

Use C# to upload and download files from an FTP server?

Version 1.0
October 30, 2007

By Zach Smith

Many applications require the ability to upload and download files via FTP. Even automated processes regularly interact with [FTP servers](#) to transfer data. Recognizing this, Microsoft has given developers a fairly straight forward method to implement this functionality. This document concentrates on showing you the easy way to take advantage of what Microsoft has provided in the [.NET Framework](#).

Preliminary thoughts

Before we get into moving files around I would like to bring a few things to light:

- Each request will need a NetworkCredentials object attached to its Credentials property. This tells the request how to authenticate against the FTP server.
- The URI provided to the request object will include the file name you want to upload or download. For example, if we're downloading the file "data.xml" from some.ftp.com, our URI will be ftp://some.ftp.com/data.xml.
- You need to be familiar with the Stream object. You'll use this object both when uploading and downloading files.

These may be simple and you may think "man, who wouldn't understand that?" Well, when I first started moving files to/from FTP servers I didn't understand some of these concepts so I thought they would be important! Now let's get to the code.

Download files

Downloading files is significantly easier than uploading them, so we'll start out with downloading. What we need to do is setup a WebClient object and set the Credentials property to our login information.

The next step is to call the DownloadData method of the WebClient object and supply the URI of the file we want to download. The DownloadData method returns an array of bytes which represent the downloaded file. This byte array is then written to a file, and the download is complete. The complete code for this is shown in **Figure A**.

Figure A

```
private void btnDownload_Click(object sender, EventArgs e)
{
    //Create a WebClient.
    WebClient request = new WebClient();

    //Setup our credentials
    request.Credentials =
        new NetworkCredential(this.txtUserName.Text,
                             this.txtPassword.Text);

    //Download the data into a Byte array
    byte[] fileData =
        request.DownloadData(this.txtAddress.Text + "/" +
                             this.txtFileToDownload.Text);

    //Create a FileStream that we'll write the
    // byte array to.
    FileStream file =
        File.Create(this.txtDownloadPath.Text + "\\ " +
                   this.txtFileToDownload.Text);

    //Write the full byte array to the file.
    file.Write(fileData, 0, fileData.Length);

    //Close the file so other processes can access it.
    file.Close();

    MessageBox.Show("Download complete");
}
```

Download files

Be aware that the final `file.Close()` call is crucial. Anytime you open a file in this manner you need to close it or the file will not be accessible by other processes. Closing the stream also commits the changes to disk.

Upload Files

Uploading files is significantly more complicated than downloading files. For one thing, we have to deal with two streams of data. One for the FTP connection and another for the file we're reading from disk to upload.

These are the steps we take to upload a file:

1. Create a `FtpWebRequest` object
2. Set the `FtpWebRequest.Method` property to `UploadFile`
3. Set the `FtpWebRequest.Credentials` property to our login information
4. Get the request stream of the `FtpWebRequest` object. This is the stream we'll write to.
5. Open the file we're going to upload and get a stream to its data
6. Read the file's data from the input stream and write it into the request stream
7. Close the uploaded file's stream and the request stream

The code for this is shown **Figure B**.

Figure B

```
private void btnUpload_Click(object sender, EventArgs e)
{
    //Get a FileInfo object for the file that will
    // be uploaded.
    FileInfo toUpload = new FileInfo(this.txtFile.Text);

    //Get a new FtpWebRequest object.
    FtpWebRequest request =
        (FtpWebRequest)WebRequest.Create(
            this.txtAddress.Text + "/" + toUpload.Name
        );

    //Method will be UploadFile.
    request.Method = WebRequestMethods.Ftp.UploadFile;

    //Set our credentials.
    request.Credentials =
        new NetworkCredential(this.txtUserName.Text,
                               this.txtPassword.Text);

    //Setup a stream for the request and a stream for
    // the file we'll be uploading.
    Stream ftpStream = request.GetRequestStream();
    FileStream file = File.OpenRead(this.txtFile.Text);

    //Setup variables we'll use to read the file.
    int length = 1024;
    byte[] buffer = new byte[length];
    int bytesRead = 0;

    //Write the file to the request stream.
    do
    {
        bytesRead = file.Read(buffer, 0, length);
        ftpStream.Write(buffer, 0, bytesRead);
    }
    while (bytesRead != 0);

    //Close the streams.
    file.Close();
    ftpStream.Close();

    MessageBox.Show("Upload complete");
}
```

