APPLICATION NOTE

Data Logger

Data Streaming

To an FTP or HTTP Server

Revision: 07/2023

RELIABLE MONITORING SINCE 1974

Campbell SCIENTIFIC®

Table of Contents

1. Introduction

Campbell Scientific data loggers can be set up to stream data. This allows the data logger to sit behind a firewall and push its stored data, in an easy-to-read format, to a computer. No data logger software is required on the receiving computer to control the process. For our purposes, "streaming" is a way to transmit data directly from data table memory to a destination without first having to create a local file copy of the data to be transferred.

Both HTTP and FTP are standard computer network communications protocols commonly used to copy or move files between computers. We will primarily cover streaming to an FTP server using `FTPClient()`. Use `HTTPPut()` to stream to an HTTP server. Although the servers are set up differently, the streaming portions of the *CRBasic* instructions are the same.

See the *CRBasic Editor* help for detailed instruction information and program examples: https://help.campbellsci.com/crbasic/landing/Content/crbasic-help-home.htm.

For best results, use the latest operating system (OS) in your data logger.

For troubleshooting, see https://s.campbellsci.com/documents/us/technical-papers/ftp-troubleshooting.

2. Requirements

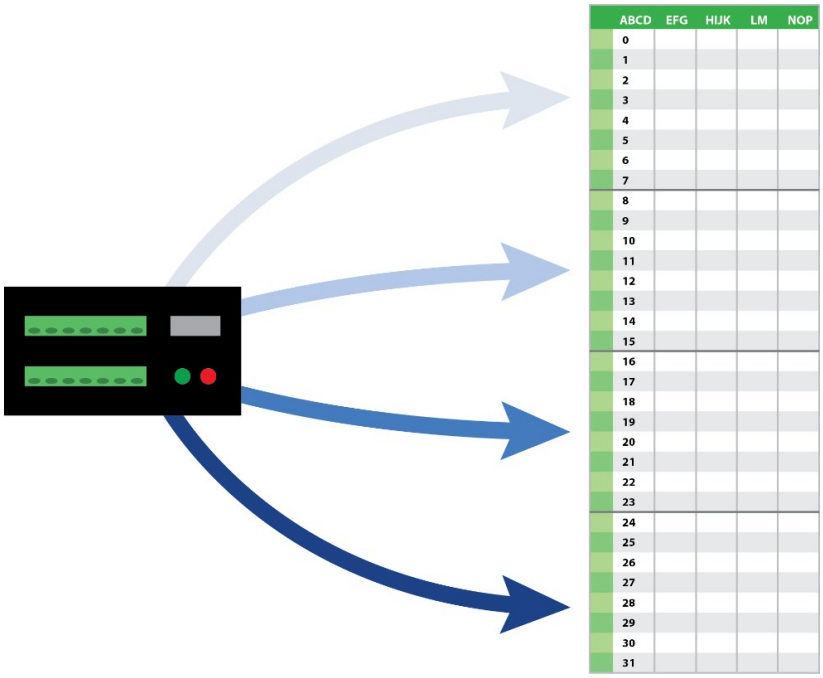
1. An IP-enabled data logger with the latest OS. For example,
    - GRANITE series, CR6, CR1000X, CR310
    - CR350 series, CR300 series with WIFI or CELL option
    - CR3000, CR1000, CR800 series with compatible cellular, Ethernet, or Wi-Fi hardware (see www.campbellsci.com/communications)
2. An IP connection
3. An HTTP or FTP server set up on a computer
    - Its IP address (for example: 192.168.123.456) or a fully qualified domain name (for example: computer-name.domain.com)
    - The user name and password

- The permissions to read and write files
- Folder(s) or directories set up to receive files. `FTPClient()` and `HTTPPut()` will not automatically create directories.

> TIP:
> Contact your network administrator for help in setting up a server.

3. Simple single file scenario

The simplest configuration writes a single data file to the server. New data will be appended to that file on a time interval. This is very similar to the way *LoggerNet* collects data.



> NOTE:
> In the example program, the tables names (highlighted) must match.

