

ajmUpload V2.6

ajmUpload

**HTTP File
Upload
Manager**



User Guide and Reference Manual

December, 2004

Table of Contents

TABLE OF CONTENTS	2
I. WHAT'S NEW	4
WHAT'S NEW IN VERSION 2.6	4
WHAT'S NEW IN VERSION 2.4	5
WHAT'S NEW IN VERSION 2.2	5
1. INTRODUCTION	7
AJMUPLOAD	7
2. SYSTEM REQUIREMENTS	9
HARDWARE REQUIREMENTS	9
SOFTWARE REQUIREMENTS	9
3. PRODUCT OVERVIEW	10
AJMUPLOAD	10
4. INSTALLATION GUIDELINES	12
INSTALLATION PARAMETERS	12
INSTALLATION CONSIDERATIONS	12
5. QUICKSTART	14
INSTALLATION AND SETUP	14
6. USER GUIDE	16
FEATURES	16
OVERVIEW	16
FILE UPLOAD DETAILS	17
SECURITY CONSIDERATIONS	18
STARTING AND STOPPING AJMUPLOAD	18
CONFIGURING AJMUPLOAD	19
USING THE STATUS DISPLAY	19
USING THE SAMPLES	19
USING AJMUPLOAD WITH HTML	21
USING THE SAMPLE SCRIPTS	22
USING AJMUPLOAD WITH ASP.NET	24
USING AJMUPLOAD WITH ASP	25
USING AJMUPLOAD WITH PHP	26
USING AJMUPLOAD WITH COLF FUSION	26
USING AJMUPLOAD WITHOUT THE PROGRESS BAR	28
7. CUSTOM PROGRESS PAGE	29

OVERVIEW	29
BUILDING A CUSTOM PROGRESS PAGE	29
SAMPLE PROGRESS PAGE	32
8. REFERENCE GUIDE	33
AJMUPLOAD CONFIGURATION.....	33
BUFFER SIZE.....	33
MAX UPLOAD SIZE	33
DOMAIN.....	33
PORT.....	34
UPLOAD DIRECTORY.....	34
COMPLETION PAGE.....	34
ALLOW CUSTOM ASP.NET PROGRESS PAGES.....	35
RECEIVE TIMEOUT	35
PROGRESS WINDOW CLOSE.....	35
UPLOAD COMPLETION ACTION	35
AUTH PLUGIN	35
DIAGNOSTIC MODE.....	36
LOG RAW DATA	36
SYSTEM LOG	36
DIAG LOG.....	36
DATA LOG	36
VIEW DIAGS	37
ADMIN ID	37
PASSWORD.....	37
APPENDIX A. USING THE AUTHORIZATION PLUGIN	38
AUTHORIZATION OVERVIEW	38
WRITING THE PLUGIN	38
CREATING HTML FOR THE PLUGIN	41
DEBUGGING THE PLUGIN	42
USING THE SAMPLE PLUGINS	43
GETTING TECHNICAL SUPPORT.....	44
SUPPORT	44

i. What's New

What's New in Version 2.6

ajmUpload File Upload Manager Version 2.6 is an interim release that includes several minor program additions and support for custom progress pages.

Product Renamed

ajmUpload has been renamed to ajmUpload Professional to differentiate it from the new products in the ajmUpload line of File Upload Management Tools. These include ajmUpload.NET and ajmUpload Enterprise.

Custom Progress Pages

The upload Progress Window can be customized with your own ASP.NET template. A sample progress page is included and the standard Progress Page is, of course, still available for use.

New Samples for Cold Fusion

The Upload Completion Page samples now include an example designed specifically for Cold Fusion MX.

Improved Sample Pages

We've revamped the ASP, ASP.NET and PHP Upload Completion samples to make them easier to implement.

Trial Period Extended

We've extended the trial period from 15 to 30 days to facilitate a more thorough evaluation.

What's New in Version 2.4

ajmUpload File Upload Manager Version 2.4 is a major new release containing major new functionality and improved control over several important program features.

Authorization PlugIn

The Authorization PlugIn can be used to authenticate and/or authorize uploads before the entire file is sent to the server.

Improved Progress Window Handling

The upload Progress Window can be configured to automatically close when the upload completes or it can be rendered with a **Close** button added at the bottom of the page.

Improved Completion Page Handling

The default action at the end of an upload is to display an upload Summary Page for five seconds before transferring control the upload Completion Page. There are now options to transfer control immediately and a new option to preserve all form variables in the upload page and pass them to the Completion Page.

Improved Control over Timeouts

The new **Receive Timeout** configuration option allows more control over the amount of time ajmUpload will wait between messages before canceling the upload. The old default was fixed at four minutes.

What's New in Version 2.2

ajmUpload File Upload Manager Version 2.2 is a major new release containing a series of enhancements focused on ease of use and improved performance.

Asynchronous Communications

ajmUpload has been redesigned to use Asynchronous Communications to increase throughput and overall performance. On very busy servers, we've seen throughput improvements of up to 30%.

Performance Improvements

A careful analysis on the managed code in ajmUpload identified areas of high-processor utilization that were subsequently re-written in unmanaged assembler. This has shown dramatic improvements in CPU Utilization.

Improved Status Reporting

The Status display now includes details on any ongoing upload as well as all configuration options.

Real-time Configuration Updates

Most updates to system configuration are applied as soon as they are saved and no longer require a restart. The only notable exception to this is the Port ajmUpload listens on.

Improved Logging

ajmUpload now creates a usage log that conforms to the W3C extended format used by IIS.

