csXMultiUpload - Version 1.1
ActiveX Control to Select and Upload Files Using HTTP

This ActiveX control allows the end user to select multiple files for uploading to a remote server. It uses an HTTP post operation to send the files in the same way that a browser form can upload files using the form tag <input type=file...>. The server must be running a script or cgi that is capable of saving the files.

csXMultiUpload can be embedded in a web page (Internet Explorer only) using Javascript or VBScript, or it can be used within a COM enabled programming environment.

The files to be uploaded are selected using a standard file open dialogue box and multiple selections are allowed by holding the CTRL or SHIFT keys on the keyboard. Files can also be dragged from Windows Explorer. The files are displayed in the control before uploading and more files can be added or existing files can be removed. When the Upload button is clicked each file is posted to the URL in turn, complete with any form variables. There is a separate post operation for each file.

We have another file upload control, csXThumbUpload, that can preview and edit images as well.

Using These Instructions

These instructions are divided into a number of sections with quick links to some of the sections below. 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. The PDF version also has bookmarks for direct navigation to each heading.

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

- Installation and Getting Started

- The User Interface

- Saving Uploads on a Server

- Properties and Methods

- Alphabetical List of Commands

Chestysoft, August 2010
www.chestysoft.com

TABLE OF CONTENTS

1. Installation and Getting Started

Running the installation executable will copy the files into either the default folder or a folder of your choice. It will also register the OCX and place a link to the instructions in the Start Menu. A sample HTML page is supplied and this can be reached from the Start Menu but it will not work until the *RemoteURL* property has been completed.

1.1. Using csXMultiUpload in Internet Explorer

The following code is used to create an instance of csXMultiUpload in a web page.

Full Version

```
<OBJECT id="upload" classid="CLSID:C5BD337E-B091-46B1-9D2E-AA8F5AF11929"
width="0" height="0"></OBJECT>
```

Trial Version

```
<OBJECT id="upload" classid="CLSID:26EC0426-4E0C-4469-86BF-3BAE6910FA4A"
width="0" height="0"></OBJECT>
```

The control has a minimum width and height of 377 x 477 so this will be its default size if smaller values are used.

The object tag with the class ID is sufficient to run the control on a computer where the installer has been used. To run the control on another machine it will be necessary to add a CODEBASE attribute pointing to the CAB file. A signed CAB file is provided and this will need to be moved from the folder where it was installed to a folder within your web site. The CODEBASE attribute uses a relative path.

Example:

```
<OBJECT id="upload" classid="CLSID:C5BD337E-B091-46B1-9D2E-AA8F5AF11929"
codebase="csXMultiUpload.cab" width="0" height="0"></OBJECT>
```

1.1.1. Licensing and LPK Files

csXMultiUpload is a licensed control and so a Licence Package File is required on the server. This allows the licensed copy of the control to be placed on the server but an unlimited number of users can connect to the server and use the control without requiring their own licence.

A Licence Package File is supplied with the installer and it will be in the same folder as the csXMultiUpload control. This can be copied to the server, or a .lpk file can be generated on the server as described below.

A Licence Package File can be produced using the Microsoft LPK File Generation Tool, which can be downloaded - here. Before running the tool, make sure the csXMultiUpload control has been registered on the server and the licence file (csXMultiUpload.lic or csXMultiUploadTrial.lic) is present in the same folder. Running the installer will achieve this. The LPK Tool will display a list of available controls and then prompt for a name and location of the .lpk file. Save this file into the folder in the web site where it will be used.

The .lpk file is called using the following code, which must be placed before the object tag which calls the csXMultiUpload control.

<OBJECT CLASSID="clsid:5220cb21-c88d-11cf-b347-00aa00a28331"><PARAM NAME="LPKPath" VALUE="csxmultiupload.lpk"></OBJECT>

The class ID is the same for all .lpk files. The value of the LPKPath parameter is a relative path pointing to the .lpk file and it must not start with "http://". The example above would be used unchanged if the .lpk file is in the same folder on the web server as the web page. The .lpk file supplied with the trial version of csXMultiUpload is called "csxmultiuploadtrial.lpk".