CRBasic Example 1: Single appended file

```
Public LoggerTemp, BattV
Public FTPResult

'Define Data Tables.
DataTable (FTPTest,1,-1) 'Set table size to -1 to autoallocate.
```

CRBasic Example 1: Single appended file

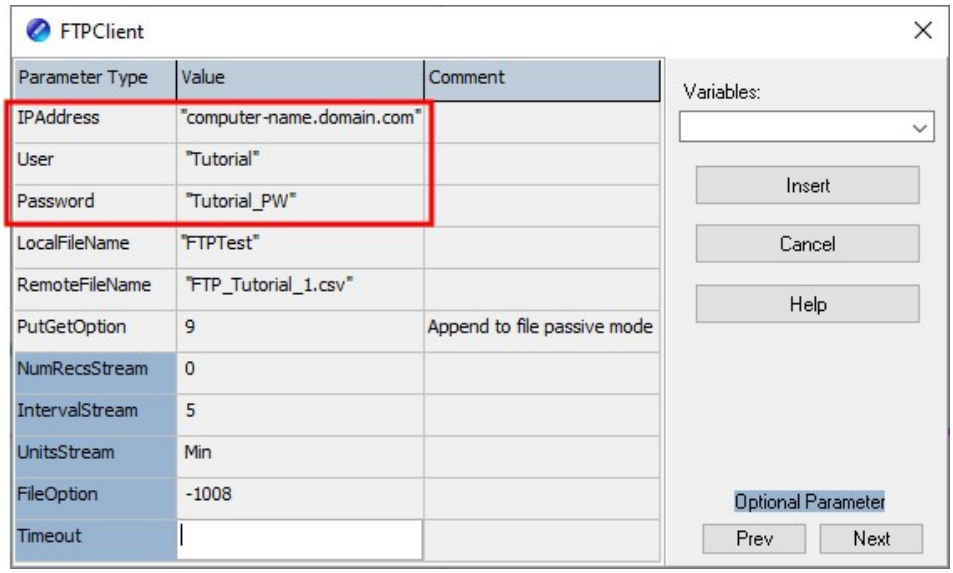
```
  DataInterval (0,15,Sec,10)
  Minimum (1,BattV,FP2,False,False)
  Sample (1,LoggerTemp,FP2)
EndTable

'Main Program
BeginProg
  Scan (1,Sec,0,0)
    PanelTemp (LoggerTemp,250)
    Battery (BattV)
    CallTable FTPTest
NextScan

SlowSequence
Do
  Delay(1,10,Sec)
  'Create file named FTP_Tutorial_1.csv and append data to the file every
  '5 minutes
  FTPResult=FTPClient ("computer-name.domain.com", "Tutorial", "Tutorial_PW",
"FTPTest", "FTP_Tutorial_1.csv", 9, 0, 5, Min, -1008)
  Loop
EndProg
```

The pertinent instruction in CRBasic Example 1 (p. 2) is **FTPClient()** configured as such:

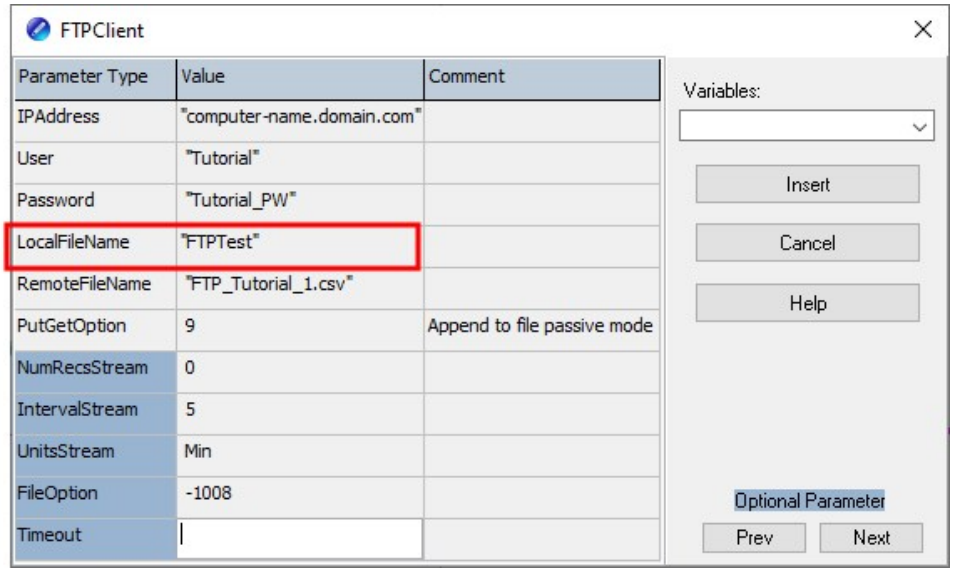
```
FTPResult=FTPClient ("computer-name.domain.com", "Tutorial", "Tutorial_PW",
"FTPTest", "FTP_Tutorial_1.csv", 9, 0, 5, Min, -1008)
```

The variable **FTPResult** will be **–1** if successful, **0** if it fails, or **–2** if execution did not occur when the instruction was called (for instance, when the time into interval conditions are not met). In this example, during normal successful operations, you will see a result code of **–2** most of the time. It will change to **–1** for a few seconds near the top of every five minutes.

