App. Note Code: 1D-AA

APPLICATION NOTE

FTP Streaming

Revision: 1/17

Table of Contents

PDF viewers: These page numbers refer to the printed version of this document. Use the PDF reader bookmarks tab for links to specific sections.

FTP Streaming

1. Introduction

Campbell Scientific dataloggers can be set up to stream data to an FTP server. This allows the datalogger to sit behind a firewall and push its stored data, in an easy-to-read format, to a computer. No datalogger software is required on the receiving computer to control the process.

FTP stands for File Transfer Protocol. It is a standard computer network communication protocol commonly used to copy or move files between computers.

"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.

In early 2013, Campbell Scientific implemented the ability to stream data from dataloggers using FTP. Refer to TABLE 1-1 for the operating systems that support FTP streaming.

TABLE 1-1. First OS to Support FTP Streaming

Datalogger	OS Version
CR6	01
CR300 Series	05
CR1000	26
CR3000	26
CR800 Series	26
CR200(X) Series	does not support

Several changes and enhancements have been made to streaming capabilities in subsequent OS releases. For best results, use the latest OS in your datalogger. (See *Sending an OS to a local datalogger video tutorial*.)

2. Requirements

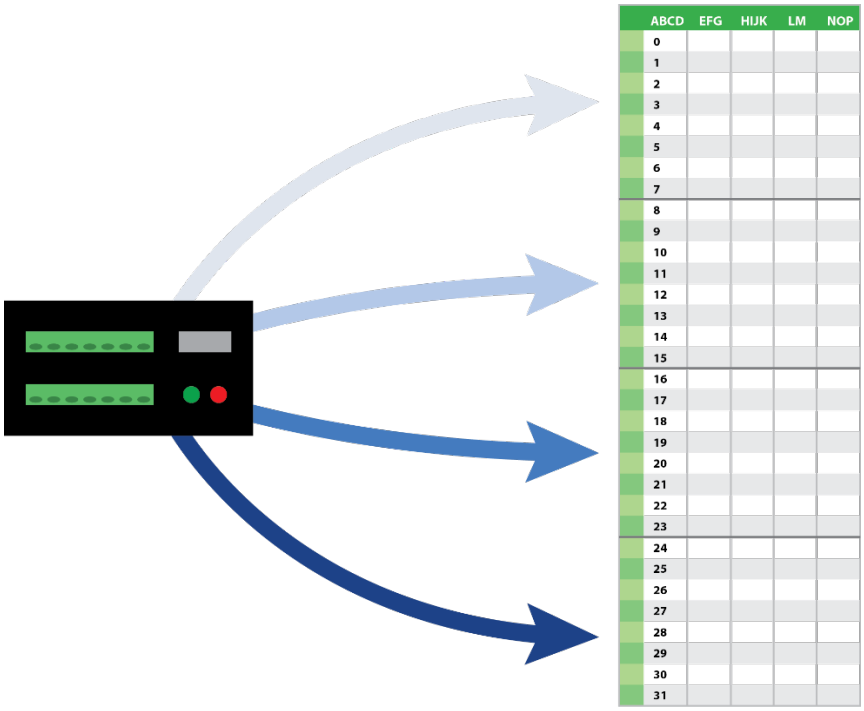
In this paper we'll cover two typical scenarios using the **FTPClient()** CRBasic instruction to stream data from a data table to an FTP server. In both scenarios you will need:

1) An FTP server set up on a computer.
   - Its IP address (i.e., 192.168.123.456) or a fully qualified domain name (i.e., computer-name.domain.com)
   - The user name and password
   - The permissions to read and write file
   - Folder(s) or directories set up to receive files. **FTPClient()** will not automatically create directories

2) An IP enabled datalogger with the latest OS. For example,
   - CR6
   - CR3000, CR800 series with compatible hardware such as an NL device (i.e., NL201, NL240, NL121, etc.) or cellular digital gateway (RavenXTV or RV50). (See *www.campbellsci.com/internet-ip-networks*.)
3) An IP connection.

3. Simple Single File Scenario

The simplest configuration will write a single data file to the FTP server. New data will be appended to that file on a time interval set in the **FTPClient()** instruction. This is very similar to the way *LoggerNet* collects data.

