McIDAS-XRD Tutorial

Scripting in Python
updated May 2019

Introduction

The McIDAS-XRD Python package allows users to create Python scripts to run McIDAS-X commands. The advantages of running McIDAS-X commands in a Python environment include but are not limited to:

- Setting up the "mcenv" environment is simpler and removes the shell scripting concepts of EOF and exit 0.
- Users can take advantage of Python's superior text handling capabilities.
- Users can take advantage of Python's superior date/time functionality.
- Python has many libraries for doing math, image manipulation and other data transformations.
- Python is more like a programming language than other traditional McIDAS scripting languages.

The package is part of McIDAS-XRD so certain McIDAS-X commands are not compatible with Python syntax. Specifically, any McIDAS-X command using single quote marks cannot be used. Additionally, the package is only compatible with Linux and macOS operating systems and running McIDAS-X 2017.1 or later.

Note that several McIDAS-X commands that point to Unidata servers (lead.unidata.ucar.edu and atm.ucar.edu) include MCC=NONE to specify that no compression should be used in the data transfer. MCC is a McIDAS-X keyword for MCCOMPRESS. Unidata's ADDE servers don't support MCC=COMPRESS data transfers, meaning only MCC=NONE and MCC=GZIP can be used. For more information, see Using Compressed Data Transfers in the McIDAS-X User's Guide.

https://www.ssec.wisc.edu/mcidas/doc/users_guide/current/intro-33.html

In order to run the sample scripts included in this tutorial, you must download the *mcxpy.zip* file from the McIDAS Website (https://www.ssec.wisc.edu/mcidas/software/x/download/xrd-files/#python) to your **$HOME/mcidas/mcxpy** directory. (Note: you will need to manually create the **mcxpy** subdirectory.)

Table of Contents

Installing McIDAS-XRD Python

Assuming a standard installation of McIDAS-X, where McIDAS was installed as user *mcidas* and is being run from the user account that is set up for McIDAS-X access, run the following commands:

```
cd $HOME
mcxpyinstall
```

mcxpyinstall is the installation script to set up the Python “subprocess” module.

How McIDAS-XRD Python Works

The Python “subprocess” module is used to spawn an instance of the “mcenv shell” as a background process. McIDAS commands are started via the “mcenv” session using Python functions. Command line parameters passed as a single string. For example:

```
mcenv.logon('DEMO 1234')
mcenv.dataloc('ADD BLIZZARD GEOARC.SSEC.WISC.EDU')
```

mcenv.dsinfo('I BLIZZARD')

Neither **dataloc()** nor **dsinfo()** are explicitly defined functions. When an implicit function **mccmd('arg1 arg2 arg3')** is called, the mcenv instance searches the PATH environment variable for a **mccmd.k** McIDAS command/program (which corresponds to the "MCCMD" McIDAS-X command), and then runs **mccmd.k arg1 arg2 arg3** in the mcenv shell subprocess.

Syntax Rules and Examples

To use a Python module in a Python program/script, the mcidasx module must be "imported":

import mcidasx

To begin using the mcidasx module's mcenv "session", create an instance of the mcenv() object and assign it to a local variable ("mc" in this example):

mc = mcidasx.mcenv()

The **-f** (frame size), **-i** (image colors), and **-g** (graphics colors) mcenv options can be passed as arguments to the mcenv() object's instantiation:

mc = mcidasx.mcenv(f=['3@1000x2000', '4@500x500'], i=150, g=16)

The argument passed to **f=** can be either a list of strings (above), or just an individual string:

mc = mcidasx.mcenv(f='10@480x640')

The mcenv executable must be found in the **PATH** environment variable, otherwise the mcenv() instantiation will fail. Existing **PATH** and **MCPATH** environment variables may be sufficient for some uses, but defining these explicitly within a script may be desirable:

```
import os
os.environ['PATH'] = '/path/to/mcidas/dir/bin:%s' % os.environ['PATH']
os.environ['MCPATH'] = '/path/to/project/data/dir:/path/to/mcidas/data'
```