1. Introduction

ajmUpload

The wide acceptance of web browsers and their ease of use make HTTP uploading a convenient way of receiving data from a wide assortment of users and machines. Unfortunately, limitations in the upload protocol and some web servers can make HTTP uploading impractical or inconvenient.

- The protocol doesn't allow for feedback to be sent during uploading, so the uploader sees nothing in his browser until the upload completes. If you're uploading large files, you have no idea how much data has been sent or even if you're still uploading.
- Most web servers will receive the entire upload before passing control to your web page. This means that you cannot perform any authentication or authorization until the entire file is uploaded – wasting time and bandwidth.
- Additionally some server implementations make uploading large files impossible through a browser. ASP.NET will receive the entire data stream into memory before attempting to write the file to disk. IIS servers running ASP.NET can easily run out of memory if they receive many files simultaneously or even a single very large file. ASP.NET will reset itself long before the large upload completes.

These issues can be resolved using specialized upload software or FTP, but many users are reluctant to install proprietary software to perform uploads because of increased administration and the security risks involved and FTP is too inconvenient for many users and provides very little control or security at the server..

ajmUpload is a File Upload Manager that's designed to address many of these issues. These are some of the product features:

- HTML-only progress bar that provides ongoing feedback to the uploader.
- Authorization Plug-In to authorize files before they're uploaded
- HTTP uploads up to 4Gb in size
- Low memory consumption

- Requires no client software other than a browser
- Unsurpassed compatibility with ASP, ASP.NET, PHP, PERL and all other server environments
- Detailed Status Display shows you how many uploads are in progress, files and bytes received, etc/

ajmUpload installs in minutes and configuration is a snap. Use the Quick Start section of the manual to get up and running quickly. Check out the samples section of the manual for a discussion of the sample upload pages provided with ajmUpload.

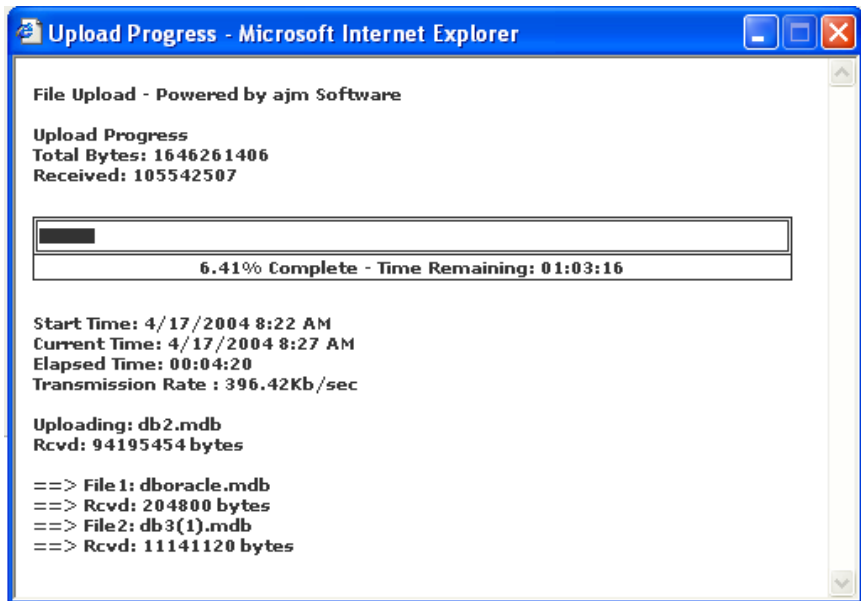


Figure 1.1. HTML only Progress Bar

2. System Requirements

Hardware Requirements

Computer/Processor

PC with a Pentium Celeron 500MHz Processor
(Pentium 4 1.7GHz or higher recommended)

Memory

128Mb of RAM
256Mb Recommended)

Hard Disk

500Kb of available hard-disk space

Network Attachment

Software Requirements

Operating System

Microsoft Windows NT 4.0 with Service Pack 6
Microsoft Windows 2000
Microsoft Windows XP
Microsoft Windows Server 2003

Software

Microsoft .NET Framework V1.1

(The Authorization Plug-In requires Visual Studio .NET or compatible IDE and compiler.)

3. Product Overview

ajmUpload

ajmUpload is a stand alone File Upload Manager that runs independently of any web servers. It uses high speed Asynchronous Communications techniques and multi-threaded techniques to achieve very high throughput with low processor utilization. All systems that utilize ajmUpload contain at least three components – a web server, the upload server and, of course, the client – as shown in Figure 3.1.