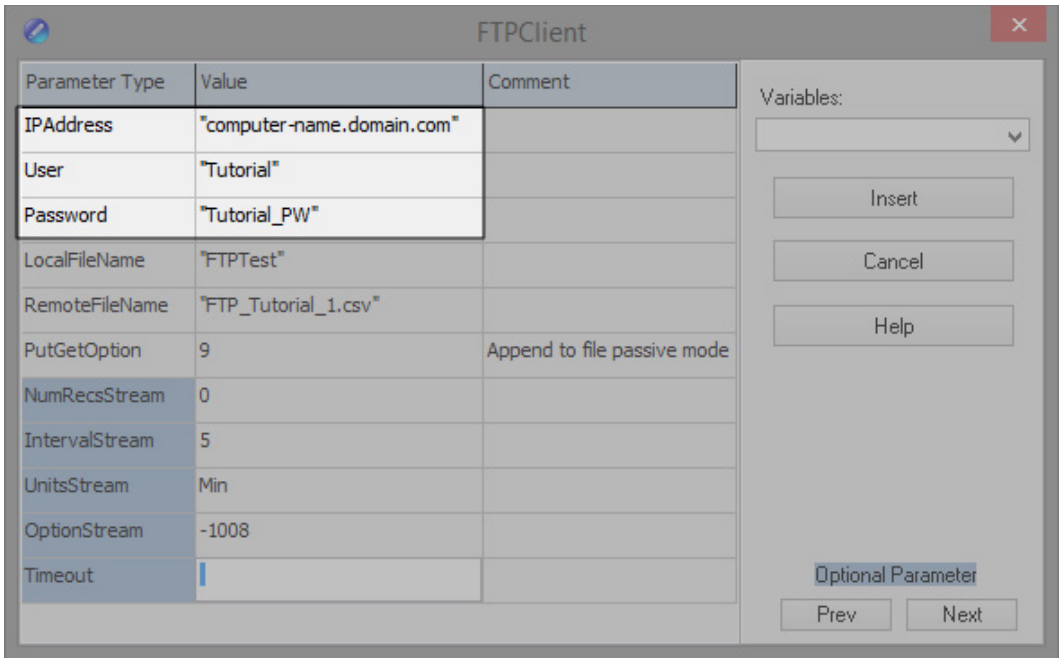


The pertinent instruction in CRBasic Example A-1, *Single Appended File* *(p. A-1)*, is **FTPClient()** configured as such:

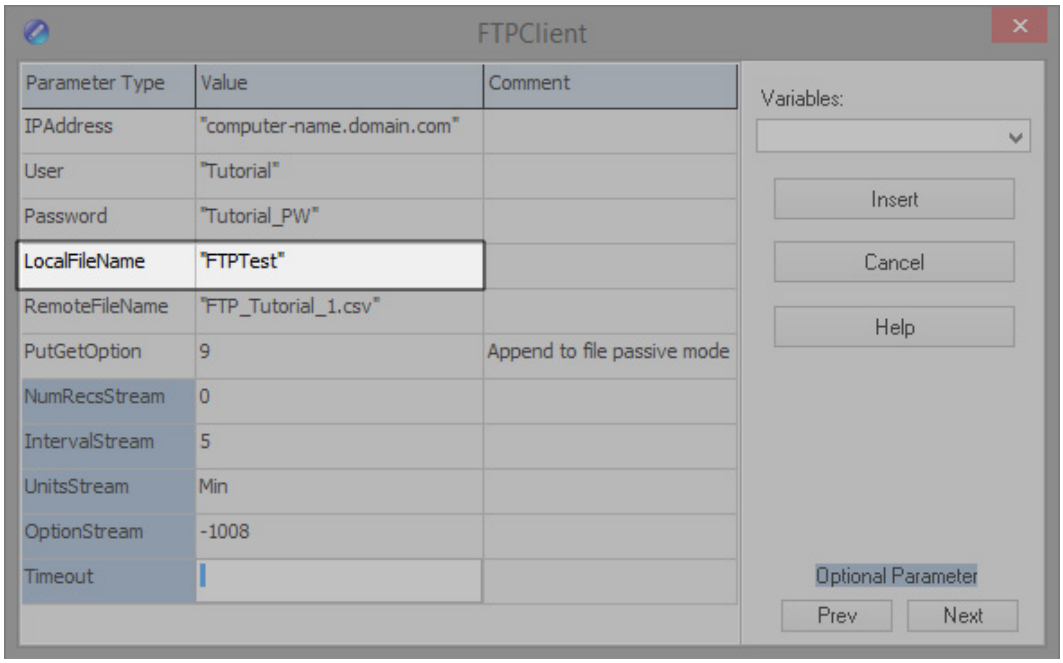
```
FTPResult=FTPClient ("computer-name.domain.com", "Tutorial",
"Tutorial_PW", "FTPTest", "FTP_Tutorial_1.csv", 9, 0, 5, Min,
-1008)
```

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 5 minutes.

