

# Re: System.Net.FTP and VMS

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2006-10/msg03756.html>

---

- *From:* Arne Vajhøj <arne@xxxxxxxxxx>
  - *Date:* Sat, 21 Oct 2006 21:06:37 -0400
- 

AndersGson wrote:

"Arne Vajhøj" wrote:

Mark Rae wrote:

Has anyone successfully used the FTP stuff in the System.Net namespace against a VMS FTP server?

```
FtpWebRequest objFTPRequest =  
(FtpWebRequest)WebRequest.Create("ftp:// +  
CGlobal.gstrFTPServer);  
objFTPRequest.Credentials = new  
NetworkCredential(CGlobal.gstrFTPUsername,  
CGlobal.gstrFTPPassword);  
objFTPRequest.UseBinary = false;  
objFTPRequest.UsePassive = true;  
objFTPRequest.Method =  
WebRequestMethods.Ftp.ListDirectoryDetails;  
FtpWebResponse objFTPResponse =  
(FtpWebResponse)objFTPRequest.GetResponse();  
Stream objResponseStream =  
objFTPResponse.GetResponseStream();  
StreamReader objReader = new  
StreamReader(objResponseStream);  
string strFiles = objReader.ReadToEnd();
```

This fails on the GetResponse() line with the following error:

```
{"The remote server returned an error: (550) File unavailable  
(e.g., file not found, no access)."} }
```

No what happend is:

## Re: System.Net.FTP and VMS

```
client->server: PWD
server->client: disk:[dir]
client->server: CWD disk:[dir]/
```

The .NET FTP client apparently only works with servers that uses / as directory separator (yes – Windows FTP servers usually emulates that).

I will require some coding to workaround that.

> Hi, I have the same problem. Can you remove the final "/" that FTPWebRequest  
> adds to it's CWD-command?

As I said, then it would require some coding !

:–)

I had some basic FTP code on the shelf that needed a facelift.

And it was not that difficult to create a replacement for FtpWebRequest.

See the code below.

Note that I have not really added something VMS specific. I have just not done something that I know will break with VMS.

I have tested against my VMS box running HGFTP and two public FTP servers running Multinet and DEC TCP/IP.

Arne

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;

namespace E
{
    public class FtpConnection
    {
        private TcpClient ctrl;
        private NetworkStream ctrlstm;
        private StreamWriter ctrlwrt;
        private StreamReader ctrlrd;
        private TcpClient data;
        private NetworkStream datastm;
        private void SendCmd(string cmd)
        {
```

```
ctrlwrt.WriteLine(cmd);
ctrlwrt.Flush();
}
private string ReceiveStatus()
{
    StringBuilder sb = new StringBuilder();
    string line;
    do
    {
        line = ctrlrdr.ReadLine();
        sb.Append(line + Environment.NewLine);
    }
    while((line.Length < 3) || ((line.Length != 3) && (line[3] == '-')));
    return sb.ToString();
}
public string Command(string cmd)
{
    SendCmd(cmd);
    return ReceiveStatus();
}
private void SetupData()
{
    string dataaddr = Command("PASV");
    string[] addrparts = dataaddr.Split("(").ToCharArray()[1].Split(",").ToCharArray();
    string datahost = addrparts[0] + "." + addrparts[1] + "." + addrparts[2] + "." + addrparts[3];
    int dataport = int.Parse(addrparts[4]) * 256 + int.Parse(addrparts[5]);
    data = new TcpClient(datahost, dataport);
    datastm = data.GetStream();
}
public FtpConnection(string host, string username, string password)
{
    ctrl = new TcpClient(host, 21);
    ctrlstm = ctrl.GetStream();
    ctrlwrt = new StreamWriter(ctrlstm, Encoding.Default);
    ctrlrdr = new StreamReader(ctrlstm, Encoding.Default);
    ReceiveStatus();
    Command("USER " + username);
    Command("PASS " + password);
}
public string Dir()
{
    StringBuilder sb = new StringBuilder();
    SetupData();
    Command("LIST");
    StreamReader datadr = new StreamReader(datastm);
    string line;
    while((line = datadr.ReadLine()) != null)
    {
        sb.Append(line + Environment.NewLine);
    }
    datadr.Close();
}
```

