Intended for
European Commission, DG Environment

Reference
Specific Contract 070201/2018/787684/SFRA/ENV.C.2

Date
March 2020

SUPPORT ON THE IMPLEMENTATION OF THE URBAN WASTE WATER TREATMENT DIRECTIVE (91/271/EEC)

STANDALONE QA/QC MODULE - INSTALLATION GUIDE AND USER MANUAL OF UWWTD-SIIF PLATFORM – VERSION 1.0

RAMBOLL

International Office for Water

SUPPORT ON THE IMPLEMENTATION OF THE URBAN WASTE WATER TREATMENT DIRECTIVE (91/271/EEC) STANDALONE QA/QC MODULE - INSTALLATION GUIDE AND USER MANUAL OF UWWTD-SIIF PLATFORM – VERSION 1.0

Project name	**Support on the implementation of the Urban Waste Water Treatment Directive (91/271/EEC)**
Project no.	**Specific Contract 070201/2018/787684/SFRA/ENV.C.2**
Recipient	**European Commission, DG Environment**
Document type	**Standalone QA/QC module - Installation guide and user manual**
Version	**draft**
Date	**06/03/2020**
Authors	**Benoît Fribourg-Blanc & Nicolas Dhuygelaere (OIEau)**

Ramboll
35, Square de Meeûs
1000 Brussels
Belgium

T +32 02 737 96 80
F +32 02 737 96 99
https://ramboll.com

&

OIEau
15 rue Edouard Chamberland
87065 Limoges Cedex

T +33 555 11 47 97
F +33 555 11 47 48

CONTENTS

1.	**Introduction**	**1**
2.	**Prerequisites and system settings**	**1**
2.1.	Recommended environment	1
2.2.	System setting	1
3.	**Install the QA/QC standalone module**	**3**
3.1.	Clone GitHub repository	3
3.2.	Install python libraries	3
3.3.	Apache settings (optional)	4
3.4.	Run the QA/QC standalone module	5
4.	**Use the QA/QC standalone module**	**6**
4.1.	Where can I found an online version of the module?	6
4.2.	WPS basics	6
4.3.	"UWWTDParser" service	7
4.4.	Testing interface	7
5.	**List of test**	**10**
5.1.	Implemented tests	10
5.2.	How to implement a new custom test	19
6.	**Annexe 1: XML Output of UWWTDParser service**	**20**

1. INTRODUCTION

This document is presenting how to install the UWWTD SIIF QA/QC standalone module. This API can be invoked for testing article 15 report file XML in order to check if XML syntax and QA/QC rules defined by EEA are well implemented.
The API is based on PyWPS framework (https://pywps.org/). PyWPS is an implementation of the Web Processing Service standard from the Open Geospatial Consortium. PyWPS is written in Python.
An online instance of the standalone module is host by Office International de l'Eau and available here: https://uwwtd.eu/qa-qc-module/wps
The module is freely distribute and reusable under Creative Commons licence: https://creativecommons.org/licenses/by/4.0/ . Code are available on a GitHub repository https://github.com/OIEau/uwwtd-wps-parser

2. PREREQUISITES AND SYSTEM SETTINGS

2.1. Recommended environment

The standalone QA/QC module can be installed on different kinds of webserver with linux or windows operating system and different versions of python language (branch 2.7 or 3.x). However, the module was developed under Centos 7, apache 2.4 and python 3.6, so we recommend to use the same environment.
Operating system: Linux like Centos 7
RAM: 2 GBytes
CPU: 2 core
Hard disk: 10 GBytes
Python: 3.6 or more
Web Server: Apache 2.4 (optional)
The hosting server must have an Internet access and allow XML file upload.

2.2. System setting

2.2.1. Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. This programming language is used by the pyWPS framework operated by the QA/QC module.

```
yum install epel-release
yum groupinstall 'Development Tools'
yum install python36 python-pip
pip install -U pip
pip install -U virtualenv
virtualenv -p python3 py36
```

2.2.2. Git

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
Git is required for getting a clone of the QA/QC module on the GitHub repository. In theory, Git is already installed during the previous step with the line: yum groupinstall 'Development Tools'

If it is not the case you can run the following command line in the console:

```
yum install git
```

2.2.3. GDAL

GDAL is an opensource translator library for raster and vector geospatial data formats that is released under an X/MIT style Open Source License by the Open Source Geospatial Foundation. This library is used by pyWPS for geographical data processing.
GDAL is installed with the following command line:

```
yum install gdal gdal-devel
```

2.2.4. Httpd Apache (optional)

Httpd Apache is an opensource web server, the installation is not mandatory because the standalone QA/QC module can run without this webserver on port 5000. But for security and load management reasons, it is better to set all input & output flows with apache.
Httpd Apache is installed with the following command line:

```
yum install httpd
apachectl start
systemctl enable httpd.service
```

