Norsk institutt for luftforskning
Norwegian Institute for Air Research

Final report

Support in development and implementation of Air Quality e-Reporting in West Balkan

Rune Å. Ødegård, Michael J. Kobernus

NILU report 5/2017

NILU report 5/2017	ISBN: 978-82-425-2872-8 ISSN: 2464-3327	CLASSIFICATION: A – Unclassified (open report)
DATE 10.02.2017	SIGNATURE OF RESPONSIBLE PERSON Kari Nygaard, Managing Director (sign.)	NUMBER OF PAGES 23
TITLE Final report Support in development and implementation of Air Quality e-Reporting in West Balkan		PROJECT LEADER Rune Å. Ødegård NILU PROJECT NO. O-116081
AUTHOR(S) Rune Å. Ødegård, Michael J. Kobernus		QUALITY CONTROLLER Kjersti K. Tørnkvist
REPORT PREPARED FOR European Environment Agency (EEA)		CONTRACT REF. 3437/B-2016/IPA-2015/EEA.56560

ABSTRACT

The Norwegian Institute for Air Research (NILU), 4sfera Innova SLU (4sfera) and EKONERG have been commissioned to develop a software solution for Air Quality e-Reporting for the West Balkan countries

The software solution has been successfully installed and training has been provided. For the reporting of air quality according to the air quality directives 2004/107/EC and 2008/50/EC, Albania, Montenegro and Serbia has used the software for reporting 2015 data to EEA.

NORWEGIAN TITLE

KEYWORDS

Air Quality | West Balkan | e-Reporting

ABSTRACT (in Norwegian)

PUBLICATION TYPE: Digital document (pdf) | COVER PICTURE: Source: NILU

Preface

The Norwegian Institute for Air Research (NILU), 4sfera Innova SLU (4sfera) and EKONERG were commissioned to develop a technical solution for Air Quality E-Reporting for the West Balkan countries.

The Balkans is a peninsula and cultural area in South-East Europe. It derives its name from the Balkan Mountains. These stretch from the eastern border of Serbia to the Black Sea. For the purposes of this report, the Balkans comprises the following countries. Serbia, Albania, Bosnia and Herzegovina, FYROM, Montenegro and Kosovo.

This document describes the delivered solution, the time scale and the responsibilities for all the contributors and a summary of the final implementation for each state. This includes the development, installation and staff training for the completed solution.

Contents

Summary

This document comprises the final report detailing the Air Quality e-Reporting project for the West Balkan countries. It details the project plan, development, on site implementation, testing and staff training of the delivered solution for the project, the e-Reporting application called 'Raven.'

The following is an overview of the major milestones achieved during the project:

- **First version of the IT-software package named RAVEN created**
- **Establishment of the open source repository at** https://git.nilu.no/eea-tools/raven
- **Workshop in Zagreb 20-21 September**
- **Templates created**
- **Individual "go through"/guides in use of the templates**
- **Technical documentation of RAVEN**
- **Bug fix updates and new deployment of RAVEN**
- **Upload of dataflow D and E1a to CDR (Central Data Repository)**

Final report

1 Core objectives

The core objective of this project was to provide assistance to the West Balkan countries (Albania, Bosnia and Hercegovina, FYROM, Kosovo, Montenegro and Serbia) in order to implement an e-Reporting solution for:

Data flows D (assessment methods)
E1a (primary validated data) and
E2a (primary Up-To-Date data)

This was achieved by:

1. Assisting in developing local expertise and technical knowledge to establish and implement an AQ e-Reporting system.

2. Assisting in the development of a national system for AQ e-Reporting for data flows D, E1 and E2a, including all data files and mechanisms for data transfer as defined in the air quality implementing provisions (IPR) (2011/850/EU).

3. Developing a technical solution and training of local expertise within the national institutions that will ensure the management and operation of an e-Reporting system.

These objectives were all realised. This document shall outline the achievements of the project. A full technical overview is provided in the Appendix.

2 Development

The main development outcome of the project was the software application, RAVEN. This is a locally installed application that enables the import of comma separated value files (CSV), via an http application programming interface (API). Based on these imports, a user can download the content for Dataflow D and E1a in a correctly configured format.
This data can then be uploaded into RAVEN.

The RAVEN application allows the Member State to upload both E2a and E1a data to the EEA. However, there is an additional feature which enables the EEA to fetch E2a data directly (similar as a Sensor Observation Service) from the Members States RAVEN application. This can be achieved only if the Member States update their data regularly and that the application does not reside behind a firewall.

