Using HTTPS to download production and near real-time GPM files from PPS

By Owen Kelley and Chris Cohoon for PPS, 16 May 2024
This document can be downloaded from the PPS website: https://pps.gsfc.nasa.gov

Outline

1. Introduction

The Precipitation Processing System (PPS) at NASA Goddard is the data processing center for NASA's Global Precipitation Measurement (GPM) mission. Following NASA guidance in 2000, PPS replaced FTP access to its online archives with FTPS and HTTPS access. Among these two options, HTTPS has the advantage of avoiding the potential firewall issues you may encounter with FTPS. In typical months, 60% to 90% of PPS file downloads occur by HTTPS rather than by FTPS.

PPS has two online archives, one for near real-time data (jsimpson) and one for production data (arthurhou). The present document describes several options for download files from these archives using HTTPS. One options is to use a web browser to interactively navigate the archive's directory tree and then click to download an individual file from the archive. The top level URLs for the production and near real-time archives are the following:

https://arthurhouhttps.pps.eosdis.nasa.gov/gpmdata/ (production)
https://jsimpsonhttps.pps.eosdis.nasa.gov/ (near real-time)

When you visit these URLs with a web browser, you will be prompted to type in an e-mail address that has been previously registered with PPS. This email address serves as your username and password for these downloads. To register your email address with PPS, visit this URL:

https://registration.pps.eosdis.nasa.gov/ (registration)

If you are planning on writing scripts to download files, then you might want practice interactively using a slightly different URL in your web browser. These URLs below contain the string "text/" immediately after the server name to distinguish them from the normal Apache handled URLs stated above.

*https://arthurhouhttps.pps.eosdis.nasa.gov/**text/**gpmdata/* (alternative for production)
*https://jsimpsonhttps.pps.eosdis.nasa.gov/**text/*** (alternative for near real-time)

The URLs that include "text" will generate a text list the contents of a directory including subdirectories

(entries ending in slash) and download files (entries not ending in slash).

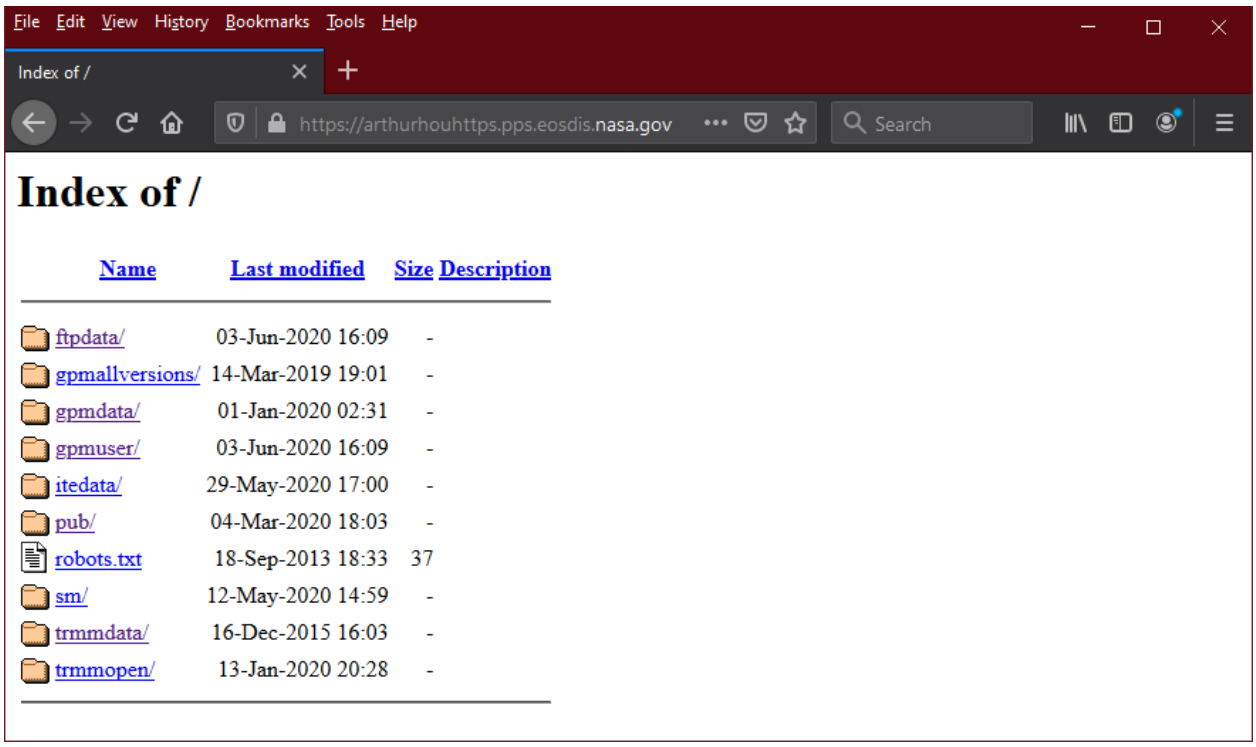
Another option is to use a Linux command-line tool like *wget* or *curl* or a programming language like Python to issue HTTPS requests, navigate the PPS online archives, and download files. For more information about the PPS archive, please contact the PPS Helpdesk, helpdesk@mail.pps.eosdis.nasa.gov. For more information about the GPM mission, visit the GPM website: https://gpm.nasa.gov/.