3. INSTALL THE QA / QC STANDALONE MODULE

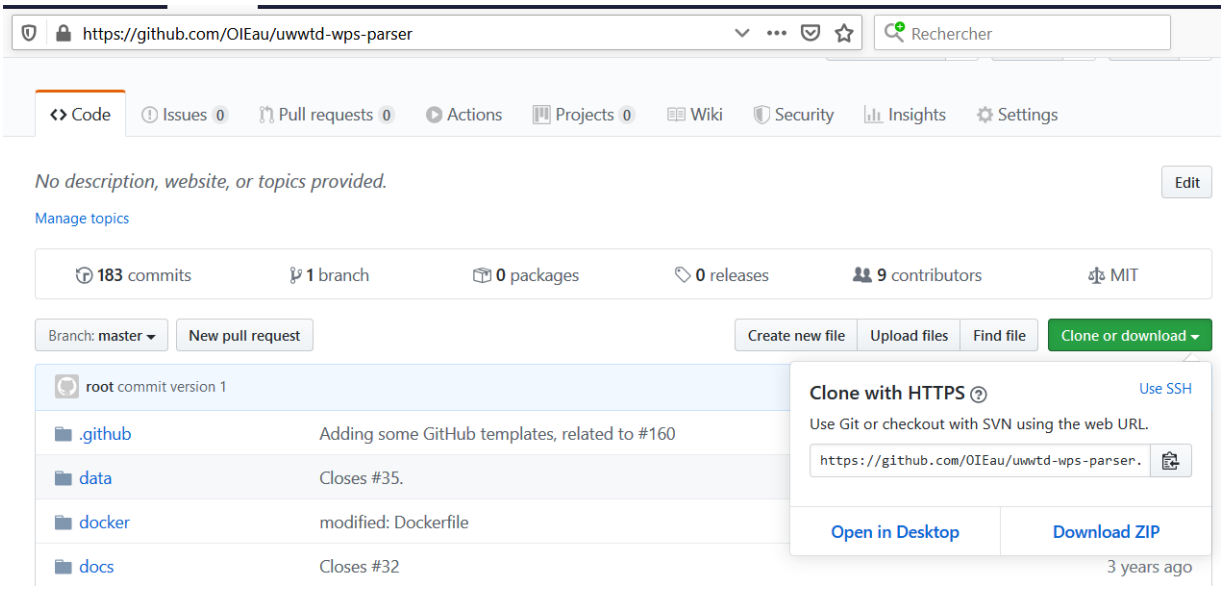
The QA/QC standalone module is based on opensource project PyWPS. The Github repository: https://github.com/OIEau/uwwtd-wps-parser , contain all needed source code for PyWPS install but additional information can be found here : https://pywps.readthedocs.io/en/latest/install.html

3.1. Clone GitHub repository

You can get source code on git repository with the following commands

```
cd /var/www
git clone https://github.com/OIEau/uwwtd-wps-parser.git siif_parser
```

If these command lines don't work, you can download the zip file at: https://github.com/OIEau/uwwtd-wps-parser, and unzip it in path /var/www/siifparser of your server:



3.2. Install python libraries

3.2.1. Required libs for PyWPS

PyWPS need some specific libraries and environment variables for running, following command lines will set required ones:

```
cd /var/www/siifparser
pip install GDAL==1.10.0 --global-option=build_ext --global-option="-
I/usr/include/gdal"
pip install -r requirements.txt
pip install -r requirements-dev.txt
```

3.2.2. Required libs for QA/QC module

Additional python libraries are needed for the QA/QC module, following command lines will install required ones:

```
pip install lxml
pip install sqlite3
```

3.3. Apache settings (optional)

This step is needed if you want Apache as a front webserver running on port 80. By default the QA/QC module is running on port 5000. However this settings is strongly recommended.

Execute the following command lines in the console:

```
cd /etc/httpd/conf.d
cat > siifqaqc.conf
```

And then, add the following content in the file:

```
ProxyPass /siifparser http://localhost:5000
ProxyPassReverse /siifparser http://localhost:5000

#Alias /siifparser "/var/www/siifparser"
<Directory "/var/www/siifparser/">
        AllowOverride All
</Directory>
```

You can also add these lines in the <VirtualHost> definition.

Nota: after a change in Apache settings, you have to restart the Apache service with the following command line:

```
apachectl graceful

OR

apachectl restart
```

3.4. Run the QA/QC standalone module

For starting the PyWPS server, you have to run the “run.sh” script in the project root:

```
cd /var/www/siifparser
./run.sh
```

Nota: this script have be executed at each server reboot.

You can test the service by calling the following URL:
http://[*DNS or IP address of my serveur*]/siifparser/wps?service=WPS&request=GetCapabilities

4. USE THE QA/QC STANDALONE MODULE

4.1. Where can I found an online version of the module?

Office International de l'Eau maintain and host his own instance of the SIIF QA/QC standalone module. This service is available at the URL: https://uwwtd.eu/qa-qc-module/wps and implement all the WPS standard operation like the GetCapabilities operation: https://uwwtd.eu/qa-qc-module/wps?service=WPS&request=GetCapabilities .