The first set of parameters are based on the receiving FTP server. See requirement 1 in Requirements (p. 1). Your IP address, user name, and password will be different from the following example screenshot.



Next, in **LocalFileName** specify the name of the **DataTable** that contains the data to copy to the FTP server. CRBasic Example 1 (p. 2) shows the table name in the instruction `DataTable (FTPTest,1,-1)`. Type the Table Name in quotes, and double-check to ensure there are not any typos.

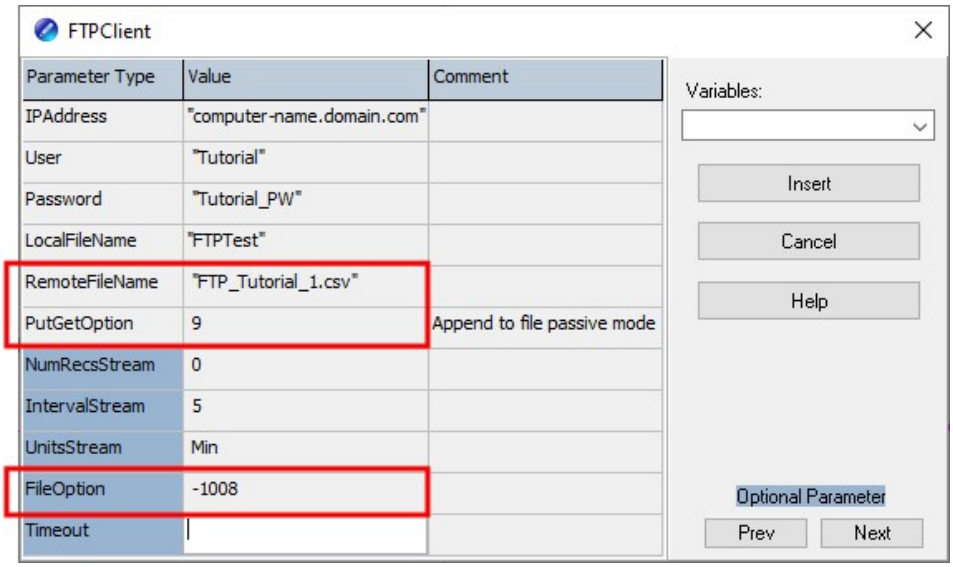


Because the objective is to have a single file contain a continuous data set, assign a static **RemoteFileName**. You will be working with a copy of this file to analyze your data; so, give it a meaningful name and useful extension. This example uses a .csv extension, but you could use .dat or .txt.

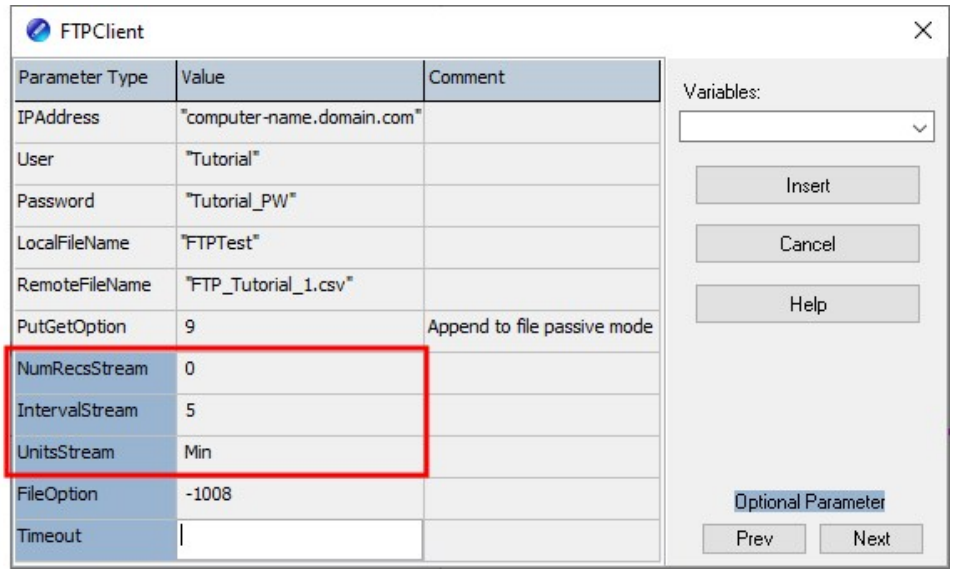
The **PutGetOption** code of **9** specifies that the data logger will be appending to a file and using what is called a passive connection. Whether a connection is active or passive depends on how