2.1 RAVEN

RAVEN is the name given to the complete technical solution. The name RAVEN comes from a popular series of novels, and now Television show, where ravens are used as messengers. Since the show is filmed largely in Croatia, and the software application carries messages (XML files) the name RAVEN was chosen as being both appropriate and fit for purpose. Simply put, RAVEN is a data transfer, or messenger service.

The overall concept is outlined in figure 1. The main e-Reporting component was developed by NILU. However, additional development work was also carried out by 4sfera, with regards to the templates used to define the input data. Further development work was also performed by EKONERG, whose operator created many scripts in order to enable the export of data from various data sources for many of the participating member states.

Once a working prototype was complete, installation was carried out at each of the participating member state's local servers by engineers from EKONERG, where testing and training also were performed.

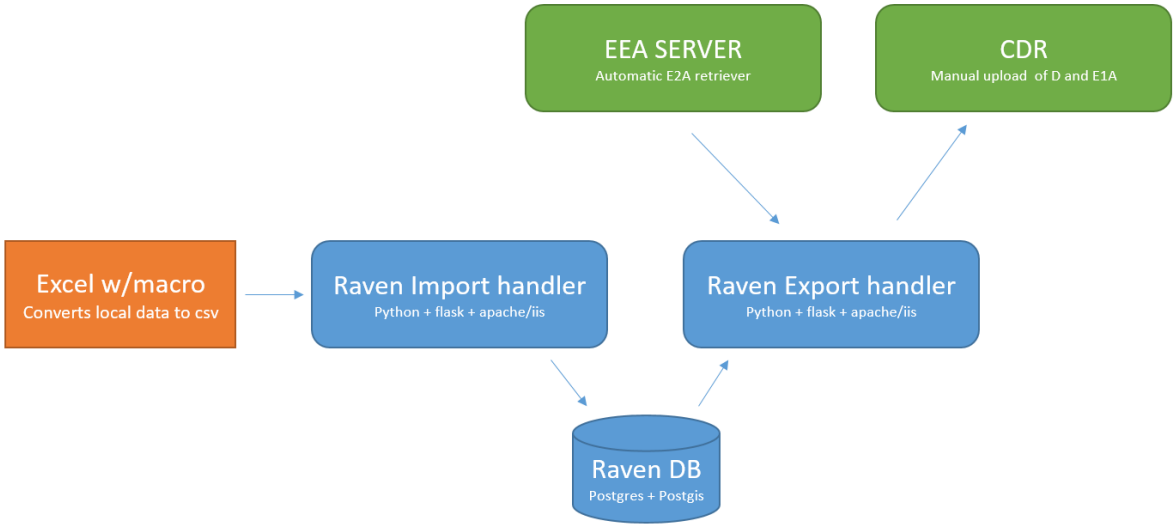


Figure 1: Business workflow of the AQ e-Reporting system RAVEN

The database was designed as a relational model. The database model is shown in the figure below.