4.2. WPS basics

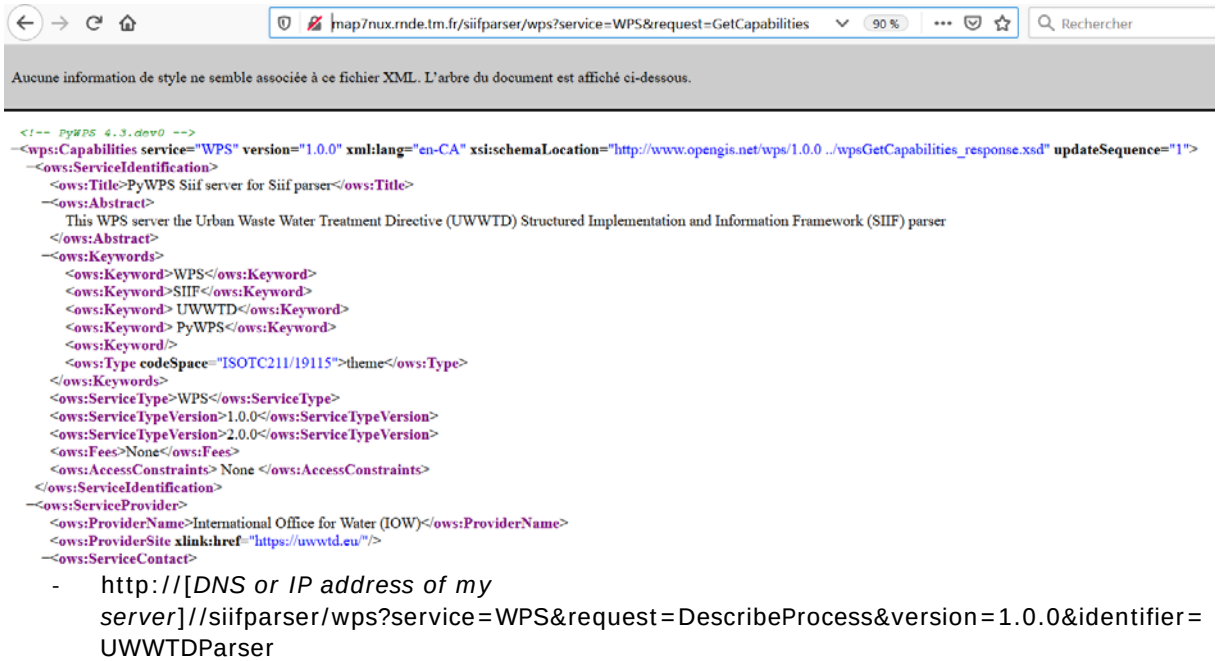
The QA/QC standalone module is based on PyWPS framework. PyWPS implement the OGC Web Processing Service (WPS): http://www.ogc.org/standards/wps . This standard define interface and method of the webservice.

WPS defines three operations:

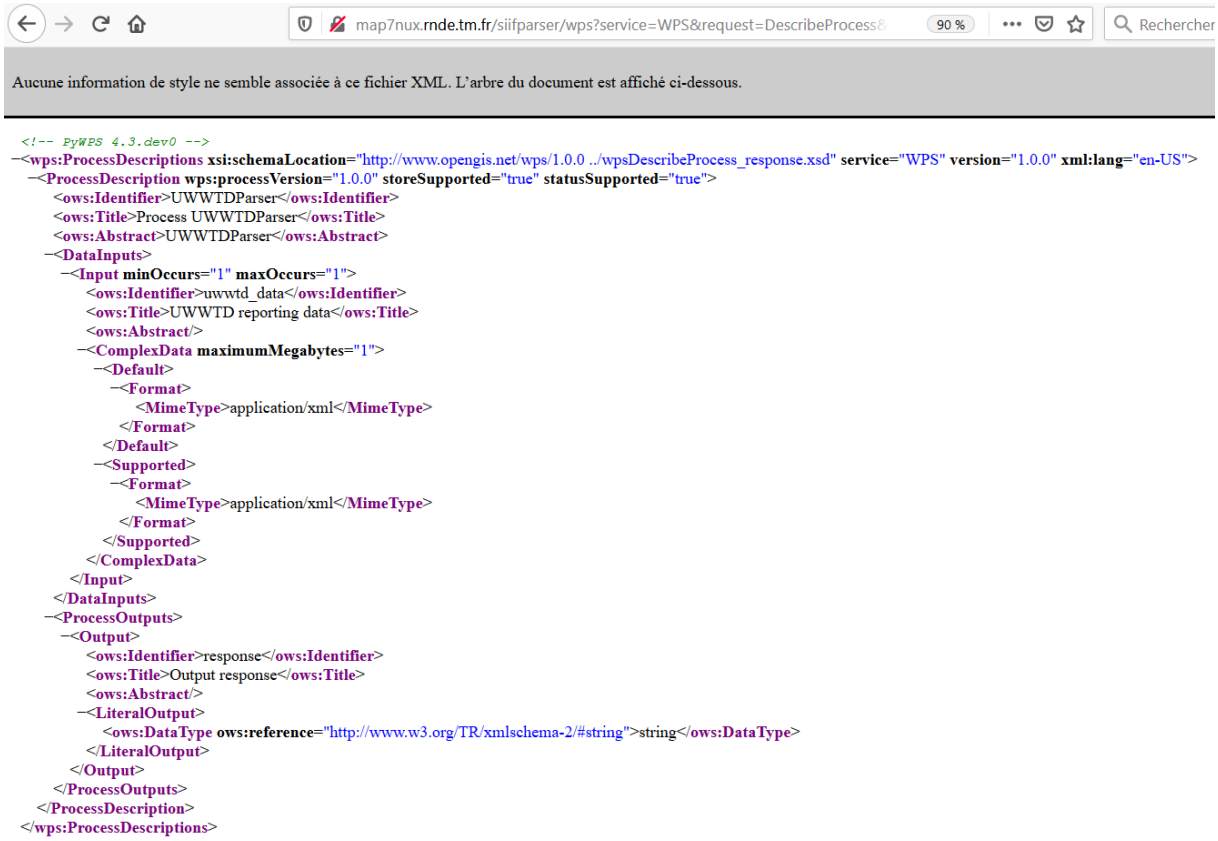
1. *GetCapabilities* returns service-level metadata
2. *DescribeProcess* returns a description of a process including its inputs and outputs
3. *Execute* returns the output(s) of a process