```
datastm.Close();
data.Close();
ReceiveStatus();
return sb.ToString();
}
public string ShortDir()
{
    StringBuilder sb = new StringBuilder();
    SetupData();
    Command("NLST");
    StreamReader datadr = new StreamReader(datastm);
    string line;
    while((line = datadr.ReadLine()) != null)
    {
        sb.Append(line + Environment.NewLine);
    }
    datadr.Close();
    datastm.Close();
    data.Close();
    ReceiveStatus();
    return sb.ToString();
}
public void Mkdir(string dir)
{
    Command("MKD " + dir);
}
public void Rmdir(string dir)
{
    Command("RMD " + dir);
}
public void ChDir(string dir)
{
    Command("CWD " + dir);
}
public void Upload(string filename, bool binary)
{
    if(binary)
    {
        Command("TYPE I");
        SetupData();
        Command("STOR " + filename);
        FileStream fs = new FileStream(filename, FileMode.Open, FileAccess.Read);
        byte[] b = new byte[100000];
        int n;
        while((n = fs.Read(b, 0, b.Length)) > 0)
        {
            datastm.Write(b, 0, n);
        }
        fs.Close();
        datastm.Close();
        data.Close();
    }
}
```

```
ReceiveStatus();
}
else
{
Command("TYPE A");
SetupData();
Command("STOR " + filename);
StreamReader sr = new StreamReader(filename, Encoding.Default);
StreamWriter datawrt = new StreamWriter(datastm, Encoding.Default);
string line;
while((line = sr.ReadLine()) != null)
{
datawrt.WriteLine(line);
}
sr.Close();
datawrt.Close();
datastm.Close();
data.Close();
ReceiveStatus();
}
}
public void Download(string filename, bool binary)
{
if(binary)
{
Command("TYPE I");
SetupData();
Command("RETR " + filename);
FileStream fs = new FileStream(filename, FileMode.Create, FileAccess.Write);
byte[] b = new byte[100000];
int n;
while((n = datastm.Read(b, 0, b.Length)) > 0)
{
fs.Write(b, 0, n);
}
fs.Close();
datastm.Close();
data.Close();
ReceiveStatus();
}
else
{
Command("TYPE A");
SetupData();
Command("RETR " + filename);
StreamReader datadr = new StreamReader(datastm, Encoding.Default);
StreamWriter sw = new StreamWriter(filename, false, Encoding.Default);
string line;
while((line = datadr.ReadLine()) != null)
{
sw.WriteLine(line);
}
}
}
}
```

```
}
sw.Close();
datadr.Close();
datastm.Close();
data.Close();
ReceiveStatus();
}
}
public void Logout()
{
Command("QUIT");
ctrlwrt.Close();
ctrlrdr.Close();
ctrlstm.Close();
ctrl.Close();
}
public void XFtpDir()
{
SetupData();
Command("LIST");
}
public void XFtpShortDir()
{
SetupData();
Command("NLST");
}
public void XFtpDownload(string filename, bool binary)
{
if(binary)
{
Command("TYPE I");
SetupData();
Command("RETR " + filename);
}
else
{
Command("TYPE A");
SetupData();
Command("RETR " + filename);
}
}
public Stream XFtpGetStream()
{
return datastm;
}
public void XFtpEnd()
{
ReceiveStatus();
datastm.Close();
data.Close();
}
```

```
}
public class XFtpWebRequestCreate : IWebRequestCreate
{
public WebRequest Create(Uri url)
{
return new XFtpWebRequest(url.Host, url.AbsolutePath);
}
}
public class XFtpWebRequest : WebRequest
{
private string host;
private string dir;
private string fnm;
private ICredentials cred;
private bool usebin;
private bool usepas;
private string method;
public XFtpWebRequest(string host, string path)
{
this.host = host;
if(Path.GetDirectoryName(path) != null)
{
this.dir = Path.GetDirectoryName(path).Replace(@"\", "/");
}
else
{
this.dir = path;
}
this.fnm = Path.GetFileName(path);
}
public override WebResponse GetResponse()
{
if(!usepas)
{
throw new Exception("Only passive mode supported");
}
FtpConnection con = new FtpConnection(host, ((NetworkCredential)cred).UserName,
((NetworkCredential)cred).Password);
con.ChDir(dir);
if(method == WebRequestMethods.Ftp.ListDirectoryDetails)
{
con.XFtpDir();
}
else if(method == WebRequestMethods.Ftp.ListDirectory)
{
con.XFtpShortDir();
}
else if(method == WebRequestMethods.Ftp.DownloadFile)
{
con.XFtpDownLoad(fnm, usebin);
}
}
```