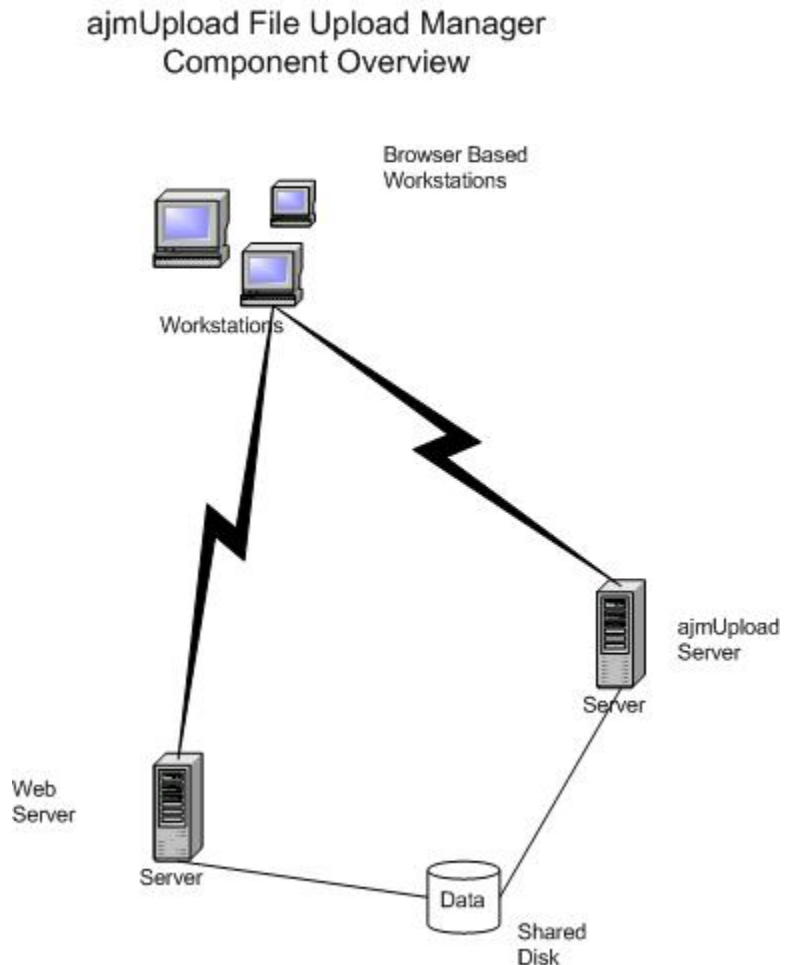


Figure 3.1 – ajmUpload Component Diagram

The Upload server and the Web Server need access to a shared disk where the uploaded files are placed. The workstations may be any platform that supports a standard browser such as Internet Explorer, Netscape, Safari, Opera, etc. The Web Server and the Upload Server may coexist on the same machine or they may be implemented on different machines. Your HTML Upload page directs uploaded files to the Upload Server. At the end of the Upload Session, when all files have been received, the Upload Server transfers control to a designated page on your web server. The Upload Server is completely independent of the Web Server and is compatible with any Web Server Platform and Technology, including ASP, ASP.NET, PHP, Perl, JSP, Cold Fusion, Unix, Linux, IIS, Apache and a host of others.

4. Installation Guidelines

Installation Parameters

ajmUpload installation is quick and easy, but you should consider and identify a few key parameters before configuring the program. The parameters you need to define are:

- The upload directory
- The domain
- The port on which ajmUpload listens

The **upload directory** should normally be accessible by both ajmUpload and your web server. This is normally not a problem unless you install ajmUpload on a different machine. If you do this and you'd like to access uploaded files from your web server, the upload directory should be a shared network drive. If there is no need to have uploaded files processed by your web server, then a shared drive is not needed.

The **domain name** is generally the same as the domain used to access your web server. If ajmUpload is installed on a separate machine, you may wish or need to use a different domain name. This domain name must be entered into the ajmUpload configuration and will be used by your web pages that facilitate file uploading.

ajmUpload will listen for HTTP requests on a **port** specified by you. Select a port that is not in use by any other programs running on the same machine. We recommend using a port over 4096. The selected port must be entered into the ajmUpload configuration and will be used by your web pages that facilitate file uploading.

Installation Considerations

You should estimate the amount of disk space needed for uploads and insure that adequate space is available. ajmUpload will not refuse upload requests when there isn't enough space to accommodate the files being uploaded. This feature is by design as the program assumes that files will be

moved out of the upload directory regularly and space will become available.

ajmUpload may create and write to a number of log files during normal operation. The location of these log files defaults to the program directory. If this presents a security risk in your environment, use the Configuration program to change the location of the logs.

The ajmUpload installation program will install ajmUpload as a service, but will not start the service. Configure the program and then start ajmUpload using the Service Control Manager. Note that rebooting your computer after installation will cause the service to start, so we recommend you configure ajmUpload immediately after installation.

5. QuickStart

Installation and Setup

ajmUpload can normally be installed and configured in less than 5 minutes. Follow these steps to get up and running quickly using the sample upload pages included with ajmUpload.

1. Select a machine and install ajmUpload.
2. Start the Configuration program from the ajm Software/ajmUpload Program Group.
3. On the options tab:
 - a. Enter the domain name of the machine that your installing ajmUpload on
 - b. Enter the port on which ajmUpload should listen for HTTP requests. If you're not sure which port to choose, start with 22401.
 - c. Enter the directory that will hold all uploaded files
 - d. Enter the URL of the web page that should be retrieved when an upload completes. Note that this field should contain a complete URL. A sample Completion page is `http://yourserver/UploadComplete.htm`.
4. Click the Save button to save these options and close the configuration program.
5. Copy files `su.js`, `upload.htm` and `UploadComplete.htm` from the Samples directory to the root directory of your web server.
6. Edit `su.js` and change the variable assignments for 'myserver' and 'myport' to the values you just entered in the configuration program. (e.g.)
 - a. `var myserver = www.myserver.com; //do not include http://`
 - b. `var myport = 22401;`
7. Start the Service Control Manager (SCM) from the Start Menu. It's usually found in the Administration Tools Group and called 'Services' (do not choose 'Component Services').

8. Find 'ajm Software File Upload Manager' and right-click on it to display a popup menu. Click 'Start' on the popup menu.

You're done!!!