You can invoke GetCapabilities and DescribeProcess on your instance with the following URL and obtain an XML answer as bellow:

- http://[*DNS or IP address of my server*]/siifparser/wps?service=WPS&request=GetCapabilities



- http://[*DNS or IP address of my server*]//siifparser/wps?service=WPS&request=DescribeProcess&version=1.0.0&identifier=UWWTDParser



map7nux.rnde.tm.fr/siifparser/wps?service=WPS&request=DescribeProcess&

Aucune information de style ne semble associée à ce fichier XML. L'arbre du document est affiché ci-dessous.

```
<!-- PyWPS 4.3.dev0 -->
<wps:ProcessDescriptions xsi:schemaLocation="http://www.opengis.net/wps/1.0.0 ../wpsDescribeProcess_response.xsd" service="WPS" version="1.0.0" xml:lang="en-US">
  <ProcessDescription wps:processVersion="1.0.0" storeSupported="true" statusSupported="true">
    <ows:Identifier>UWWTDParser</ows:Identifier>
    <ows:Title>Process UWWTDParser</ows:Title>
    <ows:Abstract>UWWTDParser</ows:Abstract>
    <DataInputs>
      <Input minOccurs="1" maxOccurs="1">
        <ows:Identifier>uwwtd_data</ows:Identifier>
        <ows:Title>UWWTD reporting data</ows:Title>
        <ows:Abstract/>
        <ComplexData maximumMegabytes="1">
          <Default>
            <Format>
              <MimeType>application/xml</MimeType>
            </Format>
          </Default>
          <Supported>
            <Format>
              <MimeType>application/xml</MimeType>
            </Format>
          </Supported>
        </ComplexData>
      </Input>
    </DataInputs>
    <ProcessOutputs>
      <Output>
        <ows:Identifier>response</ows:Identifier>
        <ows:Title>Output response</ows:Title>
        <ows:Abstract/>
        <LiteralOutput>
          <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string">string</ows:DataType>
        </LiteralOutput>
      </Output>
    </ProcessOutputs>
  </ProcessDescription>
</wps:ProcessDescriptions>
```

4.3. "UWWTDParser" service

The "UWWTDParser" is the service for running QA/QC tests on Article 15 UWWTD's report file. Currently this service only support the following versions of the report:

- XML file for Article 15 of 10th report version 2016 defined by the data dictionary http://dd.eionet.europa.eu/datasets/3194

The service can be invoke with the "Execute" operation with the following input parameters:

Parameter ID	Parameter label	Type	Constraint
uwwtd_data	UWWTD reporting data	XML File	minOccurs="1" maxOccurs="1" maximumMegabytes="1"

And output following values

Parameter ID	Parameter label	Type	Constraint
response	Output response	string	

See annexe 1 for output example.

4.4. Testing interface

For test your ows system, you can create a testing interface using the following script write in PHP. You can create a php file in wamp server or apache directory.

```
<! DOCTYPE html>
<html>
<body>
   <form action="" method="post" enctype="multipart/form-data">
      <input type="file" name="fileToUpload" id="fileToUpload">
      <input type="submit" value="Upload UWWTD reporting file" name="submit">
   </form>
   <br/>
</body>

<?php

$wps_url = 'https://uwwtd.eu/qa-qc-module/wps'; //Change this URL with your own
instance of QA/QC module

if(isset($_POST["submit"])) {
   if (isXML($_FILES["fileToUpload"]["tmp_name"])) {
      $wps_request = encapsulateXmlInWps($_FILES["fileToUpload"]["tmp_name"]);
      $results     = sendRequesToWps($wps_request, $wps_url);
      echo "<div><textarea rows='50'
cols='200'>".html_entity_decode($results)."</textarea></div>";
   }
   else {
      echo "Incorrect XML file";
   }
}

function isXML($url)
{
   libxml_use_internal_errors(false);
   $doc = new DOMDocument('1.0', 'utf-8');
   $doc->loadXML(file_get_contents($url));

   $errors = libxml_get_errors();
   if (empty($errors)) {
      return true;
   }
   return false;
}

function encapsulateXmlInWps($file)
{
   $xml_content = file_get_contents($file);
   $xml_content = str_replace('<?xml version="1.0" encoding="UTF-8"?>', '', $xml_content);
   $xml_content = str_replace('<!--?xml version="1.0" encoding="UTF-8"?-->','',
$xml_content);
```

```
    return
        <<<EOD
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0"
    xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
    <ows:Identifier>UWWTDParser</ows:Identifier>
    <wps:DataInputs>
        <wps:Input>
            <ows:Identifier>uwwtd_data</ows:Identifier>
            <wps:Data>
                <wps:ComplexData mimeType="application/xml">
                $xml_content
                </wps:ComplexData>
            </wps:Data>
        </wps:Input>
    </wps:DataInputs>
</wps:Execute>
EOD;
}

function sendRequesToWps($xml, $url)
{

    // initialise the curl request
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: text/xml'));
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $xml);
    $result = curl_exec($ch);
    curl_close($ch);
    return $result;
}

?>
```

5. LIST OF TEST

5.1. Implemented tests

Implemented tests are based on EEA's QA/QC tests used in Reportnet. All XSD validation test are made but the full set of custom rules was not implemented currently.

