# Data Quality web services API

## Introduction

The SDQ is implemented in the form of micro webservices with allow to use of the functionality independently or parallelly as required. The service catalogue includes:

* Landsat-8
  + NDVI service
  + Classification service 1/2/3
  + Completeness
  + Accuracy
  + Precision
* Sentinel 2
  + NDVI service
  + Classification service 1/2/3
  + Completeness
  + Accuracy
  + Precision

## Prerequisite

The services used the following packages:

pip install

* Earthpy
* RasterIO
* Flask
* Glob
* Numpy
* urllib3
* JSON
* ploty

## Demonstration

This section demonstrates the sample script to test the SDQ functionality independently.

ID: this is the path to the sentinel data folder with bands in .jp2 form

* **Completeness**

A black screen with blue and red text

Description automatically generated

* **Classification 1**A black background with red and blue text

  Description automatically generated
* **Classification 2**

A computer screen shot of text

Description automatically generated

* Accuracy

A black screen with blue and red text

Description automatically generated

* NDVI

A screenshot of a computer

Description automatically generated

## Interfacing with CAMEO warehouse.

In this section the independent services are merged as one and are connected with a warehouse to search data, fetch metadata, download data and unzip the data for further processing. Figure 16 shows the architecture of interfacing the SDQ service the warehouse using intermediate processing using Dell services.

A screenshot of a computer screen

Description automatically generated

Figure 16. Interfacing with CAMEO warehouse

Figure 17 showcases the workflow of the service which needs metadata-id as input. Then the metadata related to this ID is fetched which in turn provides the data-id which is used to download the data from the S3 bucket and lastly unzip. This data further acts as input to SDQ service.

A diagram of a data flow

Description automatically generated

Figure 17. SDQ interfacing with warehouse workflow

**Experiment 1:**

**Install following packages**

Python 3.8

python -m pip install requests

pip install Flask

pip install earthpy

pip install rasterio

pip install glob

**Step to start web service:**

1. Start the open VPN
2. Start web service on local server

Open terminal with active directory having the dell\_interface files

flask --app dell\_interface run

1. Demo script to call WP3 service
   1. Open “dell interface.ipynb” in visual studio and play
   2. Or you can simply run the given script:

where ID is the metadata id of the data to be processed

import requests

from flask import jsonify

url = "http://127.0.0.1:5000/wp3/SDQ"

#data={"ID":"LC08\_L2SP\_206023\_20230513\_20230518\_02\_T1" }

data={"ID":"2cb98887-8afe-4275-8eca-255957bbbf12" }

response = requests.post(url, json=data)

Sample outcome should be:

{"completeness":{"no\_data": 33.49271561609598},

"Precision" :"HIGH",

"veg\_classification":

{"Concrete\_Percentage": 2.9001876688317374,

"Green\_Percentage": 1.3946787106388228,

"Water\_Percentage": 61.783079257912554},

"basic\_classifcation":

{"Bare\_soil": 2.850083590352935,

"sparse\_veg": 0.05249392718028793,

"Moderate\_veg": 0.0020020864206606493,

"Dence\_veg": 1.3902867755166766,

"water": 61.783079257912554},

"All\_class":

{"No\_veg": 2.8939915845201467,

"Bare\_area": 0.007609226347567622,

"low\_veg": 0.002237339687269883,

"Moderate Veg": 1.3910282289155762,

"high\_veg": 61.783079257912554}}

**Experiment 2: Run Completeness service**

**Install following packages**

Python 3.8

python -m pip install requests

pip install Flask

pip install earthpy

pip install rasterio

pip install glob

**Step to start completeness web service:**

1. Start the open VPN
2. Start web service on local server

Open terminal with active directory to catalog folder

…catalog> flask --app completeness run

1. Demo script to call WP3 service

you can simply run the given script. where ID is the path to the sentinel data folder with bands in .jp2 form. Sample is shown below where bands are stored in IMG\_DATA folder

import requests

from flask import jsonify

url = "http://127.0.0.1:105/wp3/complete"

data={"ID":"IMG\_DATA/\*B?\*.jp2"}

response = requests.post(url, json=data)

outcome will be

{"completeness":{"no\_data": 33.49271561609598}}

**Experiment 3: Run classification of vegetation service**

**Install following packages**

Python 3.8

python -m pip install requests

pip install Flask

pip install earthpy

pip install rasterio

pip install glob

**Step to start classification of vegetation web service:**

1. Start the open VPN
2. Start web service on local server

Open terminal with active directory to catalog folder

…catalog> flask --app classification\_veg run

1. Demo script to call WP3 service

you can simply run the given script. where ID is the path to the sentinel data folder with bands in .jp2 form. Sample is shown below where bands are stored in IMG\_DATA folder

import requests

from flask import jsonify

url = "http://127.0.0.1:105/wp3/classification\_veg"

data={"ID":"IMG\_DATA/\*B?\*.jp2"}

response = requests.post(url, json=data)