The first set of parameters are based on the receiving FTP server. See requirement #1 in Section 2, *Requirements* *(p. 1)*. Your IP address, user name, and password will be different from the example screenshot shown below.



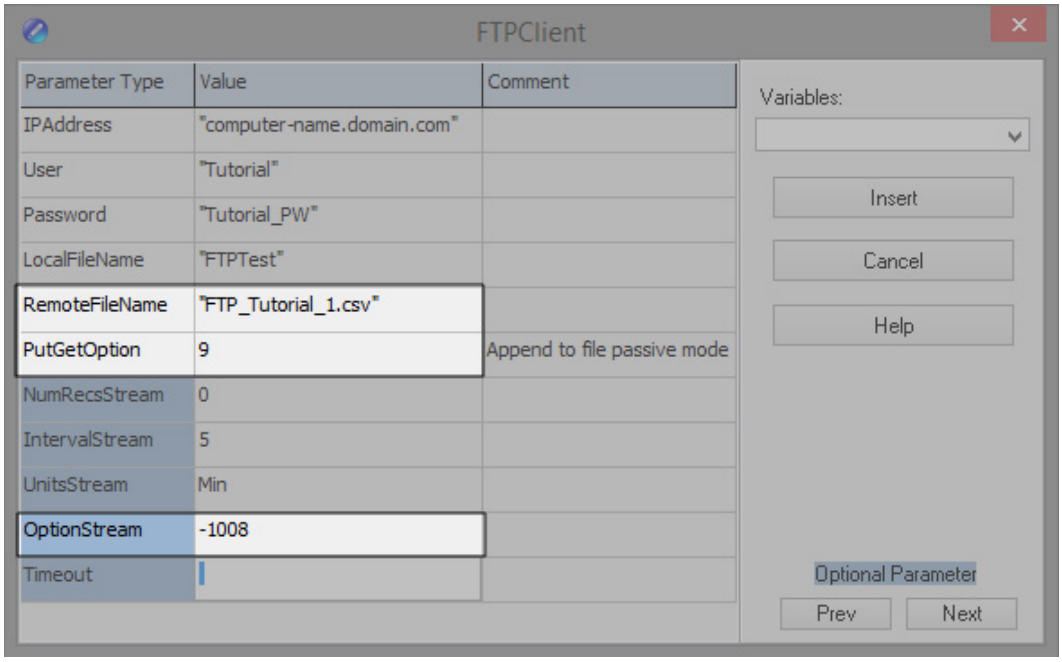
Next, in ***LocalFileName*** specify the name of the ***DataTable*** that contains the data you wish to copy to your FTP server. CRBasic Example A-1, *Single Appended File (p. A-1)*, shows the table name in the instruction **DataTable** (**FTPTest**,1,–1). Type the Table Name in quotes and double-check to make sure you don't have any typos.



