Precipitation Processing System (PPS) FTPS access to the arthurhou server

Updated: 25 July 2023

Executive Summary

Organizations will **need to open/allow access to all ports in the range of 64000-65000** for the system 'arthurhouftps.pps.eosdis.nasa.gov' and/or 'arthurhou.pps.eosdis.nasa.gov'.

A programming language, such as Python (using the *ftplib* library), or a client program capable of FTPS interactions, such as: 'lftp', 'wget', 'curl', or 'FileZilla', must be used to access the system as normal FTP clients will be unable to do so. Each of these clients has their own peculiarities of use. Some simple examples of usage are provided in the full text below along with any known issues, version, or operating system limitations.

The PPS implementation of FTPS on servers relies on what is called Explicit FTPS, rather than what is known as Implicit FTPS, and as such the initial connection uses port 21 – just like normal FTP does – rather than using port 990 as Implicit FTPS does.

Background

NASA information security management authorities mandated that continued use of the File Transfer Protocol (FTP) should not be allowed, even when used to provide access to publicly-available, non-sensitive information. This decision was made because FTP login credentials are sent in clear text. For this reason, they mandated that all FTP sites either convert to some form of encrypted login mechanism or be shut down. PPS determined, after reviewing available options, that the fastest mechanism to meet the NASA mandate, while hopefully allowing for the least amount of code changes needed by our user base was to transition from FTP to FTPS (with HTTPS as a secondary option). This was completed in January 2021; only FTPS and HTTPS access have been operational since then. This document describes FTPS access to the PPS Production Archive called "arthurhou", while a separate document on the PPS website describes HTTPS access.

FTPS is basically FTP with the capability to provide encryption to either/both the login credentials and the data transfer. There are two implementations of FTPS known as Implicit and Explicit. The Implicit FTPS is the older, original form and is similar to how HTTPS differs from HTTP in that rather than using the existing FTP port (21) it works on a specific port (990) for connections and assumes that all traffic communicating with the port is encrypted from the very first connection. The newer method, Explicit FTPS, on the other hand uses the same port infrastructure as FTP and requires a STARTTLS command to be issued to begin the encryption channel. While it does not apply to PPS installations, this Explicit method allows for a server to support both FTP and FTPS on the same system concurrently. Current "best practices" in IT security indicate that Explicit FTPS is the preferred method of implementing FTPS, and as such, PPS has proceeded with this method in its implementation.

Encrypted Connected Lag Issues

Unfortunately, the imposition of encryption onto an FTP connection produces a noticeable lag. On interactive connections, the lag is usually only noticeable on the initial connection, while subsequent commands tend to be completed quickly. However, when operations are conducted as a series of independent connections, as is often the case with scripted retrievals, the lag applies to each connection individually. The internal tests conducted by PPS show that this lag time is usually in the 10-20 second range.

FTP(S) Data Port Issues

The existing, original PPS FTP servers have always used a limited range of ports (64000-65000) for data connections after initial connection has been made. This has never had any impact on users' ability to connect because the negotiation sequence wherein the port is promulgated from server to client have been conducted in clear text. This clear text exchange of the next port to be used for data transfers allowed firewalls conducting stateful packet inspection (SPI) to be able to determine that a subsequent connection to port N was in fact related to a previously allowed connection and thus could proceed. Unfortunately, the requirement to encrypt the login exchange means that this information can no longer be gleaned by the firewalls because it is no longer visible to them. Therefore, firewalls no longer see the initial connection and the follow-up connection to be 'related' and many, due to their own security settings, deny the data connection from proceeding, thus blocking access. In essence, this is one security system petulantly denying further access because another security system required encryption to be used, and now the first system isn't able to eavesdrop and figure out what's going on.

