

csNetDownload - Version 1.0

ASP.NET Class for Controlling File Downloads

This is a .NET class that is used to control the download of files from a server side ASP.NET application. The *StreamFile* method will load a file from disk on the server and send it to the browser. The *StreamFromURL* and *SaveFromURL* methods will retrieve a file from a different server and send this file to disk or save this file to the server.

A free, fully functional trial version of csNetDownload is available. This trial version has a built in expiry date that causes it to stop working after that time. This is the only functional limitation between the trial and full versions. This means that you can fully test if this component is suitable for your application before considering whether to license the full version.

Using these Instructions

These instructions are divided into a number of sections with quick links to each section. A full Table of Contents is available on the next page and an index listing all commands in alphabetical order is included at the back for easy reference.

Click on one of the links below to go directly to the section of interest:

- [Getting Started](#)
- [Controlling Downloads](#)
- [Retrieving Files from a Remote Server](#)
- [Server Settings and Configuration](#)
- [Alphabetical List of Commands](#)

TABLE OF CONTENTS

1. GETTING STARTED.....	3
1.1. USING CSNETDOWNLOAD IN ASP.NET	3
1.2. THE TRIAL VERSION.....	3
2. CONTROLLING DOWNLOADS.....	5
2.1. VERIFYING COMPLETED DOWNLOADS.....	5
3. RETRIEVING FILES FROM A REMOTE SERVER.....	7
4. SERVER SETTINGS AND CONFIGURATION	8
4.1. LOCATION AND PERMISSIONS FOR THE DLL.....	8
4.2. PERMISSIONS AND IMPERSONATION.....	8
5. VISUAL DEVELOPMENT ENVIRONMENTS	9
6. OTHER PRODUCTS FROM CHESTYSOFT	10
7. ALPHABETICAL LIST OF COMMANDS.....	11

1. Getting Started

1.1. Using csNetDownload in ASP.NET

The DLL file Chestysoft.csNetDownload.dll (or Chestysoft.csNetDownloadTrial.dll for the trial version) must be placed in the \bin folder for the web application. The ASP.NET machine account must be have Read and Execute permission on the DLL file.

The permissions on the \bin folder should not allow access for the anonymous internet user, IUSR_machine_name. This is to prevent users from downloading DLLs or components.

The ASP.NET script must import the namespace csNetDownload (csNetDownloadTrial for the trial version). The class name is DownloadClass.

Creating the component instance in VB.NET:

```
<%@ Page language="vb" debug="true" %>
<%@ Import Namespace = "csNetDownload" %>
<%
.
Dim Download As New DownloadClass
.
.
%>
```

Creating the component instance in C#:

```
<%@ Page language="c#" debug="true" %>
<%@ import Namespace = "csNetDownload" %>
<%
.
DownloadClass Download = new DownloadClass();
.
.
%>
```

Both these examples show use with the full version of the class.

All the remaining examples are shown using VB.NET.

1.2. The Trial Version

The trial version of csNetDownload is supplied as a separate DLL called Chestysoft.csNetDownloadTrial.dll. This trial version is fully functional but it has an expiry date, after which time it will stop working. An exception will be raised if an attempt is made to call the *StreamFile*, *StreamFromURL* or *SaveFromURL* methods after the expiry date.

The expiry date can be found by reading the *Version* property.

Version - String, read only. This returns the version information and for the trial, the expiry date.

Example in ASP.NET (VB):

```
Dim Download As New DownloadClass
Response.Write(Download.Version)
```

Visit the Chestysoft web site for details of how to buy the full version - www.chestysoft.com

The trial version has the namespace csNetDownloadTrial, compared with csNetDownload in the full version. After upgrading to the full version the scripts using the class will need to have this namespace name changed.

2. Controlling Downloads

csNetDownload allows an ASP.NET script to control file downloads by loading the file from disk and sending it to the browser using Response.BinaryWrite. The *StreamFile* method takes the file path as a parameter and there are additional properties which add some header information.

The script that controls the download does not return HTML and there must be no HTML tags or other output in the page. It is not possible to redirect this script after the download. It is possible to run code before and after the download so form variables can be read and databases can be accessed. This allows the script to verify user data before allowing the download or to record information in a database after the download. Conditional logic could be used to redirect the user instead of downloading the file.

