Web Services

Introduction

Everyone using MWA data needs access to metadata about the telescope itself, and about the observation/s that they are using. All of this metadata is stored in a PostgreSQL database on site, and this database is replicated at the Pawsey Centre in Perth. Rather than access this data via SQL queries directly to the database (as these have the potential to place enormous load on the PostgreSQL server), the access mechanism is via a set of 'web services'. Each of these services is a program that runs on a web server and accepts parameters in the URL, accesses the database to obtain and format some data, and returns it to the user.

The URL's for these web services can be entered directly into a web browser, used in shell scripts (with wget), or in your own code. Most of them return structures in JSON format - these are human readable strings, and JSON libraries are easy to find for most programming languages. Example Python code for grabbing data from these web services and JSON-decoding it are at the bottom of this page.

The URLs for these services all have a similar format, starting with the base URL, e.g.

> **Base Webservice URL**
>
> http://ws.mwatelescope.org/

There's an alternate base URL to use if your code is running on one of the computers on-site - contact Andrew Williams if you need it.

The base URL is followed by the service type (eg `metadata/` or `observation/`) and the service name (eg `find`, `obs`, `con`, `fits`, `tconv`, or `temps` for 'metadata' type services, or `errors`, `obs` or `skymap` for 'observation' type services). At the end of the URL are zero or more parameters (a question mark, followed by a series of name=value pairs separated by ampersands). The parameter values may need to be encoded if they contain characters not allowed in URL strings.

The web services are grouped by topic - the groups are:

Calibration web services

These allow the user to download calibration fits, either for a given calibration observation, or by specifying any observation and asking for the fit from the closest, best-matching calibration observation.

They are described here: Calibration web services

Observation web services

These are services that allow the user to view or download information about a single observation, typically in more depth than the metadata services (below), or more human-readable. These services include a full human-readable summary page describing the observation (eg http://ws.mwatelescope.org/observation/obs/?obs_id=1336930080, the services that produce the elements on that page (PPD plots, sky maps, error lists and tables, etc), as well as making the same information (errors, etc) available in machine-readable JSON format.

They are described here: Observation information web services

Metadata web services

These are services that allow the user to view or download observation metadata - array configuration, observation details, metafits files, telescope schedule, data files, etc. The most useful to end-users is probably the 'Find Observations' service, which can be used by scripts to search for observations with whatever criteria you need, or as a human-friendly web page to search for and view observations: http://ws.mwatelescope.org/metadata/find

They are described here: Observation metadata web services

Quality web services

The observation quality web services allow the user to get or set observation quality metrics for EoR observations, derived from the RTS pipeline.

They are described here: Observation Quality Web Services

Progress Bar web services

These form a general-purpose tool for keeping track of progress on a number of tasks being carried out in parallel, on multiple machines. The 'head' process creates a new progress task, then passes the ID of that task to each worker process. Each of the worker process then make web service calls as they work, sending their current state. Another web service generates graphs, dynamically, showing the current state of that progress task across all of the worker nodes.

They are described here: Progress Bar web service, for distributed computing

Tiles web services

These allow the user to view a map of the MWA, with all tiles shown to scale, and showing all tile faults (whole tile or individual dipoles). You can zoom in or out, or click on a tile to re-center on that tile.

They are described here: Tiles web services

Example Python code to use these web services

Python 3.x

```
#### Python 3.X code, courtesy of Nick Swainston ####################
import urllib.request
import json

# Append the service name to this base URL, e.g. 'con', 'obs', etc.
BASEURL = 'http://ws.mwatelescope.org/'

def getmeta(servicetype='metadata', service='obs', params=None):
    """Given a JSON web servicetype ('observation' or 'metadata'), a service name (eg 'obs', find, or 'con')
       and a set of parameters as a Python dictionary, return a Python dictionary containing the result.
    """
    if params:
        # Turn the dictionary into a string with encoded 'name=value' pairs
        data = urllib.parse.urlencode(params)
    else:
        data = ''

    # Get the data
    try:
        result = json.load(urllib.request.urlopen(BASEURL + servicetype + '/' + service + '?' + data))
    except urllib.error.HTTPError as err:
        print("HTTP error from server: code=%d, response:\n %s" % (err.code, err.read()))
        return
    except urllib.error.URLError as err:
        print("URL or network error: %s" % err.reason)
        return

    # Return the result dictionary
    return result
#################################################################
```

Using the getmeta() function

```
# Get and print observation info:

# Some example query data:
fname = '1095675784_20140925102450_gpubox23_01.fits'
starttime = 1095686136

obsinfo1 = getmeta(service='obs', params={'obs_id':starttime})
obsinfo2 = getmeta(service='obs', params={'filename':fname})

coninfo1 = getmeta(service='con', params={'obs_id':starttime})
coninfo2 = getmeta(service='con', params={'filename':fname})

olist = getmeta(service='find', params={'mintime':1097836168, 'maxtime':1097840480, 'obsname':'3C444%'})

faultdict = getmeta(servicetype='observation', service='errors', params={'obs_id':1199602952, 'dictformat':1})
```

The `urlopen()` call above returns a file object (supporting .read(), .readlines(), etc). The above function calls json.load() on that file object to read the contents and convert it to a python dictionary. If you are using the 'fits' or 'skymap' services, or the 'errors' service without specifying 'raw', 'cooked' or 'dictformat' output, the urllib2.urlopen() call returns a file object containing the contents of the FITS, PNG or HTML file, so you do **not** use json.load() on it. Instead, adapt the above function to simply call read() on the file object and return the string. You can then save it to disk, or in the case of the FITS service, pass it to the pyfits.HDUList.fromstring() method to create a FITS structure in memory.

The above code is in the `MandC_Core` git repository, as `mwatools/metaexample.py`

The requests package

An even easier way to use the web service is with the 'requests' library ('pip install requests'):

```
import json
import requests

result = requests.get('http://ws.mwatelescope.org/metadata/obs', data={'obs_id':1095686136})
metadict = json.loads(result.text)


post_data = {'constraints':('or',
                                ('and',
                                    ('>', 'obsid', 1210000000),
                                    ('<', 'obsid', 1220000000),
                                ('not',
                                    ('like', 'projectid', '%000%'))))
            }

result = requests.get('http://ws.mwatelescope.org/quality/select', data=json.dumps(post_data))
resdict = json.loads(result.text)
```