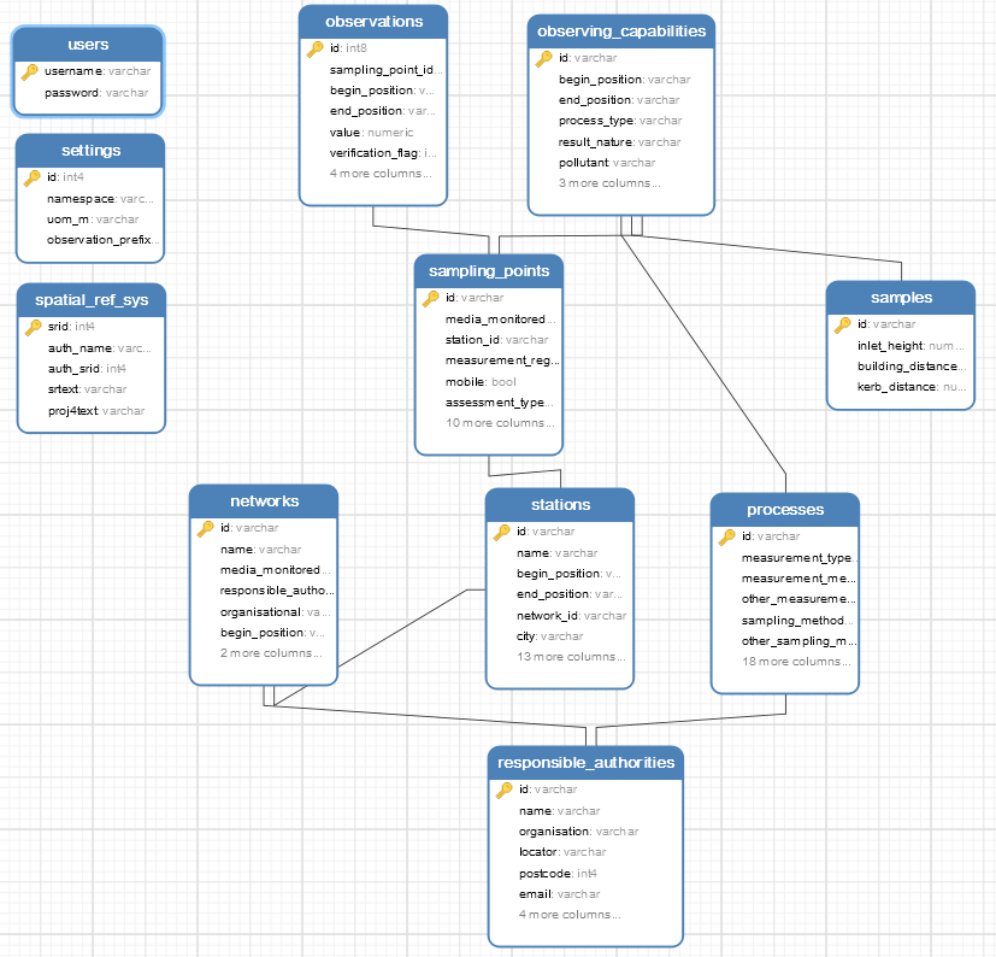


Figure 2: Database model for RAVEN

The full working prototype of RAVEN can be found and downloaded under the open source repository GitLab: https://git.nilu.no/eea-tools/raven

RAVEN Software requirements and installation documentation can be found under:

https://git.nilu.no/eea-tools/raven/tree/master

3 Workshop

A two days' workshop was arranged and held in Zagreb. The main focus of the workshop was agreeing the architecture, and the development and testing of the IT software package, RAVEN. The WB countries were invited on day two, but only representatives of two Member States, Serbia and Montenegro, were present. The following program was carried-out on day two:

- Introduction to dataflow D – importing data via templates.
- Open discussion on the current status of the database in Serbia and how and when to implement the solution.

- Open discussion on current status of the database in Montenegro and how and when to implement the solution.

With agreements made on installation dates, the next stage was deployment.

4 Deployment

Installation of the Raven solution was carried out at each of the participating Balkan countries at the following times:

Country	Date	Status
Serbia	18th/19th October	Installation successful
Montenegro	8th/9th November	Installation successful
Kosovo	17th/18th November	Installation successful
Albania	1st/2nd December	Installation successful
Bosnia and Herzegovina Sarajevo	21st/22nd November	Installation successful
Bosnia and Herzegovina – Banja Luka	13th/14th December	Installation successful

4.1 FYROM

FYROM decided to proceed with their own system developed in the EU funded Twinning project "Further strengthening the capacities for effective implementation of the acquis in the field of air quality". With support from experts from the Finnish Meteorological Institute the project has implemented solution for e-reporting. Due to this FYROM does not require installation the IT-software package RAVEN.

4.2 Serbia

EKONERG consultants visited Serbia and successfully installed the IT-software package RAVEN in Belgrade on 18th and 19th of October. This was done with cooperation of employees from SEPA (the Serbian Environmental Protection Agency). In addition to the installation, a meeting was held in SEPA with a third party who was responsible for developing an Air Quality System for SEPA. A conclusion from the meeting was that SEPA plans to send data periodically to RAVEN's database directly. Username and passwords were provided with read access to the database as well as additional rights to access the measurement tables.

SEPA was introduced to the functionality of the RAVEN application and the database structure. As part of the training, SEPA uploaded CSVs and generated the required XML for the air quality e-Reporting.

4.3 Montenegro

RAVEN was successfully installed and demonstrated at the EPA (Environmental Protection Agency). Training was provided to EPA´s IT.

A visit was made to Podgorica on 7th of November and meetings took place on 8th and 9th of November 2016, in order to install RAVEN. RAVEN was successfully installed on two virtual machines, one instance on a test environment and one on the production environment. There

were no issues with the installation of the RAVEN. Access to data for the two virtual machines was provided by personnel in Teched, the company that implements the Air Quality System in Montenegro.

4.4 Albania

RAVEN was successfully installed at the Agency of Environment and Forestry. The application was successfully demonstrated and training was provided.

Visit was made to Tirana to the Agency of Environment and Forestry on 30th of November and meetings took place on 1st and 2nd of December 2016 for installation of RAVEN. RAVEN was successfully installed on a Linux based virtual machine. The virtual machine is dedicated only for RAVEN.