Entity	Test type	Test name	DD elements	Test description
UWWTP	XSD validation	1. Mandatory values test	all elements flagged as mandatory in the DD	Tested the presence of mandatory values - uwwState, repCode, uwwCode, uwwName, uwwCollectingSystem. *The test returns mandatory parameters not reported.*
UWWTP	Custom rule	2. Record uniqueness test	uwwCode	Tested uniqueness of the records. uwwCode value must be unique for each record in the table. No multiplicities can exist. *The test returns uwwCodes, which are not unique (have duplicities).*
UWWTP	XSD validation	3. Data types test	all elements	Tested that the format of reported values matches the Data Dictionary specifications. *The test returns parameters not reported in data format as specified in the Data dictionary.*
UWWTP	XSD validation	4. Valid codes test	all elements with codelist or vocabulary	Tested the correctness of values against the respective codelists. Checked values are uwwState, uwwCollectingSystem, uwwNUTS, uwwPrimaryTreatment, uwwSecondaryTreatment, uwwOtherTreatment, uwwNRemoval, uwwPRemoval, uwwUV, uwwChlorination, uwwOzonation, uwwSandFiltration, uwwMicroFiltration, uwwOther, uwwBOD5Perf, uwwCODPerf, uwwTSSPerf, uwwNTotPerf, uwwPTotPerf, uwwOtherPerf, uwwBadPerformance, uwwAccidents, uwwBadDesign, uwwMethodWasteWaterTreated. *The test returns parameters not reported as values specified in codelists of Data dictionary.*
UWWTP	Custom rule	5. Identifier format test	uwwCode	Tested correctness of the uwwCode value format. The identifier must start with two character long country code of

Entity	Test type	Test name	DD elements	Test description
				the reporting country: {Uppercase(<countryCodePartOfTheEnvelopeURL>). *The test returns uwwCodes reported by a specific country, which do not contain a proper country code.*
UWWTP	Custom rule	7.Coordinates test 1	uwwLatitude, uwwLongitude	Tested whether coordinates have been reported for all active wastewater treatment plants and collecting systems. *The test returns all active UWWTPs/collecting systems with missing coordinates.*
UWWTP	Custom rule	9. Type of treatment test - records after deadline	uwwPrimaryTreatment, uwwSecondaryTreatment, uwwOtherTreatment, uwwNRemoval, uwwPRemoval, uwwUV, uwwChlorination, uwwOzonation, uwwSandFiltration, uwwMicroFiltration	Tested that for each active existing treatment plant, serving an agglomeration with expired implementation deadline, a type of treatment is reported. *Test returns all existing waste water treatment plants serving agglomerations with expired implementation deadline, without reported treatment type.*
UWWTP	Custom rule	11. Treatment type completness test	uwwPrimaryTreatment, uwwSecondaryTreatment, uwwOtherTreatment, uwwNRemoval, uwwPRemoval, uwwUV, uwwChlorination, uwwOzonation, uwwSandFiltration, uwwMicroFiltration	Tested whether a complete information on the treatment type is provided, for all active treatment plants, not only the highest applied treatment. E.g. in case of nutrient removal being the highest type of treatment, also secondary and (if relevant) primary treatment must be reported. *Test returns all existing waste water treatment plants incomplete information on treatment types applied.*
UWWTP	Custom rule	12. Treatment performance for N/P	uwwNTotPerf,uwwNRemoval, uwwPTotPerf, uwwPRemoval	Tested whether the existing treatment plants with the treatment performance for N and/or P is reported as pass, do have Nitrogen and/or Phosphorus treatment reported as in place. *The test returns*

Entity	Test type	Test name	DD elements	Test description
		completness test		*all active existing waste water treatment plants for which performance data indicate that the limits for total nitrogen and total phosphorus stipulated by the directive have been met, however the respective N and P removal has not been reported as in place.*
UWWTP	Custom rule	13. Treatment efficiency and performance for N/P consistency test	uwwPIncomingMeasured, uwwPDischargeMeasured, uwwPDischargeCalculated, uwwPIncomingCalculated, uwwPDischargeEstimated, uwwPIncomingEstimated, uwwPTotPerf, uwwNIncomingMeasured, uwwNDischargeMeasured, uwwNDischargeCalculated, uwwNIncomingCalculated, uwwNDischargeEstimated, uwwNIncomingEstimated, uwwNTotPerf	Tested the consistency between treatment efficiency (expressed as a ratio of a differential between incoming and discharge load and the incoming load) and total nitrogen and total phosphorus performance data. It returns records where the treatment efficiency for Nitrogen and/or Phosphorus are below the values stipulated by the Annex I of the Directive , but N and P performance data is reported as pass. And vice versa- the treatment efficiency is reported within the range required by the Directive, while the performance data is reported as fail. *The test returns all active existing waste water treatment plants with reported inconsistency between treatment efficiency and the performance data for total nitrogen and total phosphorus.*
UWWTP	Custom rule	14. Organic design capacity value presence test	uwwCapacity	Tested whether organic design capacity parameter is provided for all existing active UWWTPs. *The test returns all active existing waste water treatment plants, for which organic design capacity has not been reported.*
UWWTP	Custom rule	15. Incoming and discharged load values correctness test	uwwBODIncomingMeasured, uwwBODDischargeMeasured, uwwBODIncomingCalculated, uwwBODDischargeCalculated, uwwBODIncomingEstimated, uwwBODDischargeEstimated,	Tested whether the reported (measured, calculated or estimated) values of incoming load are larger than corresponding values of discharged load for each active existing treatment plant. T*he test returns all existing active wastewater treatment plants for which reported values of incoming load (BOD, COD, total N and total P) are lower than the values of discharged load.*