1.1.2.   Parameters

The properties for csXMultiUpload can be set using PARAM tags, which are included inside the object tag which calls the control. Here is an example using the class ID for the full version:

```
<OBJECT id="upload" classid="CLSID:C5BD337E-B091-46B1-9D2E-AA8F5AF11929"
width="0" height="0">
<PARAM name="RemoteURL" value="http://mysite.com/savefile.asp">
<PARAM name="ExtensionsAllowed" Value="*.jpg,*.bmp">
</OBJECT>
```

These tags would set the *RemoteURL* property, which is the script that will receive the uploaded files, as well as restricting the file types allowed with the *ExtensionsAllowed* property. The propeties can also be set from a Javascript function.

1.1.3.   Version

If a more recent version of the csXMultiUpload control is used in a web application it is necessary to create a new .lpk file using this version. The complete version and build number can be specified in the CODEBASE attribute and this will force the client browser to download the CAB file again if it is using an earlier version. The syntax is:

```
CODEBASE="csxmultiupload.cab#version=1,1,1,1"
```

To find the exact version number, right click on the OCX file and select "Properties". Do not right click on the .CAB file or the executable installer.

1.2.   Using csXMultiUpload in Visual Basic

The csXMultiUpload control can be imported into the Tool Box through the Project menu by selecting Components. If it has been installed with our installer it will appear in the list of available controls. Tick the box and select Apply, then Close, and the control will now appear in the Tool Box and it can be dragged onto a form.

The relevant properties for using the control do not appear in the Properties Window and must be set from code. The OnLoad event of the parent form can be a suitable place for setting properties. In the following example the csXMultiUpload control takes its default name of Upload1.

Example:

```
Private Sub Form_Load
  Upload1.RemoteURL = "http://mysite.com/savefile.asp"
End Sub
```

The *RemoteURL* property must always be set so that the control has a destination for the uploads. It is possible to have an application for uploading that contains no other code.

1.3. Other Development Environments

csXMultiUpload is an ActiveX control so it can run in a wide range of COM enabled environments. You will need to refer to the documentation for your development environment for details of how to use ActiveX controls.

1.4. The Trial Version

The trial version of csXMultiUpload is supplied as a different OCX file from the full version, called csXMultiUploadTrial.ocx, and it has a separate installer. The trial version contains a limitation where a message box pops up after each file is uploaded. This message box is removed in the full version.

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

2. Properties and Methods

2.1. Properties

RemoteURL   -   String. This property must be set before the control can be used. It is the full URL of the script that receives the uploaded files, including the "http://" prefix.

MaxFileSize   -   Integer. This sets a maximum size of file that can be downloaded, measured in bytes. When zero, the default value, there is no size limit. Files larger than *MaxFileSize* can be selected for upload but they will be rejected without an error during the upload process.

ExtensionsAllowed   -   String. It is possible to filter files by extension by setting this property. When it is an empty string, the default value, there is no file filtering. The format of the string is a comma separated list of extensions complete with the wildcard and period characters. As an example, to restrict files to .jpg and .gif extensions set *ExtensionsAllowed* to "*.jpg,*.gif". It is not case sensitive but if the string is in the incorrect format it will not be automatically corrected.

FormTagName   -   String. This is the name of the form variable containing the file. Some components and scripts that save files use this to identify the file. The default value is "csXMultiUpload".

ReturnText   -   String, read only. This contains the content returned by the remote script after uploading a file. It can be useful for debugging the remote script.

AuthenticationType   -   Integer, 0 or 1 only. When the server requires authentication this must be set to 0 for Basic Authentication and 1 for Windows Integrated (NTLM) Authentication. It must always be set to 0 when no authentication is used. (Default = 0)

Username   -   String. This is the user name that will be sent with the upload, if authentication is required.

Password   -   String. This is the password that will be sent with the upload, if authentication is required.

If authentication is used and the *Username* and *Password* properties are not set, a dialogue box will be used to collect the user name and password from the user.

FileCount   -   Integer, read only. This is the number of files currently selected in the control.

There are some additional properties which control the text for the buttons, menu items, error messages and password dialogue, but these are listed in Section 6.

2.2. Methods

The following two methods are used to manage form variables. Form variables can be included with each file upload but they must be added to the control using the *AddFormVariable* method.

AddFormVariable(*Name* As String, *Value* As String)   -   This adds a form variable which will be added to the upload post. *Name* is the form variable name and *Value* is its value. To add multiple variables, make multiple calls to the method. Once variables are set they will be used for each file upload unless cleared with *ClearVariables*.

ClearVariables ( )   -   This clears any variables already added.