4.5 Bosnia and Herzegovina

As a result of FYROM implementing their own data transfer system, and thus not requiring RAVEN, sufficient funds were released that enabled the team to install RAVEN at two locations within BiH. This is a key requirement for BiH, as the country requires separate reporting from both of its political entities. With the approval of the EEA, the funds released from FYROM were used to the install RAVEN at both Sarajevo *and* Banja Luka.

4.5.1 Sarajevo

RAVEN was successfully installed at the Federal Hydrometeorogical Institute on their server. The application was successfully demonstrated and training was provided.

EKONERG visited Sarajevo at the Federal Hydrometeorogical Institute on 20th of November and installation and training took place on 21st and 22nd of November 2016. The application was successfully installed on a Windows 7 environment.

Before providing up-to-date data, the institute needs approval from the Federal Government, which was not approved at the installation time.

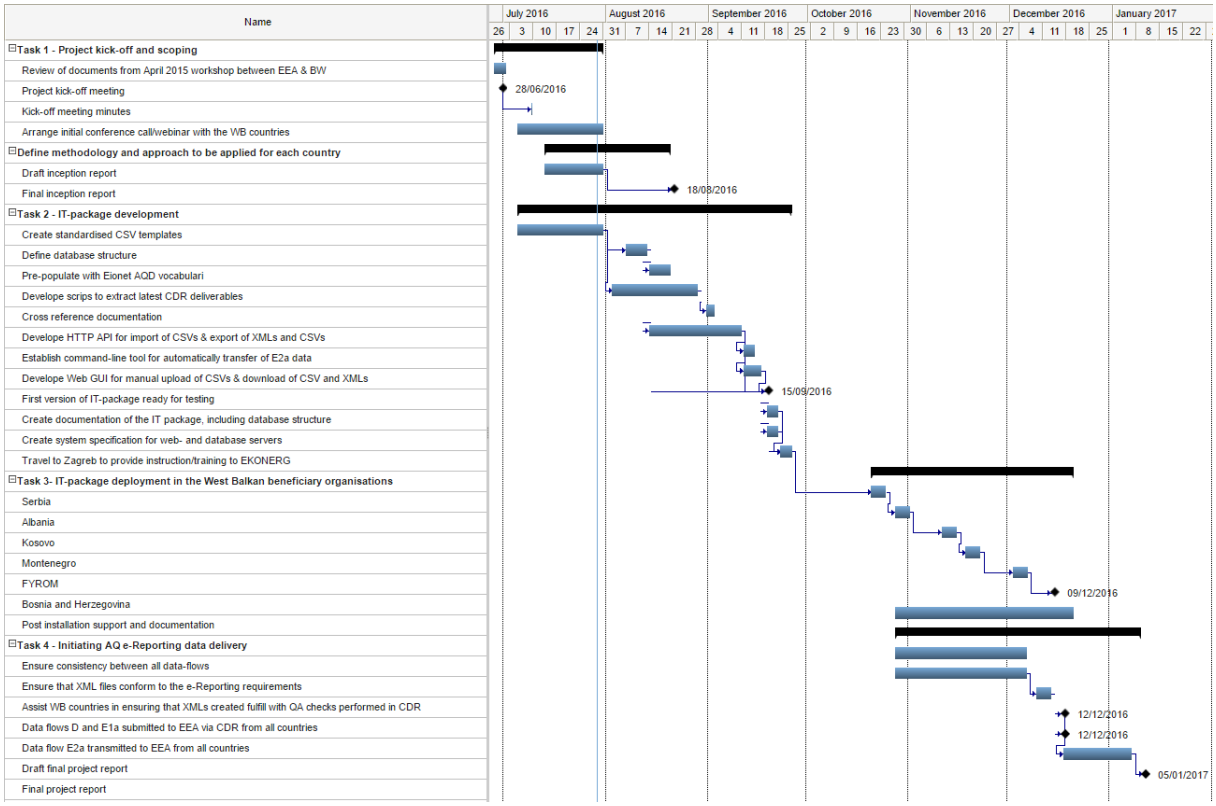
4.5.2 Banja Luka

RAVEN was successfully installed and demonstrated at the Republic Hydrometeorogical Institute, and training was provided.

A visit was made to Banja Luka to Republic Hydrometeorogical Institute on 12th of December and meetings took place on 13th and 14th of December 2016 for installation of RAVEN. RAVEN was successfully installed on Windows 7 environment.

5 Project implementation

The timeline of the project, detailed in the Inception Report, showed the various main tasks to accomplish the project goals.
The timeframe was respected, and milestones were achieved on time.



5.1 Task 1

Task 1 was a largely management and administrative task, comprising meetings, teleconferences concept development and specification development. The outcome of task 1 was the Inception Report, which was delivered in 2016. This outlined the technical concept and deliverables for the project.