6. User Guide

Features

ajmUpload features a standalone server implementation that runs as a Windows service for maximum ease of use. It can be installed on the same machine as your IIS server or on a separate machine to distribute your workload. It is compatible with any HTTP server and server scripting environment. It works with ASP, ASP.NET, PHP, PERL, etc. ajmUpload requires no client side components other than a browser and allows uploads up to 4Gb. It incorporates asynchronous multi-threaded technology and optimizes use of memory to make uploading as efficient as possible. The Authorization PlugIn provides upload authorization capabilities before the file is uploaded. The HTML based progress bar ensures maximum compatibility with today's browsers. Installation is a snap and configuration takes only a few moments. The status program gives you a detailed snapshot of ajmUpload at any time.

Overview

ajmUpload is a standalone file upload server with no dependencies on any existing web servers. It handles incoming HTTP file upload requests based on the Form-based File Upload in HTML defined in RFC 1867.

ajmUpload manages uploads using an **upload token** and an **upload summary file** created at the end of an each **upload session**. An upload session is a single request (single web page) from a browser to upload one or more files. You can upload any number of files from a single web page, but the total length of the files cannot exceed 4Gb. The actual length is slightly less than that due to HTTP header overhead. The upload token is a unique identifier associated with an upload session. It's used by the Progress Bar to retrieve status information from ajmUpload. The upload summary file is a text file that provides details about the files uploaded via ajmUpload. Its' primary use is for determining final disposition of the uploaded files.

File Upload Details

The basic steps involved in each file upload are:

1. Request upload page
2. Identify files to upload
3. POST form to ajmUpload
4. Open Progress Window
5. Upload data
6. Report Progress
7. Store Uploaded Files
8. Write Upload Summary file
9. Send summary page
10. Transfer to Upload Completion page

We'll walk through each step in detail. The sample web pages included with ajmUpload demonstrate all of the features discussed here and you can refer to them as needed.

1. Request upload page
The uploader starts his browser and requests the upload web page from your server. This web page contains a standard form with one or more `<input type="file">` elements defined on it.
2. Identify file(s) to upload
The uploader can identify each file to upload using the standard form elements (`<input type="file">`).
3. POST form to ajmUpload
The uploader clicks on the Submit button to start the upload. The Submit button will execute a Javascript function that opens a progress window and posts the web page to the ajmUpload server.
4. Open Progress Window
The Progress window requests progress from ajmUpload. The returned page contains a progress bar and upload details including bytes uploaded so far, files uploaded, time remaining to complete the upload, etc.
5. Upload data
The original browser window uploads the files to the ajmUpload server.
6. Report Progress
The Progress Window will continue to request progress from the ajmUpload server at 2 second increments. The uploader may elect to close this window as needed.

7. Store Uploaded Files
The ajmUpload server saves the uploaded files to disk in the Upload directory specified during configuration.
8. Write Upload Summary file
At the end of the upload, ajmUpload writes the upload summary file into the upload directory.
9. Send summary page
ajmUpload sends a summary page to the uploader as well as a final progress report.
10. Transfer to Upload Completion page
The summary page will transfer control to your Upload Completion page after a few seconds. The token is passed to the Upload Completion page as part of the URL and can be used to retrieve the upload summary file. The data in the upload summary file is used to move the uploaded files to their next staging area and dispose of the upload summary file.

At this point the upload is completed.

Security Considerations

ajmUpload runs as a service using the Local System account. You may change this setting after installation using the Service Control Manager (SCM). The permissions needed by the program are:

- read/write access to the upload directory
- read/write access to the log files
- read access to the configuration file located in the program directory

If any account changes are made, please insure that the ajmUpload service has appropriate access to these locations.

Starting and Stopping ajmUpload

ajmUpload runs in the background as a Windows Service. It is configured to start automatically whenever the server machine is restarted. You may use the Service Control Manager to manually start and stop ajmUpload as well. If you stop the service manually, all uploads in progress are terminated gracefully and ajmUpload attempts to notify all users that a shutdown is in progress. You may also pause ajmUpload. This allows all ongoing uploads to continue but any new upload

requests are not honored. Resuming the service will enable new uploads to commence.

Configuring ajmUpload

Use the Configuration program to maintain ajmUpload runtime parameters or to register your copy of ajmUpload. You can start the configuration program through the MS Windows Start Menu under the ajm Software/ajmUpload Program Group. You must enter a Domain, Port, Upload Directory and Completion Page. We strongly recommend that you also enter an Admin ID and a password as well. The ID and password are used when displaying the ajmUpload Status Page. See the Reference Guide found later in this manual for specific details on each field.

The runtime parameters are stored in an XML file located in the Program directory. This file cannot be moved and should not be updated through an editor. **Always use the Configuration program to change runtime parameters.**

Most changes made through the configuration program will take effect as soon as you save them. The only exception is the Port that ajmUpload listens on. Port changes will not take effect until the machine is rebooted or ajmUpload is restarted through the SCM.

Using the Status Display

ajmUpload provides a status display that shows you key statistics at any time. Check status by starting the Status program from the MS Windows Start Menu or start your browser and go to this URL:

`http://myserver:port/status.html`

Replace myserver:port with your server's URL and the port on which ajmUpload is listening. You'll be asked for an ID and Password and you should enter the ID and Password you set up during Configuration. **Note that if you do not enter an Admin ID in the configuration program, the Status Display will be disabled.** The Status Program will display licensing information (local computers only), configuration options and a number of key runtime statistics including number of upload sessions, files uploaded, bytes uploaded, etc. It will also provide a list of each upload in progress and status information for each upload.

Using the Samples

The samples are web pages that demonstrate how to upload files to ajmUpload and how to process the files in your web

server through a completion page that receives control once the upload has completed. **Each sample will require some customization for your environment. Each item that needs to be changed is noted at the top of each sample.** Most of the samples are designed to use the Progress Bar. The final sample will show you how to use ajmUpload without the Progress Bar.