```
else
{
throw new Exception("Method " + method + " is not supported");
}
return new XFtpWebResponse(con);
}
public bool UseBinary
{
get
{
return usebin;
}
set
{
usebin = value;
}
}
public bool UsePassive
{
get
{
return usepas;
}
set
{
usepas = value;
}
}
public override ICredentials Credentials
{
get
{
return cred;
}
set
{
cred = value;
}
}
public override string Method
{
get
{
return method;
}
set
{
method = value;
}
}
}
```

Re: System.Net.FTP and VMS

```
public class XFtpWebResponse : WebResponse
{
    private FtpConnection con;
    public XFtpWebResponse(FtpConnection con)
    {
        this.con = con;
    }
    public override Stream GetResponseStream()
    {
        return con.XFtpGetStream();
    }
    public override void Close()
    {
        con.XFtpEnd();
        con.Logout();
    }
}
public class MainClass
{
    private static void TestHGFTP1()
    {
        XFtpWebRequest req = (XFtpWebRequest)WebRequest.Create("xftp://arne/");
        req.Credentials = new NetworkCredential("anonymous", "arne@");
        req.UseBinary = false;
        req.UsePassive = true;
        req.Method = WebRequestMethods.Ftp.ListDirectoryDetails;
        XFtpWebResponse resp = (XFtpWebResponse)req.GetResponse();
        string dir = (new StreamReader(resp.GetResponseStream())).ReadToEnd();
        resp.Close();
        Console.WriteLine(dir);
    }
    private static void TestHGFTP2()
    {
        XFtpWebRequest req = (XFtpWebRequest)WebRequest.Create("xftp://arne/misc/");
        req.Credentials = new NetworkCredential("anonymous", "arne@");
        req.UseBinary = false;
        req.UsePassive = true;
        req.Method = WebRequestMethods.Ftp.ListDirectory;
        XFtpWebResponse resp = (XFtpWebResponse)req.GetResponse();
        string dir = (new StreamReader(resp.GetResponseStream())).ReadToEnd();
        resp.Close();
        Console.WriteLine(dir);
    }
    private static void TestHGFTP3()
    {
        XFtpWebRequest req = (XFtpWebRequest)WebRequest.Create("xftp://arne/misc/crc.c");
        req.Credentials = new NetworkCredential("anonymous", "arne@");
        req.UseBinary = false;
        req.UsePassive = true;
        req.Method = WebRequestMethods.Ftp.DownloadFile;
        XFtpWebResponse resp = (XFtpWebResponse)req.GetResponse();
    }
}
```

Re: System.Net.FTP and VMS

```
string file = (new StreamReader(resp.GetResponseStream())).ReadToEnd();  
resp.Close();  
Console.WriteLine(file);  
}  
private static void TestMultinet()  
{  
XFtpWebRequest req = (XFtpWebRequest)WebRequest.Create("xftp://ftp.tmk.com/");  
req.Credentials = new NetworkCredential("anonymous", "arne@");  
req.UseBinary = false;  
req.UsePassive = true;  
req.Method = WebRequestMethods.Ftp.ListDirectoryDetails;  
XFtpWebResponse resp = (XFtpWebResponse)req.GetResponse();  
string dir = (new StreamReader(resp.GetResponseStream())).ReadToEnd();  
resp.Close();  
Console.WriteLine(dir);  
}  
private static void TestUCX()  
{  
XFtpWebRequest req = (XFtpWebRequest)WebRequest.Create("xftp://ftp.openvms.compaq.com/");  
req.Credentials = new NetworkCredential("anonymous", "arne@");  
req.UseBinary = false;  
req.UsePassive = true;  
req.Method = WebRequestMethods.Ftp.ListDirectoryDetails;  
XFtpWebResponse resp = (XFtpWebResponse)req.GetResponse();  
string dir = (new StreamReader(resp.GetResponseStream())).ReadToEnd();  
resp.Close();  
Console.WriteLine(dir);  
}  
public static void Main(string[] args)  
{  
WebRequest.RegisterPrefix("xftp", new XFtpWebRequestCreate());  
TestHGFTP1();  
TestHGFTP2();  
TestHGFTP3();  
TestMultinet();  
TestUCX();  
Console.ReadLine();  
}  
}  
}
```