5.2 Task 2

Task 2 was the development of the IT solution and included several subtasks. The expected deliverables for Task 2 are largely software based and include scripts, software and specifications. Output scripts were installed locally for each Member State.

RAVEN Software requirements and installation documentation can be found under:

https://git.nilu.no/eea-tools/raven/tree/master

- Creation of Standardised Templates
- Definition of Database Structure
- Prepopulate with Eionet AQD Vocabulary
- Develop scripts to extract latest CDR deliverables where available
- Cross reference documentation

- Developing HTTP API for import of CSVs
- Establish command line tool for transferring E2a Data
- Developing business logic for creation of XMLs according to the e-reporting schema
- Developing HTTP API for export of XMLs and CSVs
- Create IT solution documentation, including database schema
- Create system specifications web and database servers
- Travel to Zagreb

5.3 Task 3

Task 3 was deployment of the IT solution to the West Balkan beneficiary organisations. This work was completed, and all the participating members successfully installed the Raven IT solution on their servers. See the section on Deployment, for a summary overview of the installation progress.
The full working prototype of RAVEN can be found and downloaded under the open source repository GitLab: https://git.nilu.no/eea-tools/raven

RAVEN Software requirements and installation documentation can be found under:

https://git.nilu.no/eea-tools/raven/tree/master

5.4 Task 4

Task 4 was 4 concerned with initiating Quality Assurance for the AQ e-Reporting data delivery systems.

- Ensure consistency between all data flows
- Validate XML files
- Assist Member States by checking if XML files pass CDR checks
- Ensure Dataflows D and E1a are submitted to EEA via CDR
- Ensure Dataflow E2a is submitted
- Produce Final Project Report

With the exception of Serbia, which is ready to start transferring UTD data, the other Balkan states are not in a position to transfer E2a data as they lack proper data acquisition systems. However, once that situation is resolved locally, E2a data transfer will be possible.
Validation of Task 4 subtasks occurred as the project team manually reviewed output files, including CSV files, XMLS files as well as submitted data files within the CDR. Tests were performed to ensure accuracy, and corrections made where errors were identified. Refer to the section on Software Updates for information on corrections.

6 Software Updates

After installation further updates were made on a regular basis in response to issues raised by the member states, and due to new quality control test on CDR. The table below, list up the updates:

Date	Bug fix
17 Jan, 2017	Mismatch between fromtime for observations and fromtime in the defined range. Minimum totime was used for range but minimum fromtime should be used. There was no sorting of the observations. They appeared at random order
05 Jan, 2017	Added CSV response for imports
29 Dec, 2016	Changed syntax from Equipment to equipment according to xml schema
23 Dec, 2016	Changed time from 24:00:00 to 23:59:59 Renamed syntax equipment to otherEquipment Rearranged order of child elements in RelevantEmissions
22 Dec, 2016	Changed syntax from samplingEquipment to SamplingEquipment

7 Conclusion and Summary

The project was challenging due to the very short duration and tight schedule. Furthermore, as a distributed development project where there are multiple actors contributing various aspects of the technical solution, experience dictates that this generally makes management, coordination and collaboration more difficult. However, thanks to regular communication and a shared vision, the project team members worked efficiently together. There were no major issues or roadblocks, and the proposed solution was installed at the participating institutes' local servers at the agreed times and largely without issue.

All the West Balkan countries have now used RAVEN , either for initial testing of the process or for generating XMLs for dataflow D and E1a. Most have successfully uploaded the report data to the CDR.

Appendix A

Technical Overview

Requirements
The following is required to install and run the application:

- Postgres 9.5+
- Postgis extension
- Python 2.7
- WSGI complient server
- See requirements.txt for external libraries used

Installation
The following content describes the necessary components and steps to install Raven locally.

Database
First create a databaseuser:

createuser -sP ravendb

Enter a password when asked for one. The user needs to be a superuser to be able to create the postgis extension.

Then create the empty schema for the database:

createdb --owner=ravendb ravendb

Test if it's possible to connect to the newly created database:

psql -U ravendb -h 127.0.0.1 ravendb

\q to log out of the psql shell.

The database server needs to have Postgis installed.

apt-get install postgresql-9.5-postgis-2.2

This brings a lot of dependencies.

Run the following to import the schema.

psql -U ravendb -d ravendb -h 127.0.0.1 -a -f schema.sql

There might be some more configuration on the Postgresql server after this, dependent on whether it is a new server or an older server. If there is problems getting connections from the application server, have a look at /etc/postgresql/9.5/main/pg_hba.conf and /etc/postgresql/9.5/main/postgresql.conf. pg_hba.conf to allow certain users to access from certain servers, and in postgresql.conf to allow remote connections in the first place.