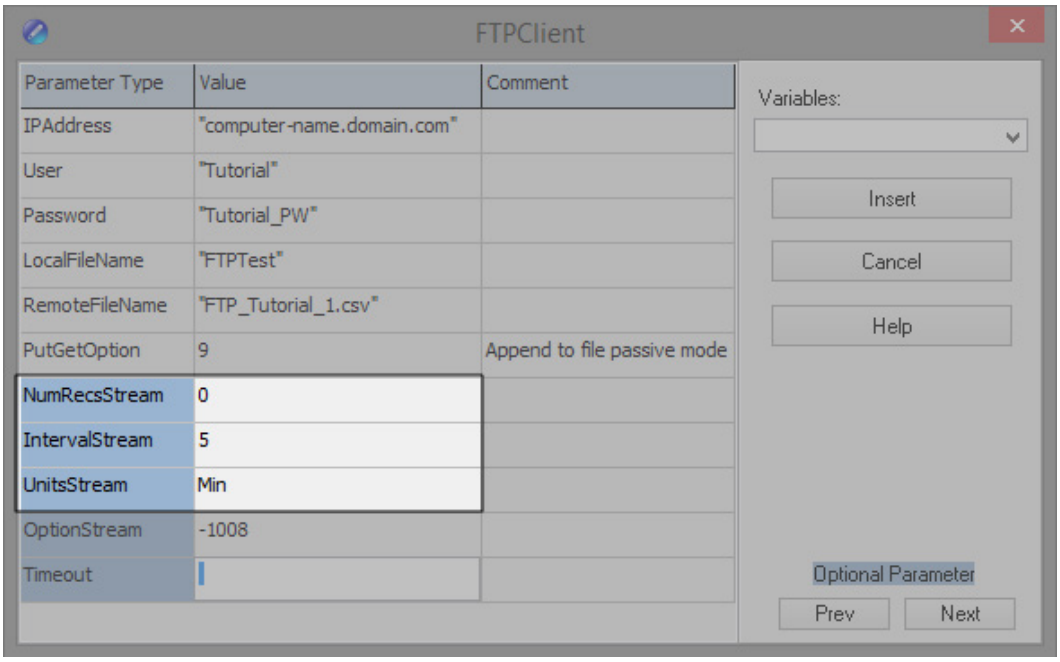
Because the objective is to have a single file contain a continuous data set, assign a static ***RemoteFileName***. You will be working with a copy of this file to analyze your data, so give it a meaningful name and useful extension. This example uses a .csv extension, but you could use .dat or .txt.

The ***PutGetOption*** code of 9 specifies that the datalogger 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 datalogger-to-computer FTP connections are most successful using passive mode.

OptionStream, also called ***FileOption***, specifies how the file will look on the FTP 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 (i.e., –1008).



The next group of parameters lets you specify how often and when you want the datalogger to initiate the connection to the FTP server and stream previously unsent data to it. ***IntervalStream*** and ***UnitsStream*** determine how often, every 5 minutes in this example. ***NumRecsStream*** lets you set an offset to the interval if desired. It 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**,**1**,**Hr** to stream hourly, or **0**,**1**,**Day** to stream once a day at midnight. To stream once a day at 8 a.m., use **8**,**24**,**Hr**.

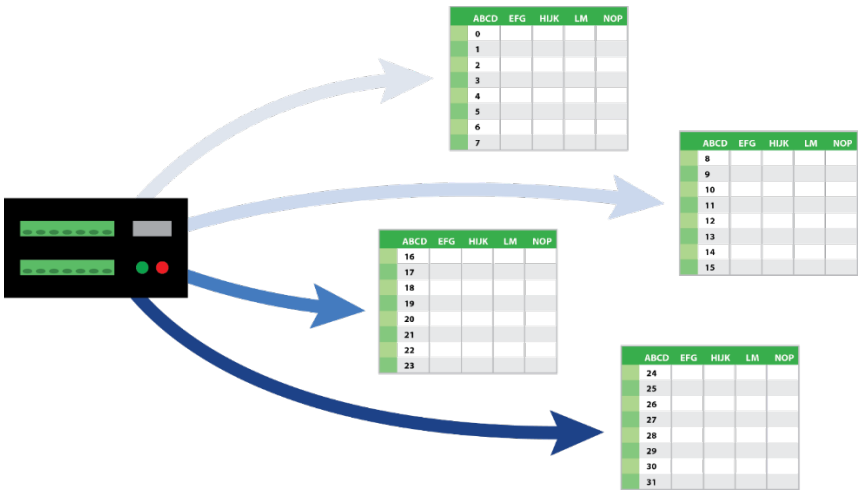


If the Ethernet connection is lost, the datalogger sends the unsent data and append when the connection comes back, similar to *LoggerNet*.

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.