the FTP server is set up. Most data logger-to-computer FTP connections are most successful using passive mode.

FileOption specifies how the file will look on the server. The option code of **–1008**, as used in this example, generates a data file that looks like the data file collected by ***LoggerNet*** when using the default settings. Available choices include binary, ASCII, XML and JSON formats, and variations on what to include in the header. This example uses option **8**, which is a full header in ASCII. To use the file name exactly as specified in **RemoteFileName** and not append an incrementing number, add 1000 to the option number. In this example, that results in 1008. To insert the header once at the top of the file, negate this number (for example: –1008).



The next group of parameters lets you specify how often and when you want the data logger to initiate the connection to the server and stream previously unsent data. **IntervalStream** and **UnitsStream** determine how often, every 5 minutes in this example. **NumRecsStream** lets you set an offset to the interval if desired. This is commonly kept at 0 so the connection and data streaming takes place at the top of the interval. Other typical intervals would be **0,12,Hr** to stream every 12 hours at noon and midnight, or **0,1,Day** to stream once a day at midnight. To stream once a day at 8 AM, use **8,24,Hr**.

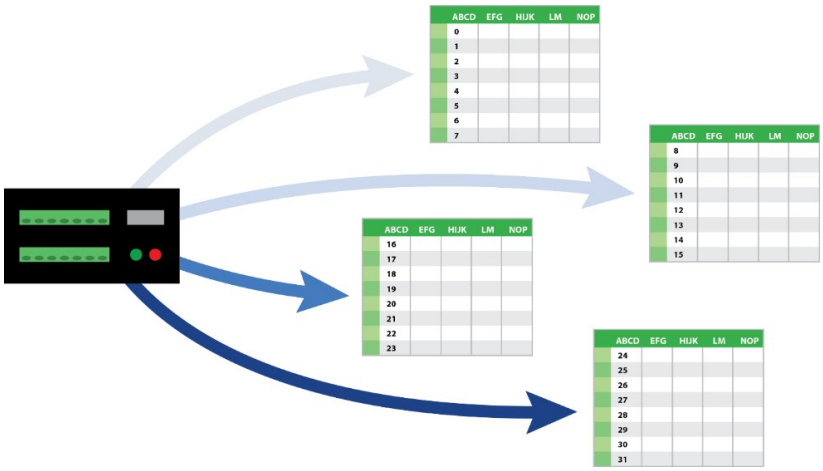


If the Ethernet connection is lost, the data logger sends the unsent data and appends when the connection comes back, similar to ***LoggerNet***.

> NOTE:
> There is no checking of table definitions. If there is a mismatch but the file name remains the same, a new header will NOT be inserted.

4. Simple multiple time-baled files scenario

Another typical scenario is to write data files containing a set amount of data and give each file a unique name. For example, every hour write a file containing an hour's worth of data. These files are sometimes referred to as bales.



NOTE:
In the example program, the tables names (highlighted) must match.