**Outcome will be:**

{'Bare\_area': 0.6651603677492776,

'Moderate Veg': 4.976758537629271e-05,

'No\_veg': 6.199711513896769,

'high\_veg': 92.85356468624856,

'low\_veg': 0.0019119047382059116}

**Experiment 4: Run classification of basic components web service**

**Install following packages**

Python 3.8

python -m pip install requests

pip install Flask

pip install earthpy

pip install rasterio

pip install glob

**Step to start classification web service:**

1. Start the open VPN
2. Start web service on local server

Open terminal with active directory to catalog folder

…catalog> flask --app classification\_client\_basic run

1. Demo script to call WP3 service

you can simply run the given script. where ID is the path to the sentinel data folder with bands in .jp2 form. Sample is shown below where bands are stored in IMG\_DATA folder

import requests

from flask import jsonify

url = "http://127.0.0.1:105/wp3/classification\_client\_basic"

data={"ID":"IMG\_DATA/\*B?\*.jp2"}

response = requests.post(url, json=data)

**Outcome will be:**

{'Concrete\_Percentage': 6.855699715661196,

'Green\_Percentage': 0.011133838308432951,

'Water\_Percentage': 92.85356468624856}

**Experiment 5: Run classification of all components web service**

**Install following packages**

Python 3.8

python -m pip install requests

pip install Flask

pip install earthpy

pip install rasterio

pip install glob

**Step to start classification web service:**

1. Start the open VPN
2. Start web service on local server

Open terminal with active directory to catalog folder

…catalog> flask --app classification\_all run

1. Demo script to call WP3 service

you can simply run the given script. where ID is the path to the sentinel data folder with bands in .jp2 form. Sample is shown below where bands are stored in IMG\_DATA folder

import requests

from flask import jsonify

url = "http://127.0.0.1:105/wp3/classification\_all"

data={"ID":"IMG\_DATA/\*B?\*.jp2"}

response = requests.post(url, json=data)

**Outcome will be:**

{'Bare\_soil': 4.494104200052422,

'Dence\_veg': 2.4883792688146356e-06,

'Moderate\_veg': 0.0006627383452609646,

'sparse\_veg': 2.372064127192677,

'water': 92.85356468624856}

**Experiment 6: Run accuracy web service**

**Install following packages**

Python 3.8

python -m pip install requests

pip install Flask

pip install earthpy

pip install rasterio

pip install glob

**Step to start classification web service:**

1. Start the open VPN
2. Start web service on local server

Open terminal with active directory to catalog folder

…catalog> flask --app accuracy run

1. Demo script to call WP3 service

you can simply run the given script. where ID is the path to the sentinel data folder with bands in .jp2 form. Sample is shown below where bands are stored in IMG\_DATA folder

import requests

from flask import jsonify

url = "http://127.0.0.1:105/wp3/accuracy"

data={"ID":"IMG\_DATA/\*B?\*.jp2"}

response = requests.post(url, json=data)

**Outcome will be:**

{'MSE': '{Band0:2696698.0095302938,Band1:3055807.194148095,Band2:3035764.511383945,Band3:4179888.7933215303,Band4:3668596.700997608,Band5:4228195.076601239,Band6:5648951.098427444,Band7:5683913.100733607,Band8:1166524.9957123233,Band9:18134.66123562961,Band10:3790157.4677747586,Band11:2415606.8433365515}',

'NMSE': '{Band0:0.4749066631529798,Band1:0.5322620512866078,Band2:0.5460748693206885,Band3:0.6204238291150582,Band4:0.5646899676884999,Band5:0.5407672844378303,Band6:0.5839553671381944,Band7:0.6116448687478854,Band8:0.5868956025404706,Band9:0.1298321010980939,Band10:0.5045739682890205,Band11:0.5205616016043498}',

'PSNR': '{Band0:-16.177519537207253,Band1:-16.7204558813289,Band2:-16.691877188149952,Band3:-18.080843665795477,Band4:-17.514196103161282,Band5:-18.130746553117202,Band6:-19.3888745430477,Band7:-19.415670686689488,Band8:-12.538136882058211,Band9:5.545709137097084,Band10:-17.655768928904013,Band11:-15.69945890412421}',

'RMSE': '{Band0:1642.162601428462,Band1:1748.0867238635774,Band2:1742.3445443952655,Band3:2044.4776333629895,Band4:1915.3581129902595,Band5:2056.2575414089642,Band6:2376.752216455776,Band7:2384.095866514937,Band8:1080.0578668350706,Band9:134.66499632654958,Band10:1946.8326758544913,Band11:1554.2222631710536}'}