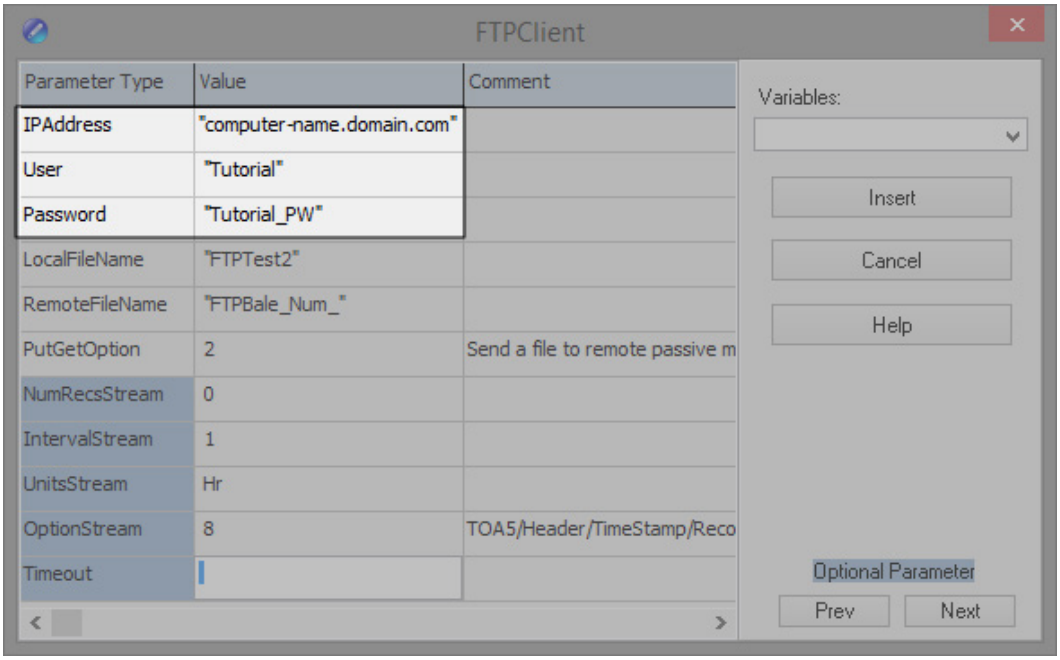


The pertinent instruction in CRBasic Example A-2, *Multiple Time-Baled Files (p. A-1)*, is **FTPClient()** configured as such:

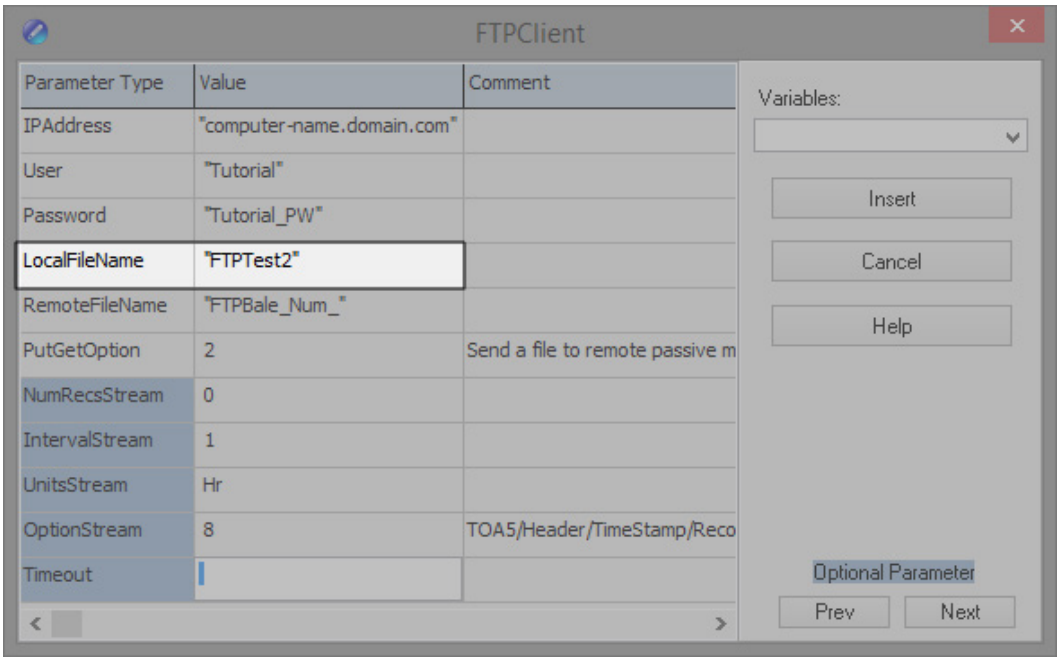
```
FTPResult=FTPClient ("computer-name.domain.com", "Tutorial",
"Tutorial_PW", "FTPTest2", "FTPBale_Num_", 2, 0, 1, Hr, 8)
```

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 timing 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 above. These are based on the receiving FTP server. See requirement #1 in Section 2, *Requirements (p. 1)*. Your IP address, user name, and password will be different from the example screenshot shown below.