The following is a simple example of the use of the command **IMGLIST**:

```
#!/usr/bin/env python
import mcidasx
import os
os.environ['PATH'] = '/home/mcidas/bin:%s' % os.environ['PATH']
os.environ['MCPATH'] = '%s/mcidas/data:/home/mcidas/data' % os.environ['HOME']
mc = mcidasx.mcenv()
logonOut = mc.logon('DEMO 1234')
datalocOut = mc.dataloc('ADD BLIZZARD GEOARC.SSEC.WISC.EDU')
dsinfoOut = mc.dsinfo('I BLIZZARD')
imglistOut = mc.imglist('BLIZZARD/IMAGES.ALL')
print imglistOut.stdout
print imglistOut.stderr
print imglistOut.retcode
```

In this example **MCPATH** is still set as it is in other McIDAS-X scripts. Initializing the McIDAS environment is done differently than in other scripts. Rather than starting a mcenv subshell, and then running commands in that subshell, the McIDAS environment is started with the command:

```
os.environ['MCPATH'] = '%s/mcidas/data:/home/mcidas/data' % os.environ['HOME']
mc = mcidasx.mcenv()
```

McIDAS-X and mcenv generally write files to the first writeable path in **MCPATH**, although certain situations may arise where this does not occur. This behavior is maintained in the McIDAS-XRD-Python package.

Stdout, Stderr, and Return Codes

When a mcenv command is run, a named tuple containing values for “stdout”, “stderr”, and “retcode” are returned. It is not necessary to capture this tuple unless one of these values is needed.

For example, we might want to add a new remote dataset using **dataloc()**, and then print the output of an **imglist()** call if the **dataloc()** command was successful:

```
dataloc_result = mc.dataloc('ADD GROUP server.domain')
if dataloc_result.retcode == 0:
   imglist_result = mc.imglist('GROUP/DESCRIPTOR FORM=ALL')
   print imglist_result.stdout
```

Some commands might not produce meaningful output, and thus there is no need to capture the output:

```
mc.logon('DEMO 1234')
mc.eg('1')
```

IMGLIST Example

The following is a simple example of the use of the command **IMGLIST**. This script can be found in **$HOME/mcidas/mcxpy/imglist_example.py**.

```
#!/usr/bin/env python
import mcidasx
import os

os.environ['PATH'] = '/home/mcidas/bin:%s' % os.environ['PATH']
os.environ['MCPATH'] = '%s/mcidas/data:/home/mcidas/data' % os.environ['HOME']

mc = mcidasx.mcenv()
mc.dataloc('ADD RTGOESR LEAD.UNIDATA.UCAR.EDU')
result = mc.imglist('RTGOESR/FD MCC=NONE')

print result.stdout
print result.stderr
print result.retcode
```

In this example **MCPATH** is still set as it is in other McIDAS-X scripts. Initializing the McIDAS environment is done differently than in other scripts. Rather than starting a mcenv subshell, and then running commands in that subshell, the McIDAS environment is started with the command:

```
mc = mcidasx.mcenv()
```

Also note that standard out is captured in the variable result and needs to be explicitly written to standard out. The **imglist_example.py** script can be run with the following command:

python $HOME/mcidas/mcxpy/imglist_example.py

The next example is a slightly more advanced version of the previous **IMGLIST** example that takes advantage of Python text handling and date manipulation capabilities. This script can be found in **$HOME/mcidas/mcxpy/imglist_advanced.py**.

```
#!/usr/bin/env python
import datetime
import mcidasx
import os

group = 'RTGOESR'
descriptor = 'FD'
server = 'LEAD.UNIDATA.UCAR.EDU'

os.environ['PATH'] = '/home/mcidas/bin:%s' % os.environ['PATH']
os.environ['MCPATH'] = '%s/mcidas/data:/home/mcidas/data' % os.environ['HOME']

mc = mcidasx.mcenv()
mc.dataloc('ADD %s %s' % (group, server))
image_date = datetime.date(2018, 9, 26)
# for image_date, substitute in today's year, month, day
# for example: datetime.date(2018, 9, 26)
result = mc.imglist('%s/%s DAY=%s TIME=%s FORM=ALL MCC=NONE' % (group, descriptor,
image_date.strftime('%y%j'), '12 18'))

print result.stdout
```

The **imglist_advanced.py** script can be run with the following command:

python $HOME/mcidas/mcxpy/imglist_advanced.py

Exercise 1: Write a short Python script that displays data in a background McIDAS-X window and saves the image as a GIF image.