CRBasic Example 2: Multiple time-baled files

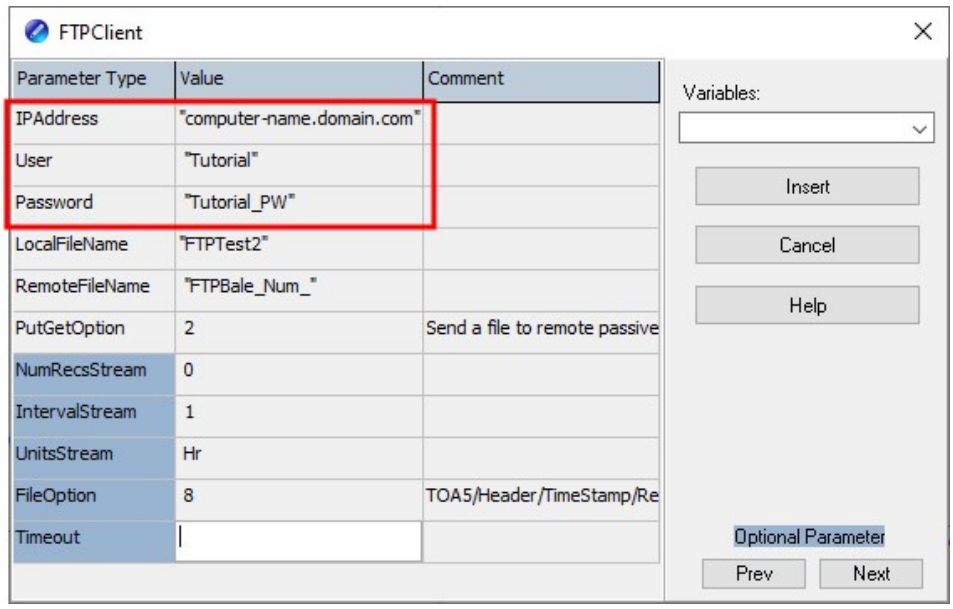
```
Public LoggerTemp, BattV
Public FTPResult

'Define Data Tables.
DataTable (FTPTest2,1,-1) 'Set table size to -1 to autoallocate.
  DataInterval (0,15,Sec,10)
  Minimum (1,BattV,FP2,False,False)
  Sample (1,LoggerTemp,FP2)
EndTable

'Main Program
BeginProg
  Scan (1,Sec,0,0)
    PanelTemp (LoggerTemp,250)
    Battery (BattV)
    CallTable FTPTest2
  NextScan

  SlowSequence
  Do
    Delay(1,10,Sec)
    'Create individual files named FTPBale_Num_0, FTPBale_Num_1, FTPBale_Num_2,
    'etc. every hour
    FTPResult=FTPClient ("computer-name.domain.com", "Tutorial", "Tutorial_PW",
"FTPTest2", "FTPBale_Num_", 2, 0, 1, Hr, 8)
  Loop
EndProg
```

The pertinent instruction in CRBasic Example 2 (p. 7) is **FTPClient()** configured as such:

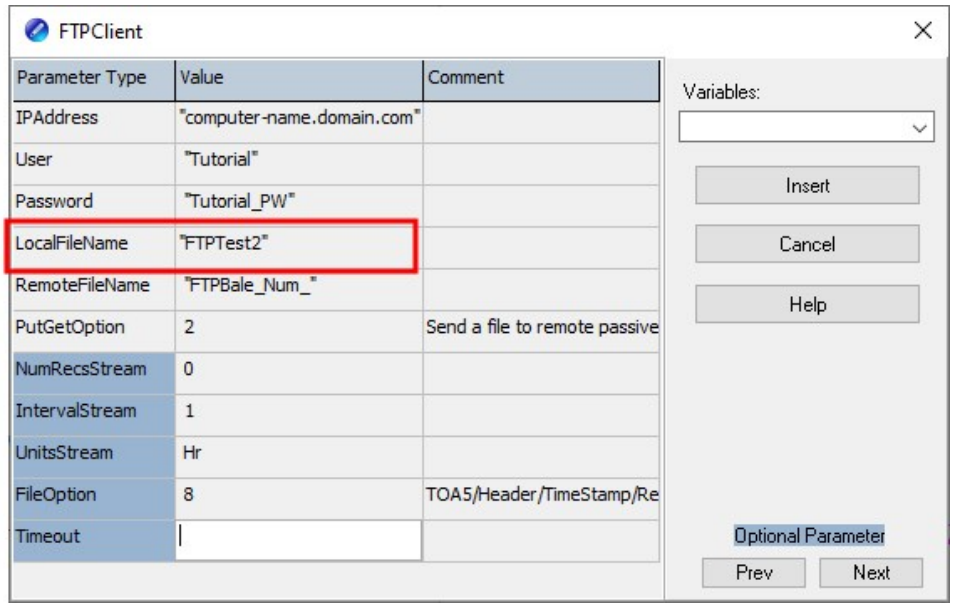
```
FTPResult=FTPClient ("computer-name.domain.com", "Tutorial", "Tutorial_PW",
"FTPTest2", "FTPBale_Num_", 2, 0, 1, Hr, 8)
```

The variable **FTPResult** will be **–1** if successful, **0** if it fails, or **–2** if execution did not occur when the instruction was called (for instance, when the time into interval conditions are not met). In this example, during normal successful operations, you will see a result code of **–2** most of the time. It will change to **–1** for a few seconds near the top of every hour.