So, while PPS hasn't changed how the data ports work, or what sequence of ports is used, many users suddenly find themselves unable to access the server because their own firewalls are blocking the connections. ***The easiest, and most straight-forward, way to deal with this issue is for users to implement (or have their IT departments implement, depending on institutional size and requirements) a firewall rule that allows access to all ports in the range of 64000-65000 for the DNS names of: 'arthurhou.pps.eosdis.nasa.gov' and 'arthurhouftps.pps.eosdis.nasa.gov'. Doing this will ensure that connections can successfully work.***

FTPS Client Examples

The following list is not meant to be exhaustive; these are simply the client programs that PPS used to test FTPS access to its server. The client programs have extensive option sets that PPS has not explored or provided documentation about, users are encouraged to investigate these options themselves as their need and desires determine. *Be mindful that some characters, such as double-dashes ('--') may not copy/paste correctly from these examples, depending on which operating system is used. If you copy these examples, double-check that each character was copied correctly.*

Python

It is strongly recommended that Python 3 rather than Python 2.7 be used when accessing the PPS FTPS server. However, PPS has been successful using Python 2.7. The following initialization may be used under Python 3:

```
from ftplib import FTP_TLS
ftps = FTP_TLS()
ftps.connect("arthurhouftps.pps.eosdis.nasa.gov")
ftps.login('uname','password')
```

The following initialization should be done before use of the Python 2.7 retrieval:

```
from ftplib import FTP_TLS
import ssl
FTP_TLS.ssl_version = ssl.PROTOCOL_TLSv1_2
ftps = FTP_TLS()
ftps.connect("arthurhouftps.pps.eosdis.nasa.gov",21)
ftps.login('uname','password')
ftps.prot_p()
```

Once the Python FTPS session has been established with the commands above, the following commands will print a list of directory contents, change the working directory, download a half-hourly IMERG-Final V07A file, then end the FTPS session:

```
ftps.retrlines('LIST')
ftps.cwd('/gpmallversions/V07/2014/06/01/imerg/')
filename = '3B-HHR.MS.MRG.3IMERG.20140601-S000000-E002959.0000.V07A.HDF5'
with open(filename, 'wb') as localfile:
   ftps.retrbinary('RETR ' + filename, localfile.write, 1024)
ftps.quit()
```

lftp (Linux)

The 'lftp' program was tested for manual FTPS connections, although the program does provide options for executing scripted connections, no example of this type is provided.

Example:

```
lftp arthurhouftps.pps.eosdis.nasa.gov -u [user name],[password]
```

Notes: The 'lftp' program needs to be fairly recent. We found during investigations that the 'lftp' client in use on Redhat Enterprise Linux (RHEL) 6 systems (and its clones such as CentOS) did not have the required built-in security libraries necessary to interact with the FTPS server. However, RHEL 7 and RHEL 8 systems (and their clones) do not have this issue and were able to successfully connect.

curl

The 'curl' program was tested for directory listing and file download. Curl is available on Linux, macOS and Windows. However, Windows and macOS users will need to download the appropriate package. Generally, in Linux distributions, curl is part of the distribution. All examples below are meant to be a single line, but due to line-wrapping in this document, they have been broken apart across multiple lines.

Example:

Directory Listing:

```
curl -4 --ftp-ssl --user [user name]:[password] ftp://arthurhouftps.pps.eosdis.nasa.gov/
gpmallversions/V07/2014/06/01/imerg/
```

File Retrieval:

```
curl -4 --ftp-ssl --user [user name]:[password] ftp://arthurhouftps.pps.eosdis.nasa.gov/
gpmallversions/V07/2014/06/01/imerg/3B-HHR.MS.MRG.3IMERG.20140601-S000000-E002959.0000.V07A.HDF5 -o
3B-HHR.MS.MRG.3IMERG.20140601-S000000-E002959.0000.V07A.HDF5
```

Notes: The 'curl' program was the only one tested which was found to work across all three current versions of Redhat Enterprise Linux (RHEL) (and clone) systems, 6x, 7x, and 8x. Note that in the example above we have forced use of the Internet Protocol version 4 (IPV4) which may be necessary depending on how your system is provisioned. We have noticed that some systems have 'default' Internet Protocol Version 6 (IPV6) addresses set but which are not set up correctly; yet the system uses this IPV6 address regardless and thus return connection attempts are lost. The PPS FTPS server is set up to correctly handle IPV6 connections but requires that IPV6 be set up correctly on the client end as well. If you're sure that: 1) IPV6 is correctly setup on your system, or 2) IPV6 is not set up at all on your system, you may omit the '`-4`' from the examples above.