- Please use dataset **RTGOESR/CONUS** on **LEAD.UNIDATA.UCAR.EDU**.
- An example solution is available on page 10 as well as in the **$HOME/mcidas/mcxpy/bash_vs_python.py** script. However, before checking the solution, it is recommended that you try to complete the exercise on your own.
- Hint: here is a bash script that does this:

```
#!/bin/bash
PATH=$PATH:/home/mcidas/bin
MCPATH=$HOME/mcidas/data:/home/mcidas/data
export PATH MCPATH

mcenv << 'EOF'
dataloc.k ADD RTGOESR LEAD.UNIDATA.UCAR.EDU
imgdisp.k RTGOESR/CONUS.-1 1 BAND=1 LAT=43 90 MCC=NONE
frmsave.k 1 $HOME/mcidas/data/wisconsin_vis_bash.gif
exit 0
EOF

exit
```

The **bash_vs_python.py** script can be run with the following command:

python $HOME/mcidas/mcxpy/bash_vs_python.py

Exercise 2: Run the Python script **$HOME/mcidas/mcxpy/sfclist.py**. Update the script to print out the dew point depression for 0Z. An example solution is available on page 11 as well as **$HOME/mcidas/mcxpy/dewpt.py**. The **sfclist.py** script can be run with the command:

python $HOME/mcidas/mcxpy/sfclist.py KMSN

where "KMSN" is any station. The **dewpt.py** script is run in a similar way, with the exception that a time must also be specified. For example:

python $HOME/mcidas/mcxpy/dewpt.py KMSN 12

Advanced Example

Now for a more advanced example. In this example, we will **IMGCOPY** an archived Meteosat-9 image to a local netcdf dataset, then use netCDF4 and numpy to perform a Normalized Difference Vegetation Index (NDVI) calculation, display the NDVI imagery using matplotlib.pyplot, and finally save the output to a PNG file. This script can be found in **$HOME/mcidas/mcxpy/ndvi.py**. This script can be run with the following command:

python $HOME/mcidas/mcxpy/ndvi.py

The following script imports various modules used by the script. Note that a few of these (matplotlib.pyplot, netCDF4, and numpy) may not be included in the default Python included on macOS or Linux systems. If you don't already have these modules installed, one method of installing them is to first install miniconda, modify your $PATH to include miniconda's bin directory, and then run the following commands:

pip install matplotlib
pip install netCDF4
pip install numpy

```
#!/usr/bin/env python
import matplotlib.pyplot as pyplot
```

```
import mcidasx
import netCDF4
import numpy
import os
import sys

os.environ['PATH']   = "%s:%s/mcidas/bin:/home/mcidas/bin" % (os.environ['HOME'],os.environ['PATH'])
os.environ['MCPATH'] = "%s/mcidas/data:/home/mcidas/data" % os.environ['HOME']

mc = mcidasx.mcenv()
mc.logon('xxx xxx')
# substitute your accounting for 'xxx xxx' above

result1 = mc.dataloc('ADD AMET09 geoarc.ssec.wisc.edu')
if result1.retcode != 0:
    sys.exit(result1.stdout)

mc.dsserve('ADD N/A NCDF 1 9999 TYPE=IMAGE')

msg_ndvi_bands = [1, 2]

imgcopy_string = 'AMET09/FD N/A.{band} SIZE=SAME BAND={band} MAG=-8 DAY=2011/08/31 TIME=12
UNIT=REFL'
for band in msg_ndvi_bands:
    imgcopy_result = mc.imgcopy(imgcopy_string.format(band=band))
    print imgcopy_result.stdout

try:
    # open the NetCDF files
    redBand = netCDF4.Dataset('%s/mcidas/data/A0001.nc', 'r')
    nirBand = netCDF4.Dataset('%s/mcidas/data/A0002.nc', 'r')

    # read data into numpy arrays
    redData = numpy.array(redBand.variables['data'][0])
    nirData = numpy.array(nirBand.variables['data'][0])

    check = numpy.logical_and(redData != 0, nirData != 0)
    ndvi = numpy.where(check, (nirData - redData) / (nirData + redData), 0)

    pyplot.imshow(ndvi, cmap=pyplot.get_cmap('PRGn'), vmin=-1, vmax=1)
    pyplot.savefig('ndvi.png')
```

```
    pyplot.show()

except:
    sys.exit('An error occurred.')
```