Linux web server
On the webserver we have to start with a configuration for apache2. Write a custom configuration file, /etc/apache2/sites-available/raven.conf.

The ServerName settings in raven.conf has to be registered as a CNAME, pointing to the A-record for the webserver, in the DNS-servers.

Make ssl-certs for https://someurl.com, and put the key and crt file into /etc/apache2/ssl.

Make a DocumentRoot directory under /var/www/html, call it raven. Give it sufficient permissions for apache2 to be able to read it.

Copy all of the files in the web folder to there.

Make a raven.wsgi file, to start the application. Here we source the virtual environment and starts the application.

Next is to make a virtual environment for python, and then installing all the modules from requirement.txt from that. psycopg2 is dependent on libpq-dev and python-dev.

virtualenv --system-site-packages .virtualenv cd .virtualenv/ source bin/activate cp //requirements.txt . pip install -r requirements.txt

Change the file database/db.py to connect to the correct database.

Windows web server

If Postgres and postgis is not installed, please do so. Download from here:https://www.postresql.org and http://postgis.net/

Please follow this article to install the application on IIS: http://netdot.co/2015/03/09/flask-on-iis/

Make sure the Python27 folder has IIS_IUSRS rights.
Install all dependencies from requirements.txt globally or with virtualenv.
To do this globally run the following line in the python script folder:

```
pip install -r requirements.txt
```

It is highly recommended that you set up your server with **https**
Also: **Change the file database/db.py to connect to the correct database.**

Importing data

Importing data is done by uploading responsible_authorities', networks, stations, sampling_points, processes, samples, observing_capabilities and observations as csv files.

See csv_examples.

Use Curl, Postman or similar tools for this.
Set Content-Type to "multipart/form-data" and use the form name "csv".
Example uploading networks with curl

```
curl -X POST -u usr:psw --form "csv=@path/to/networks.csv" https://someurl.com/imports/networks
```

If you upload a content with id's that exists, it will be updated.

There are dependencies between the csv files, so the import order is important. Start with settings, users, responisble_authorities, networks, stations, sampling_points, processes, samples, observing_capabilities and observations

Security

This application uses basic authentication.
The default user:password combination is admin:admin
It is recommended to create a new user and delete the default user.
To do this this with **json** payload, do the following:

Add new user:

```
PAYLOAD {"username":"new_username", "password":"new_password"}
POST imports/users
```

Delete default user:

```
DELETE imports/users/admin
```

Settings

The namespace needs to be set for dataflow D and E.
Example of namespace is "NO.NILU.AQD"
To change the namespace with **json** payload, do the following:

Update namespace:

```
PAYLOAD {"namespace":"some_namespace"}
POST imports/settings
```

Responisble authorities

Create a csv file with the following headers. All headers must be present:

```
id,name,organisation,address,locator,postcode,email,phone,website,is_responsible_r
eporter
```

id: A unique self created id. (Required)
name: The name of the reporter (Required)
organisation: The name of the reporting organisation (Required)
address: The address of the reporting organisation (Required)
locator: City of the reporting organisation (Required)
postcode: Post code of the reporting organisation (Required)
email: Email to the reporter (Required)
phone: Phone number to the reporter (Required)
website: Url to the reporting organisation (Required)
is_responsible_reporter: One reporter needs to be set to true. This reporter will be used for AQD_ReportingHeader. (Required)

Insert or update one or multiple reporters:

```
POST imports/responsible_authorities
```

Get all imported reporters:

```
GET imports/responsible_authorities
```

Delete a specific reporter:

```
DELETE imports/responsible_authorities/{id}
```

Networks

Create a csv file with the following headers. All headers must be present:

```
id,name,responsible_authority_id,organisational,begin_position,end_position,aggregation_timezone
```

id: The EEA code for your network with prefix. eg NET_NO019A (Required)
name: A name for the network. Can be anything you want (Required)
responsible_authority_id: The id to the responsible authority. This id must already exist in the system (Required)
organisational: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/organisationallevel (Required)
begin_position: When the network started measuring. ie 2000-01-01T00:00:00+01:00 (Required)
end_position: When the network stopped measuring. ie 2000-01-01T00:00:00+01:00 (Optional)
aggregation_timezone: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/timezone (Required)