Entity	Test type	Test name	DD elements	Test description
			uwwCODIncomingMeasured, uwwCODDischargeMeasured, uwwCODIncomingCalculated, uwwCODDischargeCalculated, uwwCODIncomingEstimated, uwwCODDischargeEstimated, uwwNIncomingMeasured, uwwNDischargeMeasured, uwwNIncomingCalculated, uwwNDischargeCalculated, uwwNIncomingEstimated, uwwNDischargeEstimated, uwwPIncomingMeasured, uwwPDischargeMeasured, uwwPIncomingCalculated, uwwPDischargeCalculated, uwwPIncomingEstimated, uwwPDischargeEstimated	
UWWTP	Custom rule	16. Incoming and discharged load values completness test	uwwBODIncomingMeasured, uwwBODDischargeMeasured, uwwBODIncomingCalculated, uwwBODDischargeCalculated, uwwBODIncomingEstimated, uwwBODDischargeEstimated, uwwCODIncomingMeasured, uwwCODDischargeMeasured, uwwCODIncomingCalculated, uwwCODDischargeCalculated, uwwCODIncomingEstimated,	Tested whether the measured or calculated or estimated values of incoming and discharged load was reported for each active existing treatment plant. *The test returns all existing active wastewater treatment plants for which values of incoming load (BOD, COD, total N and total P) and the values of discharged load have not been reported.*

Entity	Test type	Test name	DD elements	Test description
			uwwCODDischargeEstimated, uwwNIncomingMeasured, uwwNDischargeMeasured, uwwNIncomingCalculated, uwwNDischargeCalculated, uwwNIncomingEstimated, uwwNDischargeEstimated, uwwPIncomingMeasured, uwwPDischargeMeasured, uwwPIncomingCalculated, uwwPDischargeCalculated, uwwPIncomingEstimated, uwwPDischargeEstimated	
UWWTP	Custom rule	18. Potentially overloaded treatment plants test - records after deadline	uwwCapacity, uwwLoadEnteringUWWTP	Tested potentially overloaded treatment plants with monitoring data (BOD and COD) reported as passed. Returning existing active wastewater treatment plants (serving agglomerations with expired implementation deadline) with reported uwwEntering load more than 20 % higher than the reported design capacity. *The test returns all existing active waste water treatment plants for which the entering load is more than 20% higher than the design capacity.*
UWWTP	Custom rule	19. Potentially overloaded treatment plants test - all records	uwwCapacity, uwwLoadEnteringUWWTP	Tested potentially overloaded treatment plants with monitoring data (BOD and COD) reported as passed. The test returns all existing active wastewater treatment plants, serving agglomerations with expired deadline, with reported uwwEntering load more than 20 % higher than the reported design capacity.
Agglomeration	XSD validation	1. Mandatory values test	all elements flagged as mandatory in the DD	Tested the presence of mandatory values - aggState, repCode, aggCode, aggName, aggGenerated, aggPeriodOver, aggDateArt3, aggDateArt4. *The test returns mandatory parameters not reported.*

Entity	Test type	Test name	DD elements	Test description
Agglomeration	Custom rule	2. Record uniqueness test	aggCode	Tested uniqueness of the records. aggCode value must be unique for each record in the table. No multiplicities can exist. *The test returns aggCodes .which are not unique (have duplicities).*
Agglomeration	XSD validation	3. Data types test	all elements	Tested that the format of reported values matches the Data Dictionary specifications. *The test returns parameters not reported in data format as specified in the Data dictionary.*
Agglomeration	XSD validation	4. Valid codes test	all elements with codelist or vocabulary	Tested the correctness of values against the respective codelists. Checked values are aggState, aggChanges, bigCityID, aggMethodC1, aggMethodC2, aggMethodWithoutTreatment, aggNUTS, aggHaveRegistrationSystem, aggExistMaintenancePlan, aggPressureTest, aggVideoInspections, aggOtherMeasures, aggSewageNetwork, aggBestTechnicalKnowledge, aggDilutionRates, aggCapacity, aggAccOverflows. *The test returns parameters not reported as values specified in codelists of Data dictionary.*
Agglomeration	Custom rule	5. Identifier format test	aggCode	Tested correctness of the aggCode value format. The identifier must start with two character long country code of the reporting country: {Uppercase(<countryCodePartOfTheEnvelopeURL>). *The test returns aggCodes reported by a specific country, which do not contain a proper country code.*
Agglomeration	Custom rule	6. Date value tests	aggPeriodOver, aggBeginLife, aggEndLife, aggDateArt3, aggDateArt4, aggDateArt5	Tested correctness of the selected date values: 1. aggBeginLife can't be higher than aggEndLife 2. Year of the aggPeriodOver value must be in the range between 1998 and 2023 for each active agglomeration. *The test returns dates that have not been reported in a format as specified in the Data dictionary. 3.AggDateArt3, AggDateart4, AggDateart5 must be in the range 31/12/1998 and 31/12/2030. 4.Date for the implementation of article 3 requirements canot be greater than the*