Abort ( ) - This stops the upload process. This method can only be called from inside an event handler. A typical use might be to check if the user has completed any required data entry inside the *OnStartUploading* event. Calling *Abort* at this point will prevent the upload from starting but it will leave the selected files in place, so the user can be sent back to complete the task that they have missed.

GetFilePath (*Index* As Integer) - String return value. This returns the path and file name of the file at position, *Index*, in the control. Index is zero based so the first file is at Index = 0. An empty string is returned if the Index is invalid. This method is for information only, the path and file name is automatically sent as part of the upload and it can be extracted by the server side script.

3. The User Interface

The csXMultiUpload user interface is shown in the screen shot below.
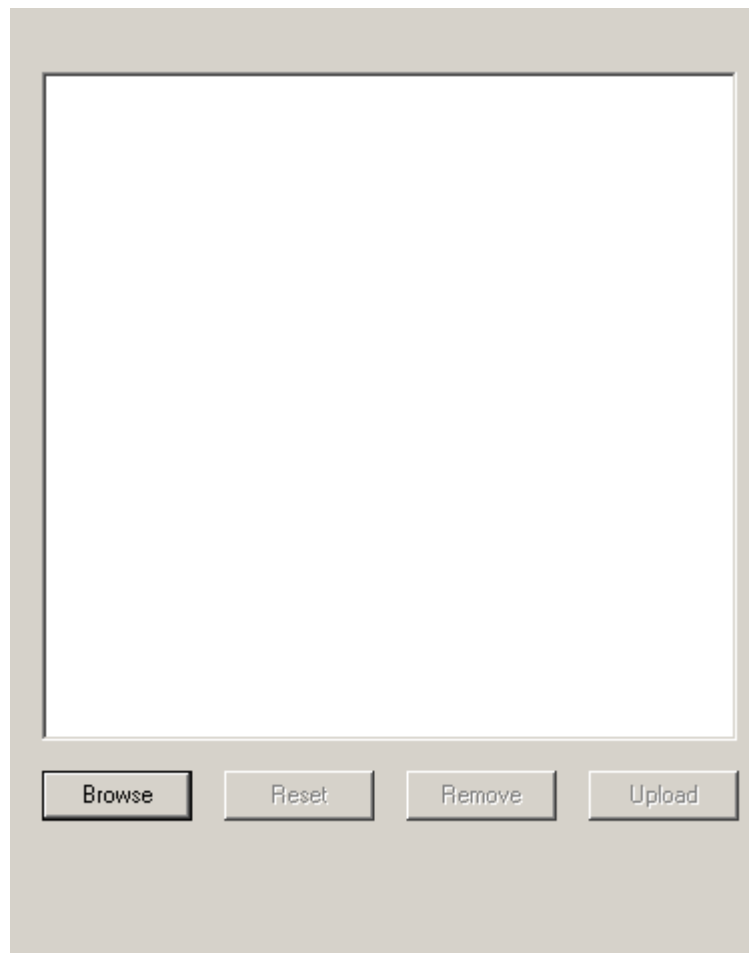
It has a minimum size of 377 x 477 pixels. If it is larger the features will spread out to fill the area.

The Browse button opens a standard file open dialogue box allowing the user to select files from their local system. Multiple selection is allowed by pressing the CTRL key while selecting the files with the mouse. Holding the shift key down allows a range of files to be selected. Files are shown in the list box (the white rectangle which is empty in the screen shot).

Files may also be added for upload by dragging and dropping from Windows Explorer. Only files can be dropped, not folders, and if the *ExtensionsAllowed* property is set, only files of an allowed type will be accepted. When used with Internet Explorer, care should be taken to drop files onto the csXMultiUpload control because if they are dropped onto other parts of the web page, the browser will attempt to open them.

The files can be cleared from the list without uploading by pressing the Reset button. Individual files can be removed by clicking on them with the mouse to highlight them and then pressing Remove. Multiple files can be highlighted this way by holding the CTRL or shift keys during selection. Holding the mouse over a file in the list will display the full file path and name as a rollover. This can be useful if the file paths are very long and too big to fit in the list box.

Files are uploaded by pressing the Upload button. During uploading some status information is displayed in the space below the buttons. This includes the number of the file being uploaded, the name of the file and a progress bar for the file. After upload, the file is removed from the list. The uploading will continue until all the files are uploaded, or the user aborts the upload, or until an error is

encountered. The trial version of csXMultiUpload will pause after each upload while it displays a reminder that a trial version is being used.