The first three parameters are the same as were used in the Simple single file scenario (p. 2). These are based on the receiving FTP server. See requirement 1 in Requirements (p. 1). Your IP address, user name, and password will be different from the following example screenshot.



Next, in **LocalFileName** specify the name of the **DataTable** that contains the data to copy to the FTP server. CRBasic Example 2 (p. 7) shows the table name in the instruction **DataTable (FTPTest2,1,–1)**. Type the Table Name in quotes, and double-check to ensure there are not any typos.

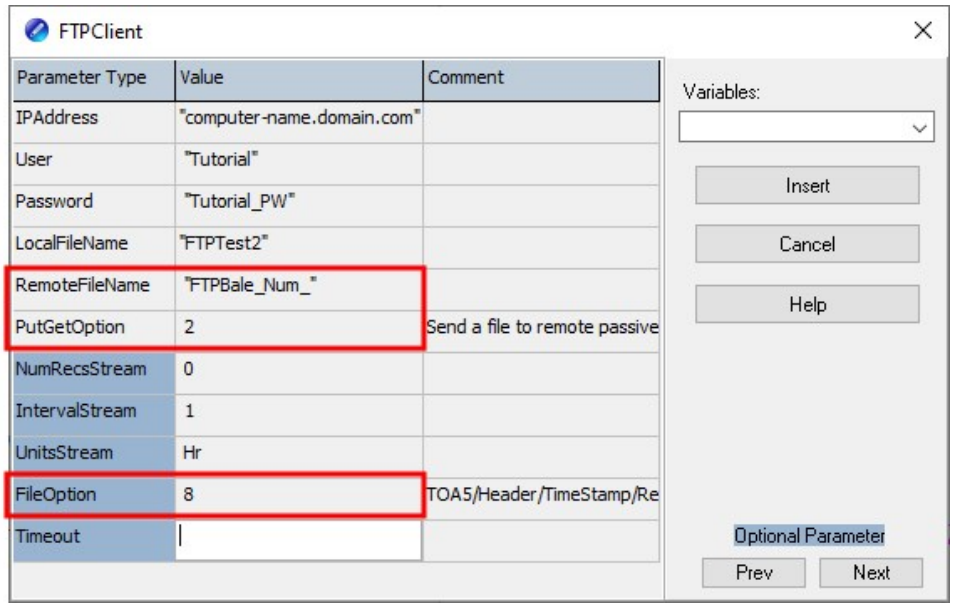


Because the objective is to have separate files containing a set amount of data, the files on the FTP server will all need unique file names. The **RemoteFileName** provides the base name for each file. **FileOption**, specifies how the file will look on the FTP server. Additionally, **FileStream** affects the name of the files. By default, the file name created on the server will automatically be appended with an incrementing file number and a ".dat" file extension.

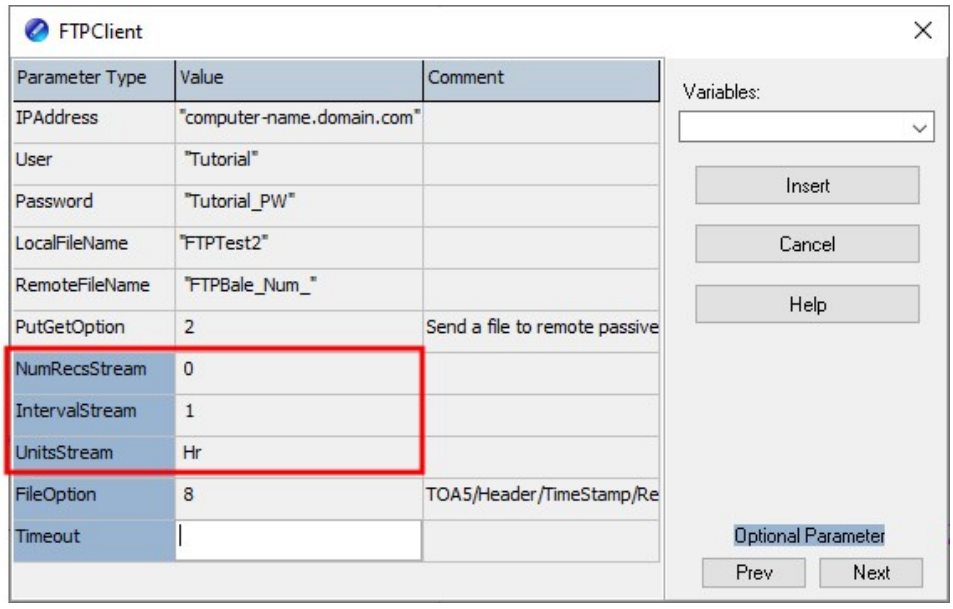
Using a **RemoteFileName** of **FTPBale_Num_** (notice the final underscore _) and an **FileOption** code of 8, as used in this example, the resulting files on the server will have names following this pattern: FTPBale_Num_0, FTPBale_Num_1, FTPBale_Num_2…

