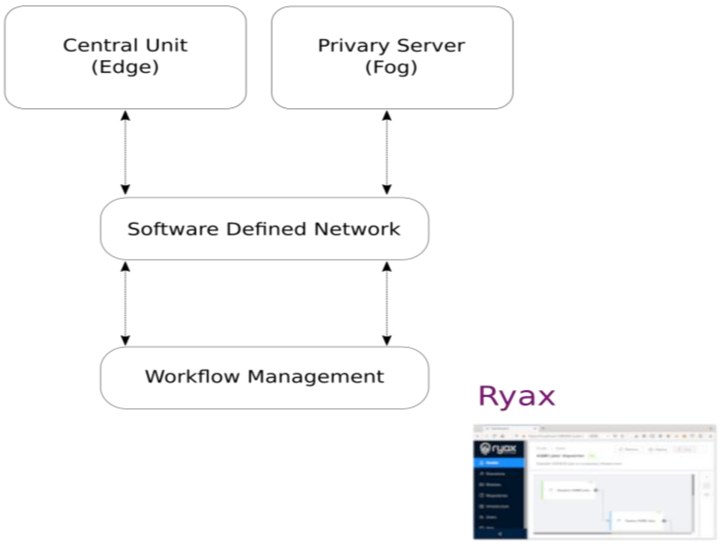

Figure 4: SDN & Workflow Management

References

(1) AQMO D5.1 - Workflow Management System

(2) Ryax documentation https://docs.ryax.tech/

Co-financed by the Connecting Europe Facility of the European Union

Annex A - CCME CLI documentation

```
usage: ccme [-h]
            {create,update,delete,start,stop,status,list,instances,ssh,createami,version,add-ttl,delete-ttl,list-ttl,create-profile,list-profiles,delete-profile}
            ...

ccme is the CCME CLI and permits launching and management
of HPC clusters in the AWS cloud.

positional arguments:
  {create,update,delete,start,stop,status,list,instances,ssh,createami,version,add-ttl,delete-ttl,list-ttl,create-profile,list-profiles,delete-profile}
    create              Creates a new cluster.
    update              Updates a running cluster using the values in the
                        config file.
    delete              Deletes a cluster.
    start               Starts the compute fleet for a cluster that has been
                        stopped.
    stop                Stops the compute fleet, leaving the master server
                        running.
    status              Pulls the current status of the cluster.
    list                Displays a list of stacks associated with CCME.
    instances           Displays a list of all instances in a cluster.
    ssh                 Connects to the master instance using SSH.
    createami           (Linux/macOS) Creates a custom AMI to use with CCME.
    version             Displays the version of CCME.
    add-ttl             Creates a TTL and associates it with a cluster.
    delete-ttl          Deletes a TTL associated with a cluster.
    list-ttl            Displays a list of TTLs associated with CCME clusters.
    create-profile      Create a new cluster profile (wizard or advanced
                        mode).
    list-profiles       Displays the list of configured Profiles configured
                        for CCME clusters.
    delete-profile      Deletes a specified CCME Profile.

optional arguments:
  -h, --help            show this help message and exit

For command specific flags, please run: "ccme [command] --help"
```

Co-financed by the Connecting Europe Facility of the European Union

Annex B - aqmoClient documentation

Available commands

```
Usage: ./aqmoClient.sh [-hV] -e https://enginframe_host:port/enginframe -u "username"
[COMMAND]
Parameters:
  -e url_enginframe                                     EnginFrame server URL
  -h, --help                                            Show this help message and exit.
  -u  user_certificate                                  Alias of the certificate in the keystore
  -V, --version                                         Print version information and exit.
  -d, --debug                                           (Hidden option) Print full stack trace if errors

Commands:
  files    (Interact with files)
      upload <vroot:/dir> <inputFiles>...                      Upload files inside a vroot
      download <vrootID> <inputFilesFullPath>... [-o=<OutputDir>] Download an archive of remote
files
       vroot-list                                       List all home vroot and names
       list <vrootID> [dir]                             List all files in a vroot or in a specific
directory if specified (heavy directory tree won't be showed)

  services  (Interact with services)
    list                                                     List Available services
    describe <ServiceID>                                     List service parameters
    execute [-p="key=value,files=file1:file2,..."] <ServiceID>   Execute the service

  spoolers  (Interact with spoolers)
    list                                                          List availables spoolers
    describe [-j="jobID1,jobID2,..."|default:all jobs] <SpoolerID>  List jobs statuses inside a spooler
    files
       list <SpoolerID>                                 List file tree inside a specific spooler
       download <SpoolerID> <inputFilesFullPath>... [-o=<OutputDir>]           Download files from
a specific spooler
```