Upload files

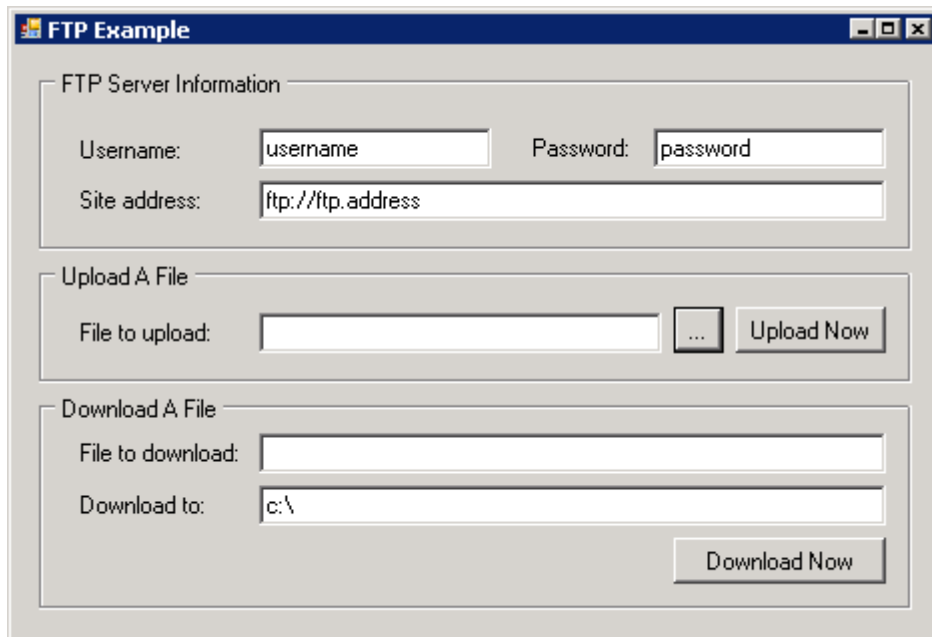
As you can see it takes over two times as many lines of code to upload than it does to download a file.

Download the sample application

This blog post is also available in PDF form as a TechRepublic download, which includes a sample Visual Studio project file exploring the coding principles outlined. The sample project includes all the code in this post, and an interface to upload/download files that may be useful in your projects.

The interface of the sample project is shown **Figure C**.

Figure C



The screenshot shows a Windows-style application window titled "FTP Example". It contains three main sections:

- FTP Server Information:** Includes input fields for "Username:" (containing "username"), "Password:" (containing "password"), and "Site address:" (containing "ftp://ftp.address").
- Upload A File:** Includes a "File to upload:" input field, a browse button "...", and an "Upload Now" button.
- Download A File:** Includes a "File to download:" input field, a "Download to:" input field (containing "c:\\"), and a "Download Now" button.

Sample project interface

Additional resources

- TechRepublic's [Downloads RSS Feed](#) **XML**
- Sign up for TechRepublic's [Downloads Weekly Update](#) newsletter
- Check out all of TechRepublic's [free newsletters](#)
- Catch up with all the [How do I](#) articles on TechRepublic.

Version history

Version: 1.0

Published: October 30, 2007

Tell us what you think

TechRepublic downloads are designed to help you get your job done as painlessly and effectively as possible. Because we're continually looking for ways to improve the usefulness of these tools, we need your feedback. Please take a minute to [drop us a line](#) and tell us how well this download worked for you and offer your suggestions for improvement.

Thanks!

—The TechRepublic Downloads Team