Insert or update one or multiple networks:

```
POST imports/networks
```

Get all imported networks:

```
GET imports/networks
```

Delete a specific network:

```
DELETE imports/networks/{id}
```

Stations

Create a csv file with the following headers. All headers must be present:

```
id,name,begin_position,end_position,network_id,municipality,eoi_code,national_station_code,latitude,longitude,epsg,altitude,mobile,area_classification,distance_junction,traffic_volume,heavy_duty_fraction,street_width,height_facades
```

id: The EEA code for your station with prefix. eg STA_NO0083A (Required)
name: A name for the station. Can be anything you want (Required)
begin_position: When the station started measuring. ie 2000-01-01T00:00:00+01:00 (Required)
end_position: When the station stopped measuring. ie 2000-01-01T00:00:00+01:00 (Optional)
network_id: The id to the network. This id must already exist in the system (Required)
municipality: The name of the municipality where the station is located. (Optional)
eoi_code: The EEA code for your station. (Required)
national_station_code: The national/local id for the station. (Required)
latitude: The latitude of the station. (Required)
longitude: The longitude of the station. (Required)
epsg: The epsg integer code of the station. eg 4326 (Required)
altitude: The altitude of the station. (Required)
mobile: Is the station mobile. True or false (Required)
area_classification: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/areaclassification (Required)
distance_junction: (Optional)
traffic_volume: (Optional)
heavy_duty_fraction: (Optional)
street_width: (Optional)
height_facades: (Optional)

Insert or update one or multiple stations:

```
POST imports/stations
```

Get all imported stations:

```
GET imports/stations
```

Delete a specific station:

```
DELETE imports/stations/{id}
```

Sampling Points

Create a csv file with the following headers. All headers must be present:

```
id,assessment_type,station_id,station_classification,main_emission_sources,traffic
_emissions,heating_emissions,industrial_emissions,distance_source,begin_position,e
nd_position,mobile
```

id: A unique id with prefix. Recommended format is stationid_componentid_uniqueid. ie SPO_NO0083A_9_1023 (Required)
assessment_type: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/assessmenttype (Required)
station_id: The id to the station. This id must already exist in the system (Required)
station_classification: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/stationclassification (Required)
main_emission_sources: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/emissionsource (Optional)
traffic_emissions: The traffic emissions. (Optional)
heating_emissions: The heating emissions. (Optional)
industrial_emissions: The industrial emissions. (Optional)
distance_source: The distance source. (Optional)
begin_position: When the station started measuring for pollutant. ie 2000-01-01T00:00:00+01:00 (Required)
end_position: When the station stopped measuring for pollutant. ie 2000-01-01T00:00:00+01:00 (Optional)
mobile: Is the sampling point mobile. True or false (Required)

Insert or update one or multiple sampling points:

```
POST imports/sampling_points
```

Get all imported sampling points:

```
GET imports/sampling_points
```

Delete a specific sampling point:

```
DELETE imports/sampling_points/{id}
```

Processes

Create a csv file with the following headers. All headers must be present:

```
id,responsible_authority_id,measurement_type,measurement_method,other_measurement_
method,sampling_method,other_sampling_method,analytical_tech,other_analytical_tech
,sampling_equipment,other_sampling_equipment,measurement_equipment,other_measureme
nt_equipment,equiv_demonstration,equiv_demonstration_report,detection_limit,detect
ion_limit_uom,uncertainty_estimate,documentation,qa_report,duration_number,duratio
n_unit,cadence_number,cadence_unit
```

id: A unique id with prefix. Recommended format is stationid_componentid_uniqueid. ie SPP_NO0083A_9_1023 (Required)
responsible_authority_id: The id to the responsible authority. This id must already exist in the system (Required)
measurement_type: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/measurementtype (Required)
measurement_method: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/measurementmethod (Optional)
other_measurement_method: A text if measurement_method is set to other. (Optional)
sampling_method: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/samplingmethod (Optional)
other_sampling_method: A text if sampling_method is set to other. (Optional)
analytical_tech: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/analyticaltechnique (Optional)

other_analytical_tech: A text if analytical_tech is set to other. (Optional)
sampling_equipment: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/samplingequipment (Optional)
other_sampling_equipment: A text if sampling_equipment is set to other. (Optional)
measurement_equipment: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/measurementequipment (Optional)
other_measurement_equipment: A text if measurement_equipment is set to other. (Optional)
equiv_demonstration: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/equivalencedemonstrated (Optional)
equiv_demonstration_report: A text if equiv_demonstration is set to other. (Optional)
detection_limit: The detection limit (Optional)
detection_limit_uom: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/uom/concentration (Optional)
uncertainty_estimate: The uncertainty estimate. (Optional)
documentation: The documentation. (Optional)
qa_report: The QA report. (Optional)
duration_number: The duration number. (Required)
duration_unit: An EEA url. See http://dd.eionet.europa.eu/vocabulary/uom/time (Required)
cadence_number: The cadence number. (Required)
cadence_unit: An EEA url. See http://dd.eionet.europa.eu/vocabulary/uom/time (Required)