All of the samples using the Progress Bar will submit the upload using a short Javascript function similar to the one shown in Figure 5.1 below. This function essentially opens a browser window that will contain the Progress Bar and submits the upload to the ajmUpload server. Before using any of the samples, change the 'myserver' and 'myport' variables located in this routine to reflect your server's URL and the port on which ajmUpload is listening. For example, if your server URL is www.acme.com and ajmUpload is configured to listen on port 81, change the variables to:

```
var myserver = www.acme.com;
var myport = 81;
```

The function is found in the file su.js located in the ajmUpload Samples directory.

```
function StartUpload() {

    var myserver = "Enter ajmUpload Server Here";
    var myport = "Enter ajmUpload Port (i.e. 81) Here";

    winOpts = "height=340,width=500,scrollbars=1,location=0,
        toolbar=0,menubar=0,resizable=0,status=0";

    if (document.UploadForm.Token) {
        Token = document.UploadForm.Token.value;}
    else {
        Token = (new Date()).getTime() % 1000000000;}

    window.open("http://" + myserver + ":" + myport +
        "/progress.html?Token=" + Token, null, winOpts);

    document.UploadForm.action = "http://" + myserver + ":" +
        myport + "/upload.html?Token=" + Token;

    document.UploadForm.submit();
}
```

Figure 6.1 Javascript Submission Procedure

Using ajmUpload with HTML

Uploading with HTML is done via standard form variables. Items to note are:

- You must use `enctype=multipart/form-data`. Use of `enctype="application/x-www-form-urlencoded"` is not supported.
- Use a standard button (`type=button`) instead of a Submit button (`type=submit`) and use an `OnClick` handler as shown in Fig 5.2 below. Note that the 'StartUpload' function is included in the Javascript file `su.js` described earlier in this chapter.
- Note that the action parameter is not included as the submission is handled via Javascript.

This HTML will open a new browser window for the Progress Bar and start uploading the file to your server.

See file `upload.htm` in the Samples directory for a more complete example of uploading via standard HTML.

```
<html>
<body>
<script src="/su.js" LANGUAGE="JavaScript"
TYPE="text/javascript"></script>
<form method="post" enctype="multipart/form-data"
name="UploadForm" id="UploadForm">
<input type="file" name="filefield"><br>
<input type="button" name="Button" value="Submit"
OnClick="StartUpload()">
</form>
</body>
</html>
```

Figure 6.2 – File Upload Submission HTML

Once the upload has completed, you may transfer to a standard HTML page to acknowledge receipt of the uploaded files. See file `UploadComplete.htm` in the Samples directory. It simply displays an acknowledgement message.

Using the Sample Scripts

The ajmUpload installation program will place all sample files into the Samples directory – usually C:\Program Files\ajm Software\ajmUpload\Samples. You can open the Samples directory from the ajmUpload Start Menu as well. ASP.NET, ASP, PHP and Cold Fusion samples are provided with this release. Please check the ajmUpload Support area on our website for additional technologies.

Each sample demonstrates how to process uploaded files in the Upload Completion Page. All samples will process and delete the summary file and then move the uploaded files to a directory of your choice. To use a sample, make the modifications described below and move it to an appropriate location on your web server. You must then enter the URL of the sample into the Completion Page field using the Configuration Program. It's located under the Options tab.

Each sample requires two modifications before use. The Upload Completion pages – named UploadComplete.xxx (where xxx is ASP, PHP, etc.) – contain variables called gblUploadDir and gblTargetDir that must be initialized to the name of the Upload Directory specified in the ajmUpload Configuration program and the name of the directory where the uploaded files will ultimately reside. Things to consider when using the samples:

- The samples are provided for demonstration purposes only. You are free to use them in any manner, but please insure that they meet your security and operational requirements.
- The sample programs generally do not perform error checking. Please add appropriate error checking as needed to meet your requirements.
- The sample programs must have permission to read and delete files in the Upload Directory as well as permission to create files and write files in the Target Directory.

The samples are generally consistent over all technologies and are made up of two components, the Completion Page and a class that encapsulates key data related to the Upload Session. The Completion Pages are named UploadComplete.xxx and the class is named UIResult.xxx. The class methods and properties are described below and any differences within specific scripting languages or technologies are highlighted in the section describing the technology or language.

The completion pages do the following:

- instantiate the UIResult class and delete the Upload Summary File
- List the uploaded files on a results page.
- Move and rename the uploaded files to their final target directory

The UIResult class contains the following properties and methods:

Properties

- Token - the Upload Token
- SessionEnd - date and time that the upload session ended
- UploadStartTime - date and time that the upload session started
- UploadStopTime - date and time that the last file was received
- IPAddr - uploader's IP Address
- TotalLength - Total number of bytes that should have been received
- HeaderLength - Length of all header data received
- ReceivedLength - Actual number of bytes received
- Files - Collection of all uploaded file names and related data.

Each **Files** element contains

- Name - Name of the uploaded file (assigned by ajmUpload)
 - SourceName - Original name of the uploaded file
 - Length - Length, in bytes, of the uploaded file
 - UIStartTime - Time the first byte of this file was received
 - UIEndTime - Time the last byte of this file was received
-
- Constructor - The constructor requires two parameters, the name of the Upload Summary File and a Boolean indicating whether the Upload Summary File should be kept or deleted. If the technology does not provide a constructor, we'll include a 'Load' function that must be called when appropriate (see description of each technology below).
 - UIMoveFile - A function to move/rename each uploaded file. ajmUpload assigns a unique name to each uploaded file for security reasons and to prevent

name conflicts. This function is used by the samples to rename each uploaded to its original name. If name conflicts are found, this method will append a numeric identifier to the file to make it unique and insure that no files are overwritten.