When uploading is in progress the Browse button changes to an Abort button and pressing this will stop the upload at the current file. It can be resumed again by pressing Upload.

There is also a pop up menu that can be displayed by right clicking on the control. This repeats the Browse, Reset and Remove options (described as Browse for Files, Clear All, and Remove Selected). There is also a command to Unselect, which removes the highlighting from any highlighted files.

The text that appears on the control can be changed using the properties described in a later section. This allows the control to be used with different languages.

There is also space for some text at the top of the control which is set using the *Text_Title* property. This property is an empty string by default.

4. Events

csXMultiUpload raises five events which can be used to integrate the control with your application.

OnStartUploading - Triggered when the Upload button is pressed and the upload process begins. It could be useful for adding form variables that have been collected from another part of the application.

OnStartFile(*FileNumber* As Integer, *FileTotal* As Integer) - Triggered as each file starts to upload. *FileNumber* counts which file is being uploaded starting with one for the first file in the list. *FileTotal* is the total number of files selected for upload. These numbers are reset when the Upload button is pressed so a second batch of files will start with *FileNumber* = 1.

OnEndFile(*FileNumber* As Integer, *FileTotal* As Integer) - Triggered after each successful file upload. *FileNumber* and *FileTotal* are as described above.

OnEndUploading - Triggered when the upload process is complete and all files have finished uploading.

OnAbort - Triggered when the user presses the Abort button.

The way in which events are handled varies depending on the programming language. Here is an example of using the *OnStartUploading* and OnStartFile events in Javascript to add some form variables.

```
<OBJECT id="upload" classid="CLSID:C5BD337E-B091-46B1-9D2E-AA8F5AF11929"
width="0" height="0">
<PARAM name="RemoteURL" Value="http://mysite.com/savefile.asp">
</OBJECT>
<SCRIPT LANGUAGE="JavaScript" FOR="upload" EVENT="onStartUploading()">
upload.ClearVariables();
upload.AddFormVariable("Name", "Value");
</SCRIPT>
<SCRIPT LANGUAGE="JavaScript" FOR="upload" EVENT="onStartFile(FileNumber,
FileTotal)">
if (FileNumber == FileTotal)
{
upload.AddFormVariable("LastFile", "true");
}
</SCRIPT>
```

The event handler is placed inside a <SCRIPT> tag where the FOR attribute matches the ID attribute for the object and the EVENT attribute specifies the event. The code inside the tag will execute when the upload process is started.

In this example a form variable is added when the uploading process starts, then an extra form variable is added before the last file is sent so that the server side script can detect the last file in the batch.

In addition to the events described above, csXMultiUpload supports a number of standard events, OnActivate, OnClick, OnCreate, OnDblClick, OnDestroy, OnDeactivate and OnPaint.

5. Saving Uploads on a Server

The script that receives the uploads on the remote server needs to be able to save the files. A method of doing this is described briefly here for four of the main scripting environments.

It is important to remember that each file is sent as a separate upload so if there are 20 files selected for upload the script will be called 20 times. The permissions might need to be configured to allow the script to save the files. In IIS in Windows the scripts run as the Internet Guest User and this account must have Write permission on the server. All four examples shown here are simply saving each file into the same folder as the script and keeping the name the same.

It is recommended that the server side script is tested first using a simple HTML upload form because any errors can usually be found easily and corrected. When the script is being tested it is important to use files that are the same size as those to be uploaded with csXMultiUpload because some scripting languages or server settings limit the sizes of uploaded files.

The content of the remote script is written to the *ReturnText* property after uploading and this can be read to assist in debugging. The *ReturnText* property could be read from inside the *OnEndUploading* event handler as this is called even when there is an error with the upload, and the value could be displayed in a message box such as the alert box in Javascript.

Support for server authentication has been added in version 1.1. The *AuthenticationType* property must be set to 1 if Windows Integrated Authentication (NTLM) is used, otherwise the uploads will fail. With Basic Authentication or no authentication, *AuthenticationType* must be set to 0. As this is the default value, the property can be left unused. The user name and password can be coded into the application by setting the *Username* and *Password* properties. If these properties are not set, a dialogue box will be used to collect the log in details from the user.