The **FileOption**code of **8**, generates a data file that looks like the data file collected by ***LoggerNet*** using the default settings, which is a full header in ASCII. Other choices include binary, ASCII, XML and JSON formats, and variations on what to include in the header.

The **PutGetOption** code of **2** specifies storing separate files on the FTP server and using what is called a passive connection. Whether a connection is active or passive depends on how the FTP server is set up. Most data logger-to-computer FTP connections are most successful using passive mode.



The next group of parameters lets you specify how often and when you want the data logger to initiate the connection to the server and stream previously unsent data to it. **IntervalStream** and **UnitsStream** determine how often, every 1 hour in this example. **NumRecsStream** lets you set an offset to the interval if desired. This is commonly kept at 0 so the connection and data streaming takes place at the top of the interval. Other typical intervals would be **0,12,Hr** to stream every 12 hours at noon and midnight, or **0,1,Day** to stream once a day at midnight. To stream once a day at 8 AM, use **8,24,Hr**.



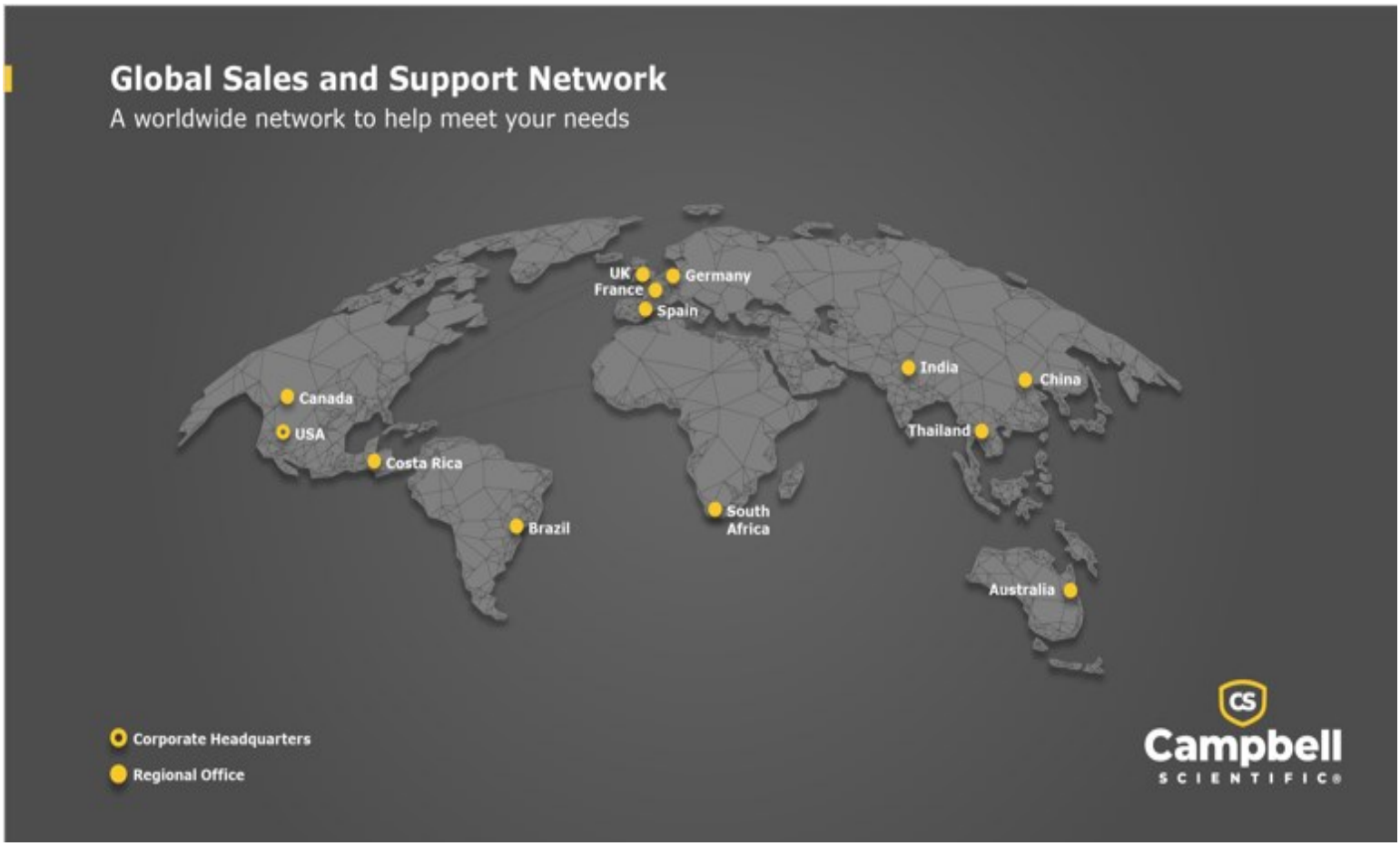
If the Ethernet connection is lost, the data logger will send the unsent data in individual files, as expected, when the connection comes back.