<p>StreamFile(<i>FileName</i>) - This method takes the file specified by the string, <i>FileName</i>, and streams it to the browser. <i>FileName</i> must be a full physical path to the file.</p> <p>Attachment - Boolean property. When a file is downloaded with <i>StreamFile</i> a header is added to indicate whether the file should be displayed inline or saved as an attachment. The header is only interpreted by Microsoft Internet Explorer. Set <i>Attachment</i> to true if the browser should prompt the user to save the file. (Default = false)</p> <p>PromptName - String property. The <i>StreamFile</i> command will add a header to specify the name of the file that is being downloaded. By default, this name will be the name of the file passed to <i>StreamFile</i> but a different name can be specified by setting the <i>PromptName</i> property. (Default = null)</p>
--

csNetDownload does not add any information about the content type or caching so these must be included in the script.

Example:

```
<%@ Page language="vb" debug="true" %>
<%@ Import Namespace = "csNetDownload" %>
<%
Response.Expires = 0
Dim Download As New DownloadClass
Response.ContentType = "application/x-zip-compressed"
Download.StreamFile("c:\files\download.zip")
%>
```

This sets the Response.Expires property to zero to prevent caching. For more advanced caching options, see the .NET Framework documentation.

This shows a zip file being downloaded so the Response.ContentType property is set to accordingly.

The *GetContentType* method can be used to find the content type. It looks for the content type in the registry given a file name or extension. If the content type for that file type is not stored in the registry it will return "application/unknown".

<p>GetContentType(<i>FileName</i>) - This method returns the content type as a string, where <i>FileName</i> is a file, or an extension of a file. If the content type for that file type is not stored in the registry it will return "application/unknown" so it is more reliable to hard code the content type where possible.</p>
--

2.1. Verifying Completed Downloads

When files are downloaded using *StreamFile* or *StreamFromURL* it is possible record whether the download was successful. This is done by using the Response.IsClientConnected command immediately after streaming the file to the browser. Here is an example:

```
<%@ Page language="vb" debug="true" %>
<%@ Import Namespace = "csNetDownload" %>
<%
Response.Expires = 0
Dim Download As New DownloadClass
Response.ContentType = "application/x-zip-compressed"
Download.StreamFile("c:\files\download.zip")
If Response.IsClientConnected = true Then
    'The download was completed
    'Do something
Else
    'The download was not completed
    'Do something else
End If
%>
```

If the connection to the client browser is still live after the download `Response.IsClientConnected` will return true, otherwise it will be false. Code can be included in the script to write to a database or a text file, but no HTML can be sent to the browser and a redirection is not possible.

This method is not completely accurate. If the download is cancelled early enough, the `If..Then..Else` statement will not be reached and neither option will be run. If the file is small, it will always show as a complete download, even if the download was cancelled. Also, this method has no way to determine if the user saved the file to disk after downloading.

3. Retrieving Files from a Remote Server

There are two methods that can retrieve a file from a remote web server. *StreamFromURL* is similar to *StreamFile*, described above, except the file is taken from a remote URL, i.e. a different server than the one running the script. *SaveFromURL* takes a file from a remote URL and saves it onto the server that is running the script.

StreamFromURL(URL) - This method streams the file at *URL* directly to the browser. *URL* is a string. The *Attachment* and *PromptName* properties are used with this method, as described in the previous section. *StreamFromURL* should not be used for large files and for files larger than 4 MB we recommend saving the file first using *SaveFromURL* and streaming the file using *StreamFile*.

SaveFromURL(URL, FileName) - This method saves the file at *URL* to disk where *FileName* is the physical path and file name on the server where the file is to be saved. *URL* and *FileName* are both strings.

Timeout - Integer property. This is the time in milliseconds that either *StreamFromURL* or *SaveFromURL* will wait for a response before raising an error. (Default = 60000, or 1 minute).

HTTPUserAgent - String property. This is the user agent value that will be sent with the request when *StreamFromURL* or *SaveFromURL* are used. It defaults to an empty string but it can be set to a value if the application needs to identify itself.

Example of streaming a file from a remote URL:

```
<%@ Page language="vb" debug="true" %>
<%@ Import Namespace = "csNetDownload" %>
<%
Response.Expires = 0
Dim Download As New DownloadClass
Download.Attachment = true
Response.ContentType = "image/gif"
Download.StreamFromURL("http://www.chestysoft.com/images/chlo.gif")
%>
```

This will download the logo from the Chestysoft web site. The *Attachment* property is set to true so Internet Explorer should prompt to open or save the file. Other browsers will probably just display the image. The exact behaviour will depend on the browser configuration.