2. User Registration

Before accessing the PPS archive, register your email address with PPS by visiting the following URL: *https://registration.pps.eosdis.nasa.gov/*. You will use this email address as both the username and password when connecting to a PPS online archive.

3. Download a File Using a Web Browser

Below is a screen capture of what the PPS Production archive's top-level directory looks like when accessed by a web browser and HTTPS via the *jsimpsonhttps* server. To view this directory, go to this URL: *https://arthurhouhttps.pps.eosdis.nasa.gov/*. When prompted, type in an e-mail registered with PPS as the username and password. Alternatively, create a .netrc file that lists your PPS-registered email address as both the username and password.



4. Download a File Using the *curl* or *wget* Utility

The *curl* utility is available on many MacOS systems, and both *curl* and *wget* are available on many Linux systems. If you define a .netrc file on a Linux system to contain your PPS-registered email address, then you may be able to use the following short form URL to download a file from PPS. Depending on the implementation of *wget* or *curl* that you are using, you may need to include various keyword between the utility name and the URL to download.

curl https://arthurhouhttps.pps.eosdis.nasa.gov/text/

wget https://arthurhouhttps.pps.eosdis.nasa.gov/text/

Alternatively, you can specify your PPS-registered email address within the URL, such as in the following snippet of BASH code. In this snippet, replace the "@" symbol in your email address with the "%40" encoding. For example, the author would incode his email address, "owen.kelley@nasa.gov", as "owen.kelley%40nasa.gov".

user="FirstName.LastName%40ServerName.com"

wget https://${user}:${user}@arthurhouhttps.pps.eosdis.nasa.gov/text/

If typed all in one line, the above example would become the following when the author uses his PPS-registered email address. Of course, you would instead use your own PPS-registered email address with the "@" encoded as "%40".

wget https://owen.kelley%40nasa.gov:owen.kelley%40nasa.gov@arthurhouhttps.pps.eosdis.nasa.gov/text/

If you forgot to encode "@" as "%40" or use an email address not registered with PPS, then you will get an error message similar to the following as your HTTPS response from the PPS server:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Unauthorized</title>
</head><body>
<h1>Unauthorized</h1>
<p>This server could not verify that you
are authorized to access the document
requested....
```

As long as the URL includes the "/text/" field, one can include wildcards as well. Without the "/text/" field, a pure Apache URL is required, which does not permit wildcards. For example, the following single-line command would list all the files for April 5, 2015, whose filenames match the wildcard expression "*GMI*S08*". Note that, when a wildcard expression is used in the URL, you must enclose the URL in double quotes to avoid having the Linux shell interfere with passing the wildcard expression to the PPS server. None of these examples will work unless you replace the string **EMAIL** with your PPS-registered email address with the "@" symbol encoded as "%40" as in "owen.kelley%40nasa.gov".

curl
*"https://**EMAIL**:**EMAIL**@arthurhouhttps.pps.eosdis.nasa.gov/text/gpmdata/2015/04/05/*/*GMI*S08**"*

When downloading an individual file, do not put a trailing "/" at the end of the request. If a trailing "/" is placed after the file name, then the server will return a "404 NOT FOUND" response. Include the "-O" keyword to make *curl* preserve the filename that exists in the PPS online archive. Note that when downloading a single file, it is possible to either include or omit the /text/ field and the results are the same.

curl -O
*https://***EMAIL***:***EMAIL***@arthurhouhttps.pps.eosdis.nasa.gov/gpmdata/2015/04/05/1C/1C-R.GPM.GMI.XCAL2016-C.20150405-S082303-E095535.006250.V07A.HDF5*

wget
*https://***EMAIL***:***EMAIL***@arthurhouhttps.pps.eosdis.nasa.gov/gpmdata/2015/04/05/1C/1C-R.GPM.GMI.XCAL2016-C.20150405-S082303-E095535.006250.V07A.HDF5*

5. Download a File Using a BASH Script

A short BASH script can be used to automate the identification and downloading of files from the PPS online archive. The same BASH command can be executed interactively from the Linux command line or executed from within a script. For example, consider the following snippet of BASH code that uses *wget*, although you will want to replace the example email address with your own PPS-registered email address with the "@" encoded as "%40".