Using ajmUpload with ASP.NET

Please read the **Using the Sample Scripts** section first.

The ASP.NET sample Upload Completion Page is called UploadComplete.aspx and can be found in the Samples directory after installation. The sample is written using in-line VB.NET code, but any .NET language can be used - either in-line or code-behind. You must change the variables gblUploadDir and gblTargetDir to point to the Upload Directory defined in the ajmUpload Configuration program and the final Destination Directory for the uploaded files.

The UIResult class is contained in the file **UIResult.aspx** located in the samples directory. Include this script in your in-line ASP.NET page using a Server-Side Include such as

```
<!--#INCLUDE FILE="UIResult.aspx" -->
```

or you may optionally compile it (requires the VB.NET compiler available from Microsoft) as a DLL to use it as a code-behind or Import. The sample provided with ajmUpload uses the SSI for convenience.

The constructor for this class takes two parameters, the name of the Upload Summary file and a Boolean set to TRUE to delete the Upload Summary File or FALSE to keep it, i.e.

```
Dim gblUploadDir As String = "C:\Uploads\  
Dim gblTargetDir As String = "C:\SavedUploads\  
Dim ul as Ulresult  
  
Dim szUploadSummary As String = Request("Token")  
Dim szSummFile As String = gblUploadDir & _  
szUploadSummary & ".ajmSumm"  
  
ul = New ULResult(szSummFile, true)
```

Figure 6.3 – ASP.NET

After executing the code in Fig. 6.3, the UIResult properties are available and can be used to process the uploaded files and provide feedback to the uploader. The ASP.NET version has one additional property called

ErrorMsg

which may contain the text of any errors encountered in processing the Upload Summary File. Refer to the sample for additional details.

Using ajmUpload with ASP

Please read the **Using the Sample Scripts** section first.

The ASP sample Upload Completion Page is called UploadComplete.asp and can be found in the Samples directory after installation. The sample is written using in-line VBScript but Jscript can also be used. You must change the variables gblUploadDir and gblTargetDir to point to the Upload Directory defined in the ajmUpload Configuration program and the final Destination Directory for the uploaded files.

The UIResult class is contained in the file **UIResult.asp** located in the samples directory. Include this script in your in-line ASP page using a Server-Side Include such as

```
<!--#INCLUDE FILE="UIResult.asp" -->
```

ASP does not provide a constructor, so a Load Method is included to populate the class properties. The Load Method two parameters, the name of the Upload Summary file and a Boolean set to TRUE to delete the Upload Summary File or FALSE to keep it, i.e.

```
Dim gblUploadDir
Dim gblTargetDir
Dim szSummFile

gblUploadDir = "C:\Uploads\"
gblTargetDir = "C:\SavedUploads\"
szUploadSummary = Request("Token")
szSummFile = gblUploadDir & Request("Token") & _
    ".ajmSumm"

Dim ul
Set ul = New ULResult
ul.Load szSummFile, true
```

Figure 6.4 – ASP

After executing the code in Fig. 6.4, the UIResult properties are available and can be used to process the uploaded files and provide feedback to the uploader. Please note that the Files collection is name mFiles in the ASP sample. Refer to the sample for additional details.

Using ajmUpload with PHP

Please read the **Using the Sample Scripts** section first.

The PHP sample Upload Completion Page is called UploadComplete.php and can be found in the Samples directory after installation. The sample was developed using PHP 4.3. You must change the variables \$gblUploadDir and \$gblTargetDir to point to the Upload Directory defined in the ajmUpload Configuration program and the final Destination Directory for the uploaded files.

The UIResult class is contained in the file **UIResult.php** located in the samples directory. Include this script in your PHP page using an Include:

```
include('UIResult.php');
```

The constructor for this class takes two parameters, the name of the Upload Summary file and a Boolean set to TRUE to delete the Upload Summary File or FALSE to keep it, i.e.

```
$gblUploadDir = "C:\\Uploads\\";  
$gblTargetDir = "C:\\SavedUploads\\";  
  
$szUploadSummary = $_REQUEST["Token"];  
$szSummFile =  
$gblUploadDir.$szUploadSummary.".ajmSumm";  
  
$ul = new Ulresult($szSummFile, true);
```

Figure 6.5 – PHP

After executing the code in Fig. 6.5, the Ulresult properties are available and can be used to process the uploaded files and provide feedback to the uploader. Refer to the sample for additional details.

Using ajmUpload with Cold Fusion

Please read the **Using the Sample Scripts** section first.

The Cold Fusion sample Upload Completion Page is called UploadComplete.cfm and can be found in the Samples directory after installation. The sample was developed using Cold Fusion MX. You must change the variables \$gblUploadDir and \$gblTargetDir to point to the Upload Directory defined in the ajmUpload Configuration program and the final Destination Directory for the uploaded files.