Entity	Test type	Test name	DD elements	Test description
				the implementation deadline for art 4 requirements. 5.Date for the implementation of article 3 requirements cannot be greater than the implementation deadline for art 5 requirements. 6.aggDateArt3, aggDateArt4, aggDateArt5 cannot be greater than aggPeriodOver
Agglomeration	Custom rule	7. Wastewater pathway test	aggC1, aggC2, aggPercWithoutTreatment	Tested whether the sum of AggC1, AggC2 and PercWithTreatment gives 100%. In other words the rule checks whether information on wastewater pathways is provided for the entire (100%) generated load of an agglomeration. *The test returns aggCodes for which the values AggC1, AggC2 and percWithoutTreatment do not meet the conditions specified above.*
Agglomeration	Custom rule	8.Coordinates test 1	aggLatitude, aggLongitude	Tested whether coordinates have been reported for all active agglomerations. *The test returns all active agglomerations with missing coordinates.*
Agglomeration	Custom rule	10. Names test	aggName	Tested whether all agglomerations have names. *The test returns agglomerations (aggCodes) with missing agglomeration name.*
Agglomeration	Custom rule	11. Generated load test	aggGenerated	Tested whether Generated load is reported for all active agglomerations, for which implementation deadline has expired. *The test returns agglomerations (aggCode), with expired implementation deadline, without reported generated load.*
Agglomeration	Custom rule	12. Wastewater pathway through collecting system test	aggC1	Tested whether the percentage of total generated load collected through collecting system is provided and meets the following criteria: 0 <= AggC1 <= 100, for all active agglomerations. *The test returns agglomerations (aggCode), for which a reported share of generated load collected in collecting systems (AggC1) is missing or is reported in incorrect format.*
Agglomeration	Custom rule	13. Wastewater pathway	aggC2	Tested whether the percentage of total generated load addressed through Individual Appropriate Systems (IAS) is provided and meets the following criteria: 0 <= AggC2 <= 100 for all active

Entity	Test type	Test name	DD elements	Test description
		addressed through IAS test		agglomerations *The test returns agglomerations (aggCode), for which a reported share of generated load addressed via IAS (AggC2) is missing or is reported in incorrect format.*
Agglomeration	Custom rule	18. Agglomeration Calculation information presence test	aggCalculation	Tested whether the information about the method of calculation of generated load (size of agglomeration) is reported for each agglomeration. *The test returns agglomerations with missing method of calculation of generated load (aggCalculation).*
Agglomeration	Custom rule	19. Generated load addressed via IAS- primary treatment information presence test	aggPercPrimTreatment	Tested whether an information about share of total generated load addressed via Individual Appropriate System (IAS) receiving primary treatment is provided for agglomerations with the fraction of generated load treated via IAS > = 2000 p.e. *The test returns agglomerations with reported share of generated load addressed via IAs >= 2000 p.e., but with missing values on type of treatment (primary).*
Agglomeration	Custom rule	20. Generated load addressed via IAS / secondary treatment information presence test	aggPercSecTreatment	Tested whether an information about a share of total generated load addressed via Individual Appropriate System (IAS) receiving secondary treatment is provided for agglomerations with the fraction of generated load treated via IAS > = 2000 p.e. *The test returns agglomerations with reported share of generated load addressed via IAs >= 2000 p.e., but with missing values on type of treatment (secondary).*
Agglomeration	Custom rule	21. Generated load addressed via IAS /more stringent treatment information presence test	aggPercStringentTreatment	Tested whether an information about a share of total generated load addressed via Individual Appropriate System (IAS) receiving more stringent treatment is provided for agglomerations with the fraction of generated load treated via IAS >= 2000 p.e. *The test returns agglomerations with reported share of generated load addressed via IAs >= 2000 p.e., but with missing values on type of treatment (more stringent).*

Entity	Test type	Test name	DD elements	Test description
Agglomeration	Custom rule	22. Period over for EU15 value test	aggPeriodOver	Tested whether the implementation deadline is provided for all active agglomerations from EU 15 MS is 2005, whereas the earliest is 1998. *The test returns all active agglomerations of EU 15 with reported implementation deadline outside the range 1998-2005.*
Agglomeration	XSD validation	1. Mandatory values test	all elements flagged as mandatory in the DD	Tested the presence of mandatory values - aggState, repCode, aggCode, aggName, aggGenerated, aggPeriodOver, aggDateArt3, aggDateArt4. *The test returns mandatory parameters not reported.*