5.1. ASP

There is no built in method for saving an HTTP upload in ASP and a third party component is required. We sell a component called csASPUpload and so we will describe how that would be used to save a file using ASP. If using a different component refer to the documentation.

```
<%@ language=vbscript %>
<%
  Set Upload = Server.CreateObject("csASPUpload.Process")
  Upload.FileSave Upload.CurrentDir & Upload.FileName(0), 0
%>
```

The *FormTagName* property of csXMultiUpload is not needed with this component, but some other components may need the value of this property when saving the file.

5.2. ASP.NET

The HTMLInputFile.PostedFile method of saving an uploaded file in ASP.NET cannot be used with csXMultiUpload. We sell a .NET class called csNetUpload that can be used in the ASP.NET script to save uploaded files. The following code would save a file using csNetUpload.

```
<%@ Page language="vb" %>
<%@ Import NameSpace = "csNetUpload" %>
<%
Dim Upload As New UploadClass
```

```
Upload.ReadUpload
Upload.SaveFile(0, Server.MapPath("./") & Upload.Filename(0))
%>
```

The *FormTagName* property of csXMultiUpload is not needed with this component, but some other components may need the value of this property when saving the file.

5.3. Cold Fusion

In Cold Fusion the cffile tag saves uploaded files. It has attributes for renaming files to avoid name conflicts. Here is a simple script that saves the files into the same folder and uses the original names.

```
<cfset CurrentDir = ExpandPath(".")>
<cfset filename=CurrentDir & "\">
<cffile action="upload" filefield="csXMultiUpload"
destination=#filename#>
```

The *FormTagName* property of csXMultiUpload is required for the filefield attribute. The default value of "csXMultiUpload" has been used.

5.4. PHP

PHP can also save the file directly. Here is a simple script that saves into the same folder and it keeps the original file name.

```
<?php
  move_uploaded_file($_FILES['csXMultiUpload']['tmp_name'],
$_FILES['csXMultiUpload']['name'])
?>
```

The *FormTagName* property of csXMultiUpload is required to identify the file for saving. The default value of "csXMultiUpload" has been used.

6. Changing Text Labels

The labels on the buttons, the menu items and the error messages can all be set as properties, so the control can be translated into different languages. The properties will only accept ASCII characters, not Unicode.

All these properties are strings and the default values are shown in brackets.

Text_Browse	-	The text on the Browse button. (Browse)
Text_Reset	-	The text on the Reset button. (Reset)
Text_Remove	-	The text on the Remove button. (Remove)
Text_Upload	-	The text on the Upload button. (Upload)
Text_Abort	-	The text on the Abort button, which is the Browse button during uploading. (Abort)
Text_Open	-	The title of the File Open dialogue. (Select Files)
Text_UploadCount	-	The text that prefixes the file counter during upload. (Uploading)
Text_Title	-	The text written along the top of the control. This is optional and defaults to an empty string.
Text_Menu1	-	The first item on the pop up menu. (Browse for Files)
Text_Menu2	-	The second item on the pop up menu. (Clear All)
Text_Menu3	-	The third item on the pop up menu. (Unselect)
Text_Menu4	-	The fourth item on the pop up menu. (Remove Selected)
Text_Error1	-	The error message that results when the connection fails, either when no internet connection is available or the web site cannot be found. (Connection failed. Check that the remote URL is valid.)
Text_Error2	-	The error that results when the remote URL returns an HTML error code. The error code will be displayed after the string, it is not part of the string property. (There was an error with the upload. HTTP Code)

The following properties control the text on the password dialogue box used during server authentication. This box will only appear if authentication is used and if the user name and password are not coded into the application.

Text_PassHeader	-	The text along the top of the dialogue box. (Enter Network Password)
Text_Pass1	-	The request for log on details. (Please type your user name and password.)
Text_Pass2	-	The heading for the domain name. (Site:)
Text_Pass3	-	The heading for the user name. (User Name)
Text_Pass4	-	The heading for the password. (Password)
Text_PassOK	-	The text on the OK button. (OK)
Text_PassCancel	-	The text on the cancel button. (Cancel)

7. 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.

ASP Components

csImageFile - Powerful server side image manipulation component.
csDrawGraph - Draw pie charts, bar charts and line graphs in ASP.
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 - A .NET component to draw pie charts, bar charts and line graphs.
csNetUpload - ASP.NET component for saving HTTP uploads.
csNetDownload - ASP.NET class to control file downloads.

Web Hosting

We can offer ASP enabled web hosting with our components installed. Click for more details.

8. Alphabetical List of Commands

Command   Page no.