The UIResult class is contained in the file **UIResult.php** located in the samples directory. Include this script in your PHP page using an Include:

```
include('UIResult.php');
```

The constructor for this class takes two parameters, the name of the Upload Summary file and a Boolean set to TRUE to delete the Upload Summary File or FALSE to keep it, i.e.

```
<cfset szToken = #URL.Token#>
<cfset szfile = "C:\Uploads\" & Token & ".ajmSumm">

<!-- Create an instance of the UlResult component -->

<cfobject name="myUlResult" component="UlResult">

<!--- Execute the Load( ) method --->
<cfset LoadResult = myUlResult.Load(szFile)>
<cfset st_Files = myUlResult.Files(>
```

Figure 6.6 – Cold Fusion

After executing the code in Fig. 6.6, the Ulresult properties are available and can be used to process the uploaded files and provide feedback to the uploader. Refer to the sample for additional details.

Using ajmUpload without the Progress Bar

If the Progress Bar is not needed, you may upload files using standard HTML as shown below. Note that the form include an 'action' keyword that will post the form to the ajmUpload server (you must change 'myserver and port' to appropriate values). In this example, we've change the Submit button to a standard 'Submit' (type="submit"). See file uploadnp.htm in the Samples directory for a complete example.

```
<html>
<body>
<form method="post" enctype="multipart/form-data"
name="UploadForm" id="UploadForm"
action="http://myserver:port/upload.html">
<input type="file" name="filefield"><br>
<input type="submit" name="submit" value="Submit">
</form>
</body>
</html>
```

Figure 5.3 – File Upload without a Progress Bar

7. Custom Progress Page

Overview

You can optionally develop a custom Progress Page for use with ajmUpload in any of the .NET languages. Any valid .NET construct is supported. The ajmUpload install creates a virtual web root directory in the ajmUpload Program directory called wwwroot. It is usually found in

C:\Program Files\ajm Software\ajmUpload\wwwroot

Your custom Progress Page(s) must be placed in this directory or a subdirectory defined here. The page may use in-line code or compiled code. If you compile your Progress Page, you must place the resulting DLL in the /bin directory under the virtual web root usually found at

C:\Program Files\ajm Software\ajmUpload\wwwroot\bin

This directory will initially contain a file called ASPHost.dll. This file is required and cannot be moved or deleted.

Note: When using custom progress pages, you MUST check the 'Allow Custom ASP.NET Progress Pages' CheckBox on the 'Options' Tab of the Configuration program. If you do not check this box, the custom page will not function.

Building a Custom Progress Page

Your page can contain any valid ASP.NET construct and can be written in any .NET language. A sample written in VB.NET is provided and can be found in the Samples directory. All upload parameters are passed as a Query string and can be accessed via the HTTPResponse.QueryString Collection or the intrinsic Request object (i.e. Request("varname")).

The actual parameters passed will vary depending on the current state of the upload. The upload can be in one of these states:

- In Progress
- Completed

- Aborted
- Error

The current state is available as a text string in the 'Status' parameter.

Sample usage is shown in Figure 6.1 below:

```

Select Case Request("Status").tolower
  Case "in progress"
    DoInProgress()
  Case "complete"
    DoComplete()
  Case "abort"
    DoAbort()
  Case "error"
    DoError()
  Case Else
    DoUnknown
End Select

```

Figure 6.1 – Determine Upload Status

The Parameters vary depending on the upload status and are:

If Request("Status") = "In Progress"

Field Name	Description
Token	Upload Token
Start	Time the Upload Started
TotalBytes	Total length of all files and header data
TotalBytesRead	length of all data received so far
CurrentFile	(Optional) Name of the file being received right now
CfileBytesReceived	(Optional) Length of all data received for CurrentFile
Files	(Optional) List of all files uploaded so far It has the format: "file1;len1;file2;len2;..." (i.e) "file1.dat;12345;file2.dat;3457789;"

If Request("Status") = "Complete"

Field Name	Description
Token	Upload Token
Start	Time the Upload Started
TotalBytes	Total length of all files and header data
TotalBytesRead	length of all data received so far
Files	(Optional) List of all files uploaded so far It has the format: "file1; len1; file2; len2; ..." (i.e) "file1.dat; 12345; file2.dat; 3457789;"

If Request("Status") = "Abort"

Field Name	Description
Token	Upload Token
Reason	Reason the upload was aborted
AbortTime	Time the upload was aborted
Start	Time the Upload Started
TotalBytes	Total length of all files and header data
TotalBytesRead	Length of all data received so far
Files	(Optional) List of all files uploaded so far It has the format: "file1; len1; file2; len2; ..." (i.e) "file1.dat; 12345; file2.dat; 3457789;"

If Request("Status") = "Error"

Field Name	Description
Reason	Error Description
AbortTime	Time the error was detected

Sample Progress Page

A sample custom progress page named 'Prg.aspx' is located in the samples directory – usually

C:\Program Files\ajm Software\ajmUpload\Samples\Custom

It was developed using VB.NET in-line code and demonstrates the techniques required to develop a custom page. Use the sample by copying it and associated graphics files to the virtual web root at

C:\Program Files\ajm Software\ajmUpload\wwwroot

You must then modify the Upload start script to use the custom page instead of the standard Progress Page. The Upload start script used by the samples is called su.js and a custom Upload start script is included in the Samples\Custom directory.

8. Reference Guide

ajmUpload Configuration

Configure ajmUpload using the Configuration program available from the Start Menu in the ajm Software Program Group. All options are stored in an XML file located in the program directory. Please DO NOT EDIT this file manually – always use the Configuration program. All changes except ports become effective immediately upon clicking Save or Apply. You must shut down ajmUpload using the Service Control Manager for changes to the upload port to become effective.

Buffer Size

Select the network buffer size used to receive uploads. In general, smaller buffer sizes consume less memory but require more frequent I/O. The correct buffer size selection depends on the application. If you anticipate that most uploads will originate via the internet on relatively slow connections, use a smaller buffer size. If you will be receiving large uploads via an internal network, use larger buffer sizes. Note that using small buffers in a local area network environment will have a severe performance impact.

