Pyfdb

Release 0.0.3

ECMWF

May 31, 2023

CONTENTS

The **pyfdb** Python module interfaces the FDB client library installed in the system.

CHAPTER

ONE

1. INSTALLATION

Install pyfdb with python3 (>= 3.6) and pip as follows:

```
python3 -m pip install --upgrade git+https://github.com/ecmwf-projects/pyfdb.git@master
```

Or from PyPi (not yet available):

```
python3 -m pip install pyfdb
```

Or with the Conda package manager:

```
conda create -n client python=3.7
conda activate client
python -m pip install --upgrade git+https://github.com/ecmwf-projects/pyfdb.git@master
```

CHAPTER

TWO

2. EXAMPLE

An example of archival, listing and retrieval via pyfdb is shown next. For the example to work, FDB5 must be installed in the system, as well as the shutil, pyeccodes and pyfdb python packages. The GRIB files involved can be found under the `tests/unit/` folder in the pyfdb Git repository (https://github.com/ecmwf-projects/pyfdb).

Initialising FDB

```
import pyfdb
import shutil

fdb = pyfdb.FDB()
```

Archive

```
key = {
    'domain': 'g',
    'stream': 'oper',
    'levtype': 'pl',
    'levelist': '300',
    'date': '20191110',
    'time': '0000',
    'step': '0',
    'param': '138',
    'class': 'rd',
    'type': 'an',
    'expver': 'xxxx'
}

filename = 'x138-300.grib'
fdb.archive(open(filename, "rb").read(), key)

key['levelist'] = '400'
filename = 'x138-400.grib'
fdb.archive(open(filename, "rb").read())

key['expver'] = 'xxxy'
filename = 'y138-400.grib'
fdb.archive(open(filename, "rb").read())
fdb.flush()
```

List

direct function, request as dictionary

```
request = {
    'class': 'rd',
    'expver': 'xxxx',
    'stream': 'oper',
    'date': '20191110',
    'time': '0000',
    'domain': 'g',
    'type': 'an',
    'levtype': 'pl',
    'step': 0,
    'levelist': [300, '500'],
    'param': ['138', 155, 't']
}

for el in pyfdb.list(request):
    print(el)
# {class=rd,expver=xxxx,stream=oper,date=20191110,time=0000,domain=g}{type=an,levtype=pl}
↪{step=0,levelist=300,param=138}
```

direct function, updated dictionary

```
request['levelist'] = ['100', '200', '300', '400', '500', '700', '850', '1000']
request['param'] = '138'

for el in pyfdb.list(request):
    print(el)
# {class=rd,expver=xxxx,stream=oper,date=20191110,time=0000,domain=g}{type=an,levtype=pl}
↪{step=0,levelist=300,param=138}
# {class=rd,expver=xxxx,stream=oper,date=20191110,time=0000,domain=g}{type=an,levtype=pl}
↪{step=0,levelist=400,param=138}
```

fdb object, request as dicitonary

As an alternative, use the created FDB instance and start queries from there

```
request['levelist'] = ['400', '500', '700', '850', '1000']
for el in fdb.list(request):
    print(el)
# {class=rd,expver=xxxx,stream=oper,date=20191110,time=0000,domain=g}{type=an,levtype=pl}
↪{step=0,levelist=400,param=138}

for el in fdb.list():
     print(el)
# {class=rd,expver=xxxx,stream=oper,date=20191110,time=0000,domain=g}{type=an,levtype=pl}
↪{step=0,levelist=300,param=138}
# {class=rd,expver=xxxx,stream=oper,date=20191110,time=0000,domain=g}{type=an,levtype=pl}
↪{step=0,levelist=400,param=138}
# {class=rd,expver=xxxy,stream=oper,date=20191110,time=0000,domain=g}{type=an,levtype=pl}
↪{step=0,levelist=400,param=138}
```

Retrieve

save to file

```
import tempfile
import os

dir = tempfile.gettempdir()

request = {
    'domain': 'g',
    'stream': 'oper',
    'levtype': 'pl',
    'step': '0',
    'expver': 'xxxx',
    'date': '20191110',
    'class': 'rd',
    'levelist': '300',
    'param': '138',
    'time': '0000',
    'type': 'an'
}

filename = os.path.join(dir, 'x138-300bis.grib')
with open(filename, 'wb') as o, fdb.retrieve(request) as i:
    shutil.copyfileobj(i, o)

request['levelist'] = '400'
filename = os.path.join(dir, 'x138-400bis.grib')
with open(filename, 'wb') as o, fdb.retrieve(request) as i:
    shutil.copyfileobj(i, o)

request['expver'] = 'xxxy'
filename = os.path.join(dir, 'y138-400bis.grib')
with open(filename, 'wb') as o, pyfdb.retrieve(request) as i:
    shutil.copyfileobj(i, o)
```

read into python object

```
datareader = pyfdb.retrieve(request)

# reading a small chunk
chunk = datareader.read(10)

print(chunk)
# bytearray(b'GRIB2\x0e\x0e\x01\x00\x00')

print('tell()', datareader.tell())
# tell() 10

# go back (partially) - seek(2)
datareader.seek(2)
print('tell()', datareader.tell())
# tell() 2

# reading a larger chunk
chunk = datareader.read(40)
```

(continues on next page)

(continued from previous page)

```
print(chunk)
# bytearray(b'IB2\x0e\x0e\x01\x00\x004\x80b\x96\xff\x80\x8ad\x01\x90\x13\x0b\n\x00\x00\
↪x01\x00\x00\x00\x00\x00\x00\x15\x00\x00\x00\x00\x00\x00\x00\x00\x00')

# go back - seek(0)
datareader.seek(0)
```

decode GRIB

```
from pyeccodes import Reader
reader = Reader(datareader)
grib = next(reader)
grib.dump()
# [...redacted...]
```