wget

The 'wget' example is provided only for file download and not for directory listing. Although directory listing would be possible, it would require the resultant output to be 'scraped' to determine what the directory structures and files available were. Wget is available for Linux, macOS, and Windows. Generally, it is included in Linux distributions but must be downloaded for Windows and macOS. Wget allows the protocol type FTPS to be listed as part of the URI, unlike curl, which specifies FTPS rather than FTP through command-line arguments. Wget also provides a simpler syntax for retrieving files listed in a list provided to the command. Note that wget may not work properly with Redhat version 6, 7, or their derivatives (e.g. CentOS).

The example below is meant to be a single line, but due to line-wrapping in this document, it has been broken apart across multiple lines.

Example:

```
wget -4 --ftp-user=[user name] --ftp-password=[password] ftps://arthurhouftps.pps.eosdis.nasa.gov/
gpmallversions/V07/2014/06/01/imerg/3B-HHR.MS.MRG.3IMERG.20140601-S000000-E002959.0000.V07A.HDF5
```

Notes: As in the examples for 'curl' above, we have forced use of the Internet Protocol version 4 (IPV4) which may be necessary depending on how your system is provisioned. We have noticed that some systems have 'default' Internet Protocol Version 6 (IPV6) addresses set but which are not set up correctly and yet the system uses this IPV6 address regardless and thus return connection attempts are lost. The PPS FTPS server is set up to correctly handle IPV6 connections but requires that IPV6 be set up correctly on the client end as well. If you're sure that 1.) IPV6 is correctly set up on your system, or b.) IPV6 is not set up at all on your system, you may omit the '`-4`' from the examples above.

FileZilla

We have successfully tested the graphical FileZilla client from Windows 10, CentOS (RHEL clone) version 7, RHEL 8 and macOS. The details of using FileZilla can be found in the FileZilla documentation and may change in various versions of FileZilla, but the basic steps are as follows. Select the File > Site Manager menu option. Click on New Site to set up a connection. Type "PPS Production Archive" or something similar in the Name field, and arthurhouftps.pps.eosdis.nasa.gov in the Host field. Leave the Port field blank. For Encryption, select "Use explicit TLS over FTP if available," and leave Logon Type as "Normal". In both the User and Password fields, enter the email address that you have previously registered with PPS. (Visit https://registration.pps.eosdis.nasa.gov/registration/ to register.) Last, click Connect.

STORM Orders

STORM provides options to receive scripts to assist in retrieving ordered data via FTPS. Three options are available, the first of which is recommended for reasons described below:

- Python script
- FTPS script
- FTPS URL

The Python script is run with the Python 3 interpreter:

```
python script.py
```

The Python script hides the details of the file transfer from the user. It attempts to connect to the PPS FTPS server but will fall back to the HTTPS server if needed. It will also perform 'sha' hash tests against downloaded data files to confirm their contents. The script will skip over files that have already been downloaded, which allows the scripts to be stopped and started again later. ***Due to these benefits, the Python script is the recommended method of obtaining data from PPS.***

The FTPS script ('ftps_get_*.txt') is meant to use used with lftp on Linux / macOS:

```
lftp -f script
```

The FTPS url file ('ftps_url_*.txt') can be directly used with newer versions (verified with 1.19 and 1.21) of wget:

```
wget -i urlfile
```

Unfortunately, curl cannot use FTPS URLs. To convert the PPS FTPS URL file to FTP URLs, run (on Linux):

```
sed -i 's/ftps:/ftp:/g' urlfile
```

On macOS, run the following:

```
sed -i '' 's/ftps:/ftp:/g' urlfile
```

The URL file can then be used with curl like:

```
xargs -n 1 curl --ftp-ssl-control -O < urlfile
```