JSON Outputs by command

Interacting with the user personal directory

List user's vroots

Command: files vroot-list

```
Example: ./aqmoClient.sh -e url_enginframe -u username files vroot-list
Output:
{
  "vroots": [
          {
          "vroot-label": "efadmin's Home",
          "vroot-id": "4501b5ed55dfdc826fa6db4acd09b24b4f552fc0"
```

```
        },
        {
        "vroot-label": "File System",
        "vroot-id": "2bd42ea296f239539a1138d95618e67dd0bd1f66"
        }
  ]
}
```

List files tree inside a vroot (or content in a specified directory)

Command: files list <vrootID> [dir]

```
Example: ./aqmoClient.sh -e url_enginframe -u username files list
4501b5ed55dfdc826fa6db4acd09b24b4f552fc0
Output:
{
  "vroot-id": "4501b5ed55dfdc826fa6db4acd09b24b4f552fc0",
  "directory-tree": [
        "test/test_subfld/emptydir/",
        "test/test_subfld/test_subfold2/10_10_01_2014_11_10_59_florent.jpg",
        "test/test_subfld/test_subfold2/client.conf",
        "test/client.conf",
        "client.conf"
  ]
}
```

```
Example: ./aqmoClient.sh -e url_enginframe -u username files list
4501b5ed55dfdc826fa6db4acd09b24b4f552fc0 test
Output:
{
  "vroot-id": "4501b5ed55dfdc826fa6db4acd09b24b4f552fc0",
  "directory-path": "test",
  "directory-content": [
        "test_subfld/",
        "somedata.dat"
  ]
}
```

Download remote files

Command: files download <vrootID> <inputFilesFullPath>... [-o=<OutputDir>]

Note: files are archived (.zip) and downloaded locally

```
Example: ./aqmoClient.sh -e url_enginframe -u username files download
4501b5ed55dfdc826fa6db4acd09b24b4f552fc0 test/somedata.dat test/test_subfld/ -o
localDir/localSubdir
```

```
Output:
{
  "downloaded": [
          {
          "file": "test/somedata.dat test/test_subfld/ ",
          "local-path": "localDir/localSubdir/EF.Download.2020-04-03_12-45-46.zip"
          }
  ]
}
```

Upload files

Command: files upload <vroot:/dir> <inputFiles>... Note: to upload at the directory root, use the following: files upload vroot:/ <inputFiles>...

```
Example: ./aqmoClient.sh -e url_enginframe -u username files upload
4501b5ed55dfdc826fa6db4acd09b24b4f552fc0:/test localfile1.dat dir/localfile2.dat
Output:
{
  "upload-status":"OK"
}
```

Interacting with the available services for the logged user

Listing availables (published for the user) services

Command: services list

```
Example: ./aqmoClient.sh -e url_enginframe -u username services list
Output:
{
  "services": [
          {
          "name": "Stress test: 1h 2CPU",
          "id": "//applications/batch_5a5dcd1bd9604106a3350882dc56faa4.published"
          },
          {
          "name": "Sample Compress Job",
          "id": "//applications/batch_builtin_sample_compress_job.published"
          },
          {
          "name": "Linux Desktop",
          "id": "//applications/interactive_builtin_linux_desktop.published"
          }
  ]
}
```