Other Python Modules

These Python modules may offer interesting possibilities in combination with McIDAS-X:

- numpy - package for scientific computing
- netCDF4 - python/numpy interface to netCDF
- basemap - library for plotting 2D data on maps
- cartopy - cartographic tools
- gdal - Geospatial Data Abstraction Library bindings

Integrating McIDAS-X into an existing script or workflow involving any of these modules is now very straightforward.

Disclaimer

This package is **NOT** supported by the McIDAS Users' Group (MUG) or any group within SSEC. The software is currently used internally by SSEC Satellite Data Services (SDS) for experimental use and is provided on an "as-is" basis, like all McIDAS-XRD sofware.

Exercise 1: A Python Solution

```
#!/usr/bin/env python
import mcidasx
import os

os.environ['PATH']   = "%s:/home/mcidas/bin" % os.environ['PATH']
os.environ['MCPATH'] = "%s/mcidas/data:/home/mcidas/data" % os.environ['HOME']

mc = mcidasx.mcenv()

mc.dataloc('ADD RTGOESR LEAD.UNIDATA.UCAR.EDU')
```

```
imglistOutput = mc.imgdisp('RTGOESR/CONUS 1 BAND=1 LAT=43 90 MCC=NONE')

frmsave_result = mc.frmsave('1 %s/mcidas/data/wisconsin_vis_python.gif')

print frmsave_result.stdout
```

Exercise 2: A Python Solution

```
#!/usr/bin/python
import mcidasx
import os
import sys

def main(argv):

    if len(argv) < 3:
        print ' Must specify a station and time'
        return
    else:
        station = argv[1]
        time = argv[2]

    os.environ['PATH']   = "%s:%s/mcidas/bin:/home/mcidas/bin" %
(os.environ['HOME'],os.environ['PATH'])
    os.environ['MCPATH'] = "%s/mcidas/data:/home/mcidas/data" % os.environ['HOME']

    mcOutput = run_mcidas(station,time)
    dewPTdepression = get_dewPT(mcOutput)
    dewPTdepression = "{:6.2f}".format(dewPTdepression)
    print ' '
    print 'Dew point depression for ' + station + ' = ' + dewPTdepression

def run_mcidas(station,time):
#
# Create McIDAS environment and run commands
#
    mc = mcidasx.mcenv()
```

```
    datalocString='ADD RTPTSRC ATM.UCAR.EDU'
    datalocOut = mc.dataloc(datalocString)

    sfclistString = '%s TIME=%s MCC=NONE' % (station, time)
    sfclistOut = mc.sfclist(sfclistString)

#
# Break output into list
#
    sfclistOutput = mccmdout2list(sfclistOut.stdout)
    return sfclistOutput

def mccmdout2list(mcoutput):
#
# Inputs output from a McIDAS command and creates a list of lines
#
    line = ''
    count = 0
    lineList = []
    for char in mcoutput:
        if char != '\n':
             line = line + char
        else:
             lineList.append(line)
             count = count + 1
             line = ''
    return lineList

def get_dewPT(outputList):
#
# Remove header and use only hourly observation
#
    outputList.pop(0)
    outputList.pop(0)
    outputList.pop(0)

    recCount = 0
    runningTot = 0
    for line in outputList:
        record = line.split()
```

```
            if 'Number' in record[0]:
                break
            elif 'S' in record[0]:
                pass
            else:
                temperature = float(record[4])
                dewPT = float(record[5])
                print temperature, dewPT

        dewPTdepression = temperature - dewPT
        return dewPTdepression

if __name__ == "__main__":
    main(sys.argv)
```