```
# -- specify in your PPS-registered email address
user="owen.kelley%40nasa.gov"

# -- specify the PPS production or near real-time archive
URL="https://${user}:${user}@arthurhouhttps.pps.eosdis.nasa.gov/text/"

# -- tack on the path and wildcard expression for those files
URL+="gpmdata/2024/05/05/gprof/2*GMI*"

# -- obtain the list of matching files
fileList=`wget -O - -o /dev/null $URL`

# -- call wget repeatedly, once for each file in the list
for file in $fileList ; do
   URLone=`dirname $URL`/`basename $file`
   echo $URLone
   wget -q $URLone
done
```

A similar BASH snippet works if one wants to use *curl* instead of *wget*, except the that last few lines are different. The *curl* command is highlighted below.

```
# -- obtain the list of matching files
fileList=`curl $URL`

# -- call wget repeatedly, once for each file identified
for file in $fileList ; do
   URLone=`dirname $URL`/`basename $file`
   echo $URLone
   curl -Os $URLone
```

```
done
```

6. Download a File Using Python

The Python 3 programming language includes methods for sending HTTPS requests and receiving HTTPS responses. In this way, one can download files from the PPS online archive. The following Python script uses the urllib Python module to perform these tasks. If you place this code in a source file called downloadQuick.py, then you could run it in Python 3 using the following command: "python downloadQuick.py" assuming that the Python 3 executable is on your execution path.

As is customary with Python scripts, the main line is at the bottom of the source file. The main line performs these steps:

1. Connect to one of the PPS online archives using a urllib object. See the loginToArchive() function.
2. Obtain a list of files in the archive that match a user-specified pattern. See the obtainFileListFromArchive() function.
3. Using a for-loop, go through this file list, downloading one file at a time. See the downloadOneFile() function.

Before this Python 3 script will work, you must replace the dummy definition of the variable "user" in the main line with your PPS-registered email address. Unlike in prior sections of the present document, the Python 3 script below uses the your true email address, i.e., keep the "@" symbol in your email address and do not replace it with "%40". Below is the Python 3 script:

```
import os, urllib.request

def loginToArchive( ppsServer, user ):
  password_mgr = urllib.request.HTTPPasswordMgrWithDefaultRealm()
  password_mgr.add_password(None,ppsServer,user,user)
  handler = urllib.request.HTTPBasicAuthHandler(password_mgr)
  opener = urllib.request.build_opener(handler)
  urllib.request.install_opener(opener)

def obtainFileListFromArchive( ppsServer, pathAndFile ):
  req = urllib.request.Request(ppsServer+pathAndFile)
  try:
    response = urllib.request.urlopen(req)
    lines = response.read().decode().split()
    return lines
  except urllib.error.URLError as e:
    print ('Failed to find ' + pathAndFile)
    if hasattr(e, 'reason'):
      print('Reason: ', e.reason)

def downloadOneFile( ppsServer, pathAndFile ):
  path,fileNoPath = os.path.split(pathAndFile)
  localpath = './' +path
  localFile = './'+pathAndFile
```

```
  # -- make the local directory if needed
  if not os.path.exists(localpath):
    try:
      os.makedirs(localpath)
    except Exception as e:
      print( 'error: unable to create localpath: ', localpath )
      quit(errorReturnCode)

  # -- make sure you have write permission to this path
  if not os.access( localpath, os.W_OK ):
    print( 'error: unable to write to', localpath )
    quit(errorReturnCode)

  # -- download the data file
  URL = ppsServer+pathAndFile
  try:
    urllib.request.urlretrieve(URL,localFile)
  except Exception as e:
    print( 'Failed to download ' +pathAndFile )

if __name__ == '__main__':
  user = "insert.your@emailAddress.here"
  pathAndFile = \
    '/text/gpmdata/2022/01/01/imerg/3B-HHR*IMERG*S00*'
  ppsServer = 'https://arthurhouhttps.pps.eosdis.nasa.gov'
  loginToArchive(ppsServer,user)
  fileList = obtainFileListFromArchive(ppsServer,pathAndFile)
  for file in fileList:
    downloadOneFile(ppsServer,file)
```