Next, in ***LocalFileName***, specify the name of the ***DataTable*** that contains the data you wish to copy to your FTP server. CRBasic Example A-2, *Multiple Time-Baled Files (p. A-1)*, shows the table name in the instruction **DataTable** (**FTPTest2**,1,–1). Type the Table Name in quotes, and double-check to make sure you don't have any typos.

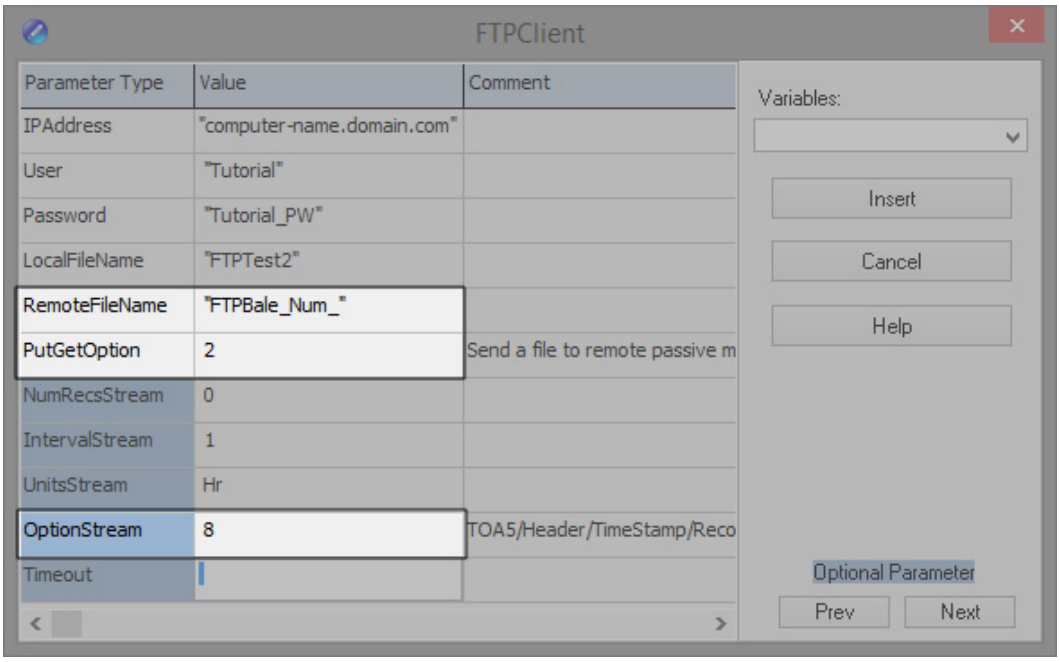


Because the objective is to have separate files containing a set amount of data, the files on the FTP server will all need unique file names. The ***RemoteFileName*** provides the base name for each file. ***OptionStream***, also called ***FileOption***, specifies how the file will look on the FTP server. Additionally, ***OptionStream*** 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.