Listing service parameters

Command: services describe <ServiceID>

```
Example: ./aqmoClient.sh -e url_enginframe -u username services describe
//applications/batch_builtin_sample_compress_job.published
Output:
{
  "service-name": "Sample Compress Job",
  "service-id": "//applications/batch_builtin_sample_compress_job.published",
  "services-parameters": [
          {
          "param-name": "Execution cluster:",
          "param-key": "cluster",
          "param-extra-desc": "",
          "param-type": "list",
          "list-choices": [
           {
          "name": "linux",
          "value": "linux:slurm"
          }
          ]
          },
          {
          "param-name": "File to compress:",
          "param-key": "file",
          "param-extra-desc": "(this file will be uploaded to the server)",
          "param-type": "Single file upload"
          },
          {
          "param-name": "Compression level:",
          "param-key": "level",
          "param-extra-desc": "",
          "param-type": "list",
          "list-choices": [
              {
          "name": "maximum",
          "value": "9"
          },
          {
          "name": "medium",
          "value": "4"
          },
          {
          "name": "minimum",
          "value": "1"
          }
          ]
          },
          {
          "param-name": "Multiple File Upload",
            "param-key": "files",
            "param-extra-desc": "",
            "param-type": "Multiple file upload"
            }
  ]
}
```

Command: execute [-p="key=value,files=file1:file2,..."] <ServiceID>

```
Example: ./aqmoClient.sh -e url_enginframe -u username services execute -
p="cluster=linux:slurm,file=hugeLocalDataFile.dat,level=9"
//applications/batch_builtin_sample_compress_job.published
Output:
{
   "spooler-name":"Job Compress hugeLocalDataFile.dat",
   "spooler-id":"spooler:///opt/nice/enginframe/spoolers/efadmin/tmp7380574954436226192.ef"
}
```

Interacting with the spoolers owned by or shared to the user

List all available spoolers

Command: spoolers list

```
Example: ./aqmoClient.sh -e url_enginframe -u username spoolers list
Output:
{
  "spoolers": [
            {
            "name": "Job Compress hugeLocalDataFile.dat",
            "id": "spooler:///opt/nice/enginframe/spoolers/efadmin/tmp7380574954436226192.ef"
            },
            {
            "name": "Job Compute atomic destruction",
            "id": "spooler:///opt/nice/enginframe/spoolers/efadmin/tmp6070959080153131875.ef"
            }
  ]
}
```

Get a spooler status

Command: spoolers describe [-j="jobID1,jobID2,..."|default:all jobs] <SpoolerID>

Note: The -j option is here to only list the status of specific jobs. All jobs are displayed by default.

```
Example: ./aqmoClient.sh -e url_enginframe -u username spoolers describe
spooler:///opt/nice/enginframe/spoolers/efadmin/tmp7380574954436226192.ef
Output:
```

```
{
  "spooler-name": "Job Compress hugeLocalDataFile.dat",
  "spooler-id": "spooler:///opt/nice/enginframe/spoolers/efadmin/tmp7380574954436226192.ef",
  "spooler-expiration-timestamp-UTC+0": 1586517032,
  "spooler-global-status": "DONE",
  "spooler-jobs": [
          {
          "jobID": 160,
          "name": "Job_Compress_client.conf",
          "status": "DONE"
          }
  ]
}
```

List files inside a spooler

Command: spoolers files list <SpoolerID>

Note: job files output may vary.

```
Example: ./aqmoClient.sh -e url_enginframe -u username spoolers files list
spooler:///opt/nice/enginframe/spoolers/efadmin/tmp7380574954436226192.ef
Output:
{
  "elements": [
          "hugeLocalDataFile.dat.gz",
          "slurm-160.out"
  ]
}
```

Command: spoolers files download <SpoolerID> <inputFilesFullPath>... [-o=<OutputDir>]

```
Example: ./aqmoClient.sh -e url_enginframe -u username spoolers files download
spooler:///opt/nice/enginframe/spoolers/efadmin/tmp7380574954436226192.ef
hugeLocalDataFile.dat.gz slurm-160.out -o somelocaldir/subfold
Output:
{
  "downloaded": [
          {
          "file": "hugeLocalDataFile.dat.gz slurm-160.out ",
          "local-path": "somelocaldir/subfold/EF.Download.2020-04-03_13-59-27.zip"
          }
  ]
}
```

Co-financed by the Connecting Europe Facility of the European Union