5.2. How to implement a new custom test

Test are defined in the "/src/siifparser" repository of the QA/QC module by a specific python file (with *.py extension). For each entity a script was created and test are implemented in it:

- For agglomeration :AgglomerationParser.py
- For Urban Waste Water Treatement Plant : UWWTPsParser.py

The main script is "SiifParser.py" so if you want to a new set of rules for a new entity (like Discharge point or receiving area) you have to create a specific file and his reference in the header of the file (e;g : **from src.siifparser.AgglomerationParser import AgglomerationParser**)

6. ANNEXE 1: XML OUTPUT OF UWWTDPARSER SERVICE


```
<?xml version="1.0" encoding="UTF-8"?>
<wps:ExecuteResponse xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0 ../wpsExecute_response.xsd"
service="WPS" version="1.0.0" xml:lang="en-US"
serviceInstance="http://localhost:5000/wps?request=GetCapabilities&service=WPS"
statusLocation="">
   <wps:Process wps:processVersion="1.0.0">
      <ows:Identifier>UWWTDParser</ows:Identifier>
      <ows:Title>Process UWWTDParser</ows:Title>
      <ows:Abstract>UWWTDParser</ows:Abstract>
            </wps:Process>
   <wps:Status creationTime="2020-03-05T12:37:57Z">
      <wps:ProcessSucceeded>PyWPS Process Process UWWTDParser
finished</wps:ProcessSucceeded>
            </wps:Status>
            <wps:ProcessOutputs>
                        <wps:Output>
        <ows:Identifier>response</ows:Identifier>
        <ows:Title>Output response</ows:Title>
        <ows:Abstract></ows:Abstract>
                                    <wps:Data>
          <wps:LiteralData  dataType="string">
=== running test 'verify_DocumentXML' ===
            ERROR
                        Line 2 Column 0: Element
'{http://dd.eionet.europa.eu/namespaces/826}MSLevel': This element is not expected. Expected
is ( {http://dd.eionet.europa.eu/namespaces/816}Reporter ).


=== running test 'verify_aggBeginLife' ===
            PASSED

=== running test 'verify_aggC1' ===
            PASSED

=== running test 'verify_aggC2' ===
            PASSED

=== running test 'verify_aggCalculation' ===
            PASSED

=== running test 'verify_aggCode' ===
            PASSED
```

```
=== running test 'verify_aggCodeFormat' ===
                PASSED

=== running test 'verify_aggDateArt3' ===
                PASSED

=== running test 'verify_aggDateArt3vsaggDateArt4' ===
                PASSED

=== running test 'verify_aggDateArt3vsaggDateArt5' ===
                PASSED

=== running test 'verify_aggDateArt4' ===
                PASSED

=== running test 'verify_aggDateArt5' ===
                PASSED

=== running test 'verify_aggDateArtvsAggPeriodOver' ===
                PASSED

=== running test 'verify_aggGenerated' ===
                PASSED

=== running test 'verify_aggLatitudeLongitude' ===
                PASSED

=== running test 'verify_aggName' ===
                PASSED

=== running test 'verify_aggPercPrimTreatment' ===
                PASSED

=== running test 'verify_aggPercSecTreatment' ===
                PASSED

=== running test 'verify_aggPercStringentTreatment' ===
                PASSED

=== running test 'verify_aggPeriodOver' ===
                PASSED

=== running test 'verify_aggPeriodOverEU12' ===
                PASSED

=== running test 'verify_aggPeriodOverEU15' ===
                PASSED

=== running test 'verify_aggWasterwaterPathway' ===
                PASSED
```

```
=== running test 'verify_allPotentiallyOverloaded' ===
        WARNING
                In the UWWTP 'LU_STEP_111_B001' with uwwState 1 and reported
has passed is potentially overloaded
                In the UWWTP 'LU_STEP_1_B001' with uwwState 1 and reported
has passed is

=== running test 'verify_incomingAndDischargedLoad' ===
        PASSED

=== running test 'verify_incomingAndDischargedLoadCompletness' ===
        WARNING
                In the UWWTP 'LU_STEP_136_B001' with uwwState 1 and
uwwCollectingSystem ISCON is missing values for discharged or incoming load

=== running test 'verify_potentiallyOverloaded' ===
        PASSED

=== running test 'verify_treatmentEfficiency' ===
        WARNING
                In the UWWTP 'LU_STEP_111_B001' with uwwState 1 has
inconsistency between reported performance and incoming measures
                In the UWWTP 'LU_STEP_301_B001' with uwwState 1 has
inconsistency between reported performance and incoming measures
                In the UWWTP 'LU_STEP_122_B002' with uwwState 1 has
inconsistency between

=== running test 'verify_treatmentPerformance' ===
        PASSED

=== running test 'verify_treatmentTypeCompletness' ===
        PASSED

=== running test 'verify_uwwCapacity' ===
        PASSED

=== running test 'verify_uwwCode' ===
        PASSED

=== running test 'verify_uwwCodeFormat' ===
        PASSED

=== running test 'verify_uwwLatitudeLongitude' ===
        PASSED

=== running test 'verify_uwwTreatmentType' ===
        PASSED
```

```
</wps:LiteralData>
                                        </wps:Data>
                                </wps:Output>
                        </wps:ProcessOutputs>
</wps:ExecuteResponse>
```