5. SlowSequence and Do/Delay/Loop

In both example programs, the FTPClient() instruction is in a Do/Delay/Loop in a SlowSequence.

```
SlowSequence
Do
  Delay(1,10,Sec)
  FTPResult=FTPClient ("computer-name.domain…     )
Loop
```

Instructions within a SlowSequence run at a lower priority than the main program scan. This makes it possible to run these instructions and not interrupt or delay instructions in the main scan. The Do/Delay/Loop runs at an approximate interval as specified in the Delay() instruction, every 10 seconds in this example, but not necessarily at the top of the interval.



Campbell Scientific Regional Offices

Australia

Location: Garbutt, QLD Australia
Phone: 61.7.4401.7700
Email: info@campbellsci.com.au
Website: www.campbellsci.com.au

Brazil

Location: São Paulo, SP Brazil
Phone: 11.3732.3399
Email: vendas@campbellsci.com.br
Website: www.campbellsci.com.br

Canada

Location: Edmonton, AB Canada
Phone: 780.454.2505
Email: dataloggers@campbellsci.ca
Website: www.campbellsci.ca

China

Location: Beijing, P. R. China
Phone: 86.10.6561.0080
Email: info@campbellsci.com.cn
Website: www.campbellsci.com.cn

Costa Rica

Location: San Pedro, Costa Rica
Phone: 506.2280.1564
Email: info@campbellsci.cc
Website: www.campbellsci.cc

France

Location: Montrouge, France
Phone: 0033.0.1.56.45.15.20
Email: info@campbellsci.fr
Website: www.campbellsci.fr

Germany

Location: Bremen, Germany
Phone: 49.0.421.460974.0
Email: info@campbellsci.de
Website: www.campbellsci.de

India

Location: New Delhi, DL India
Phone: 91.11.46500481.482
Email: info@campbellsci.in
Website: www.campbellsci.in

South Africa

Location: Stellenbosch, South Africa
Phone: 27.21.8809960
Email: sales@campbellsci.co.za
Website: www.campbellsci.co.za

Spain

Location: Barcelona, Spain
Phone: 34.93.2323938
Email: info@campbellsci.es
Website: www.campbellsci.es

Thailand

Location: Bangkok, Thailand
Phone: 66.2.719.3399
Email: info@campbellsci.asia
Website: www.campbellsci.asia

UK

Location: Shepshed, Loughborough, UK
Phone: 44.0.1509.601141
Email: sales@campbellsci.co.uk
Website: www.campbellsci.co.uk

USA

Location: Logan, UT USA
Phone: 435.227.9120
Email: info@campbellsci.com
Website: www.campbellsci.com