Insert or update one or multiple processes:

```
POST imports/processes
```

Get all imported processes:

```
GET imports/processes
```

Delete a specific process:

```
DELETE imports/processes/{id}
```

Samples

Create a csv file with the following headers. All headers must be present:

```
id,inlet_height,building_distance,kerb_distance
```

id: A unique id. Recommended format is stationid_componentid_uniqueid_counter. ie NO0083A_9_1023_1 (Required)
inlet_height: The inlet height in meters. (Required)
building_distance: The building distance in meters. (Optional)
kerb_distance: The kerb distance in meters. (Optional)

Insert or update one or multiple samples:

```
POST imports/samples
```

Get all imported samples:

```
GET imports/samples
```

Delete a specific sample:

```
DELETE imports/samples/{id}
```

Observing capabilities

Create a csv file with the following headers. All headers must be present:

```
id,begin_position,end_position,pollutant,sampling_point_id,sample_id,process_id
```

id: A unique id. Recommended format is stationid_componentid_uniqueid_counter. ie NO0083A_9_1023_1 (Required)
begin_position: When the stations started measuring with capability. ie 2000-01-01T00:00:00+01:00 (Required)
end_position: When the stations stopped measuring with capability. ie 2000-01-01T00:00:00+01:00 (Optional)
pollutant: An EEA url. See http://dd.eionet.europa.eu/vocabulary/aq/pollutantn (Required)

sampling_point_id: The id to the sampling point. This id must already exist in the system (Required)
sample_id: The id to the sample. This id must already exist in the system (Required)
process_id_id: The id to the process. This id must already exist in the system (Required)

Insert or update one or multiple observing capabilities:

```
POST imports/observing_capabilities
```

Get all imported observing capabilities:

```
GET imports/observing_capabilities
```

Delete a specific observing capability:

```
DELETE imports/observing_capabilities/{id}
```

Observations

This should be imported on a regular basis, ie hourly.
Create a csv file with the following headers. All headers must be present:

```
sampling_point_id,begin_position,end_position,value,verification_flag,validation_flag,concentration,timestep
```

sampling_point_id: The id to the sampling point. This id must already exist in the system (Required)
begin_position: When the measurement started. ie 2000-01-01T00:00:00+01:00 (Required)
end_position: When the measurement stopped. ie 2000-01-01T01:00:00+01:00 (Required)
value: The measured value. (Required)
verification_flag: The integer verification flag. See notation in http://dd.eionet.europa.eu/vocabulary/aq/observationverification (Required)
validation_flag: The integer validation flag. See notation in http://dd.eionet.europa.eu/vocabulary/aq/observationvalidity (Required)
concentration: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/uom/concentration (Required)
timestep: An EEA url.
See http://dd.eionet.europa.eu/vocabulary/aq/primaryObservation (Required)

Insert or update one or multiple observations:

```
POST imports/observations
```

Get all imported observations:

```
GET imports/observations
```

Delete a specific observation:

```
DELETE imports/observations/{sampling_point_id}/{end_position}
```

Dataflows

Dataflows do not require authentication.

D

To get dataflow D set the year and the timezone in the url. ie year=2000&tz=1

```
GET dataflows/d ?year={year}&tz={timezone:int}
```

E1A

To get dataflow E1A set the year and the timezone in the url. ie year=2000&tz=1

```
GET dataflows/e1a ?year={year}&tz={timezone:int}
```

E2A

To get dataflow E2A set the last_request in the url. ie 2000-01-01T00:00:00
This will get all imported/updated data after the set date.

```
GET dataflows/e2a ?last_request={YYYY-MM-DDTHH:mm:ss}
```

NILU – Norwegian Institute for Air Research

NILU – Norwegian Institute for Air Research is an independent, nonprofit institution established in 1969. Through its research NILU increases the understanding of climate change, of the composition of the atmosphere, of air quality and of hazardous substances. Based on its research, NILU markets integrated services and products within analyzing, monitoring and consulting. NILU is concerned with increasing public awareness about climate change and environmental pollution.

NILU's values: *Integrity - Competence - Benefit to society*
NILU's vision: *Research for a clean atmosphere*

NILU – Norwegian Institute for Air Research
P.O. Box 100, NO-2027 KJELLER, Norway

E-mail: nilu@nilu.no
http://www.nilu.no

ISBN: 978-82-425-2872-8
ISSN: 2464-3327

Norsk institutt for luftforskning
Norwegian Institute for Air Research