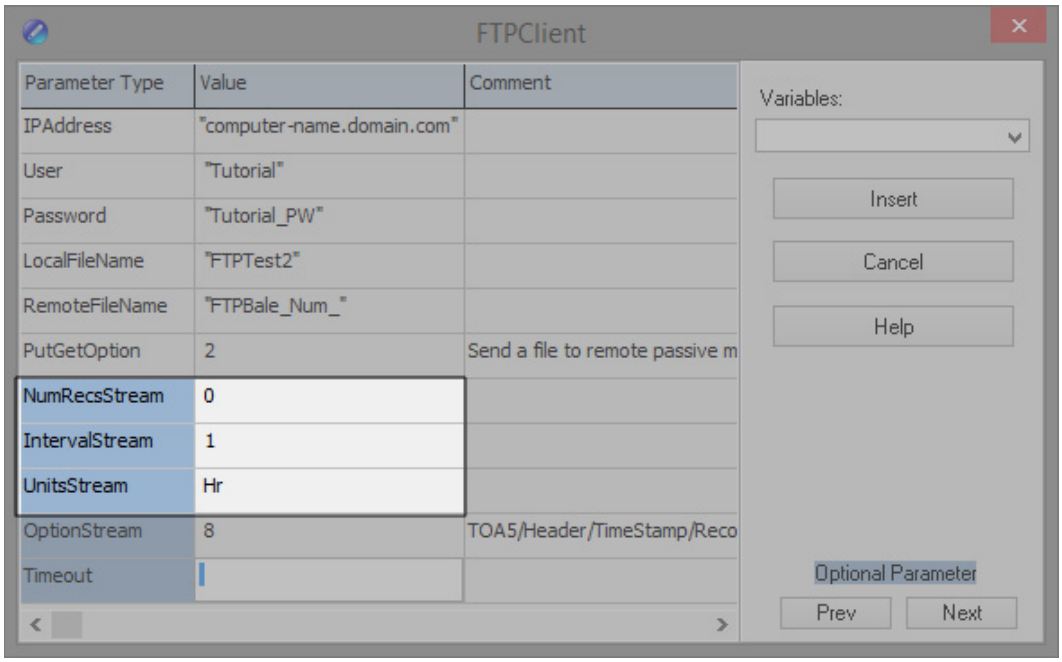
Using a ***RemoteFileName*** of ***FTPBale_Num_*** (notice the final underscore _) and an ***OptionStream*** code of 8, as used in this example, the resulting files on the FTP server will have names following this pattern: FTPBale_Num_0, FTPBale_Num_1, FTPBale_Num_2…

The ***OptionStream*** 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 datalogger-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 datalogger to initiate the connection to the FTP 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 a.m., use **8**,**24**,**Hr**.



If the Ethernet connection is lost, the datalogger 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.

Appendix A. Example Programs

NOTE In both example programs, the tables names (highlighted) must match.

CRBasic Example A-1. Single Appended File

```
Public LoggerTemp, BattV
Public FTPResult

'Define Data Tables.
DataTable (FTPTest,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 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
```

CRBasic Example A-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
```

Campbell Scientific Companies

Campbell Scientific, Inc.
815 West 1800 North
Logan, Utah 84321
UNITED STATES
www.campbellsci.com • info@campbellsci.com

Campbell Scientific Canada Corp.
14532 – 131 Avenue NW
Edmonton AB T5L 4X4
CANADA
www.campbellsci.ca • dataloggers@campbellsci.ca

Campbell Scientific Africa Pty. Ltd.
PO Box 2450
Somerset West 7129
SOUTH AFRICA
www.campbellsci.co.za • cleroux@csafrica.co.za

Campbell Scientific Centro Caribe S.A.
300 N Cementerio, Edificio Breller
Santo Domingo, Heredia 40305
COSTA RICA
www.campbellsci.cc • info@campbellsci.cc

Campbell Scientific Southeast Asia Co., Ltd.
877/22 Nirvana@Work, Rama 9 Road
Suan Luang Subdistrict, Suan Luang District
Bangkok 10250
THAILAND
www.campbellsci.asia • info@campbellsci.asia

Campbell Scientific Ltd.
Campbell Park
80 Hathern Road
Shepshed, Loughborough LE12 9GX
UNITED KINGDOM
www.campbellsci.co.uk • sales@campbellsci.co.uk

Campbell Scientific Australia Pty. Ltd.
PO Box 8108
Garbutt Post Shop QLD 4814
AUSTRALIA
www.campbellsci.com.au • info@campbellsci.com.au

Campbell Scientific Ltd.
3 Avenue de la Division Leclerc
92160 ANTONY
FRANCE
www.campbellsci.fr • info@campbellsci.fr

Campbell Scientific (Beijing) Co., Ltd.
8B16, Floor 8 Tower B, Hanwei Plaza
7 Guanghua Road
Chaoyang, Beijing 100004
P.R. CHINA
www.campbellsci.com • info@campbellsci.com.cn

Campbell Scientific Ltda.
Fahrenheitstraße 13
28359 Bremen
GERMANY
www.campbellsci.de • info@campbellsci.de

Campbell Scientific do Brasil Ltda.
Rua Apinagés, nbr. 2018 ─ Perdizes
CEP: 01258-00 ─ São Paulo ─ SP
BRASIL
www.campbellsci.com.br • vendas@campbellsci.com.br

Campbell Scientific Spain, S. L.
Avda. Pompeu Fabra 7-9, local 1
08024 Barcelona
SPAIN
www.campbellsci.es • info@campbellsci.es

Please visit www.campbellsci.com to obtain contact information for your local US or international representative.