Example of saving a file from a remote URL:

```
<%@ Page language="vb" debug="true" %>
<%@ Import Namespace = "csNetDownload" %>
<%
Response.Expires = 0
Dim Download As New DownloadClass
Download.SaveFromURL("http://www.chestysoft.com/images/chlo.gif",
"C:\images\logo.gif")
%>
```

This code will take the logo from the Chestysoft web site and save it on the server using the path and file name "C:\images\logo.gif".

4. Server Settings and Configuration

This section summarises the configuration settings that must be made on the server to run csNetDownload.

4.1. Location and Permissions for the DLL

The DLL file, Chestysoft.csNetDownload.dll (Chestysoft.csNetDownloadTrial.dll for the trial version) must be located in the binary folder for the web application. By default, this folder is called "\bin". The permission settings on the DLL must allow the ASP.NET machine account Read and Execute permission. The Internet Guest User account (IUSR_machine_name) should not be allowed to read the DLL, to protect it from unauthorised downloads.

4.2. Permissions and Impersonation

The Internet Guest User account (IUSR_machine_name) must have Read permission on any files that are to be downloaded. Similarly, this account must have Write permission on any directory where files are saved using the SaveFromURL command and Modify permission if files are to be overwritten. On systems running Windows 2003 Server, the Network_Service account must have Modify permission on files that are to be downloaded.

If an individual file does not have permission to be downloaded, the error message will be "access denied". When the entire folder does not have appropriate permissions the error message will be "file not found".

The Internet Guest User account is usually insufficient to access files on another machine across a network and sometimes it is unable to access the internet through a proxy, preventing the URL methods from working. A different account can be used by the ASP.NET application through impersonation. For full details of impersonation, refer to the .NET Framework documentation. For a simple solution a named account can be specified in the web.config file for the application, as follows.

```
<configuration>
  <system.web>
    <identity impersonate="true"
      userName="domain/username"
      password="password" />
  </system.web>
</configuration>
```

5. Visual Development Environments

When csNetDownload is used in a visual development environment such as Visual Studio or Visual Web Developer Express there are some important points to be aware of.

The script that uses csNetDownload to stream a file, either using *StreamFile* or *StreamFromURL*, must be a separate script with no HTML or other output. If the script is created by adding a new item to the project, select a new web form and then edit the source code of this form to remove all the HTML that has been added by default. Leave the `@Page` directive in the first line and then type the code that is required. Once the csNetDownload DLL has been added to the Bin directory and imported using the `Namespace` directive, the Intellisense code completion should appear during coding to assist with the syntax.

If HTML tags are left in the script they will get into the data stream and corrupt the file that is downloaded.

6. Other Products From Chestysoft

Visit the Chestysoft web site for details of other COM objects.

ActiveX Controls

[csXImage](#)

- An ActiveX control to display, edit and scan images.

[csXGraph](#)

- An ActiveX control to draw pie charts, bar charts and line graphs.

[csXThumbUpload](#)

- Upload multiple files by HTTP or FTP with previews and image edits.

[csXPostUpload](#)

- Uploads batches of files from a client to a server using an HTTP post.

[csXMultiUpload](#)

- Select and upload multiple files and post to a server using HTTP.

ASP Components

[csImageFile](#)

- Resize, edit and create images in ASP.

[csDrawGraph](#)

- Component to draw pie charts, bar charts and line graphs.

[csASPGif](#)

- Create and edit animated GIFs.

[csIniFile](#)

- Read and Edit Windows style inifiles.

[csASPUpload](#)

- Process file uploads through a browser.

[csASPZipFile](#)

- Create zip files and control binary file downloads.

[csFileDownload](#)

- Control file downloads with an ASP script.

[csFTPQuick](#)

- ASP component to transfer files using FTP.

ASP.NET

[csASPNetGraph](#)

- ASP.NET component to draw pie charts, bar charts and line graphs.

[csNetUpload](#)

- ASP.NET component for saving HTTP uploads.

Web Hosting

We can offer ASP/ASP.NET enabled web hosting with our components installed. [Click for details.](#)

7. Alphabetical List of Commands

Command	Page no.
Attachment	5
GetContentType	5
HTTPUserAgent	7
PromptName	5
SaveFromURL	7
StreamFile	5
StreamFromURL	7
Timeout	7
Version	3