Max Upload Size

Enter the maximum allowable size – in Megabytes - allowed for upload. Enter 0 to remove size restrictions. Currently, the maximum file upload size is 1 byte short of 4Gb. This size applies to the session, not the file being uploaded. You cannot upload two 3Gb files at the same time, however, they can be uploaded in separate sessions.

Domain

Enter the domain name of the server on which ajmUpload will run. This is normally the name you would use in a browser

address line to connect to the server. It can be a fully qualified domain, an IP address or a network machine name. e.g.

<http://www.myserver.com>

www.myserver.com

<http://192.168.1.1>

myserver

The scheme - http:// - is optional.

Port

Enter the port on which ajmUpload will listen for upload requests. This port can be any valid open port on the server. You may not use port 0 or any port already being used by any other product on the server. This port will be used in your upload page to route uploaded files to ajmUpload. See the sample upload pages for more details.

Upload Directory

Enter the directory where upload will be placed. ajmUpload will save all uploaded files to this directory and your application will need to move the data to an appropriate location. If left blank, uploaded files will be saved to the Uploads sub-directory in the program directory. Normally, this is:

C:\Program Files\ajm Software\ajmUpload\Uploads

If the directory does not exist, ajmUpload will create the directory when the first upload is received.

Completion Page

Enter the complete URL of the web page that will receive control after an upload completes. This page must reside on a server that has access to the upload directory and authority to move files from that location. e.g.

<http://www.myserver.com/UploadComplete.asp>

www.myserver.com/UIComplete.php

<http://192.168.1.1/Upload.aspx>

The scheme - http:// - is optional. You may not include any form variables on this URL. For example:

www.myserver.com/Upload.php?name=upload

is not allowed.

Allow Custom ASP.NET Progress Pages

Click the CheckBox to allow custom ASP.NET Progress Pages. This feature does add substantial overhead, so check it only if you're using custom progress pages.

Receive Timeout

Enter the number of minutes (between 1 and 30) that ajmUpload will wait between messages before assuming that the upload has stalled. Once this timeout is reached, ajmUpload will cancel the upload.

Progress Window Close

This option controls the disposition of the Progress Window at the end of the upload. Select **Button** to include a 'Close Window' button in the Progress Window when the upload completes – this is the default. Select **None** to place no extra objects in the Progress Window. This option requires the uploader to manually close the Progress Window. Select **Auto** to automatically close the Progress Window at the end of the upload. Note that if an error occurs during the upload, this option is ignored as the Progress Window contains the error message explaining why the upload did not complete.

Upload Completion Action

This option controls how the **Completion Page** is started. Use **Standard** to display a summary of the files received for 5 seconds before transferring to the Completion Page. Use **Immediate** to bypass the upload summary display and transfer to the Completion Page immediately after the upload completes. The **Post** option transfers control immediately to the Completion using a HTTP POST and including all form variables on the original upload page. Use the **Post** option to maintain Session State between the upload page and the Completion Page in ASP.NET.

Auth PlugIn

Enter the name and location of the Authorization Plug-In DLL. This DLL is called before each file is uploaded to the server. Please see Appendix A for details on creating and debugging

the Plug-In. Click **Permanent** to load the DLL permanently into memory or click **Test** to reload the DLL each time a file is uploaded. Use **Test** only when developing and testing the Plug-In. This option uses more resources than **Permanent** but allows the Plug-In to be replaced without shutting down ajmUpload. Once the Plug-In is written and tested, use the **Permanent** option. Please note that the **Permanent** option permanently loads the Plug-In into memory and requires that ajmUpload be stopped to replace it.

Diagnostic Mode

Check 'Log Diagnostic Data' if you would like to log all errors to the Diagnostics log. Normally, errors will not be recorded unless this box is checked. Using Diagnostic Mode does not consume any significant resources and we recommend that you check this box.

Log Raw Data

Check this box if you would like to write the raw data streams received by ajmUpload to a log file. Please note that using this feature will generate substantial disk overhead and should not be used only in a test environment to perform troubleshooting.

System Log

Enter the name of the file that will contain the Log entries generated by ajmUpload. This log conforms to the W3C Extended Format used by IIS and rolls over nightly.

Diag Log

Enter the name of the file that will contain the Diagnostic log entries. This file is used only when the 'Log Diagnostic Data' box is checked. If left blank, the log will be written to 'ACTIVITY.LOG' in the program directory.

Data Log

Enter the name of the file that will contain the raw data streams. This file is used only when the 'Log Raw Data' box is checked. If left blank, the data log will be written to 'DATA.LOG' in the program directory.

View Diags

Click this button to view the Diagnostic log. The log can optionally be viewed by any text editor,

Admin ID

Enter an ID used to view status information. Please note that if this field is left blank, the Status Page will be disabled.

Password

Enter the password used for administrative requests.

Appendix A. Using the Authorization PlugIn

Authorization Overview

The HTTP Protocol is a connectionless protocol that maintains no state information between messages. Generally this means that a browser usually sends one message to a web server and expects to receive a single response. There are exceptions, such as challenge/response authentication, but they require additional setup or configuration and usually, more programming. When uploading files, this means that you can't really authenticate or authorize the upload until the file is actually sent, in its entirety, to the server. This problem occurs because most servers will receive the entire message from the browser before acting on it. Uploaded files are usually sent in a single logical message, regardless of size. This is certainly true for PHP based sites as well as ASP.NET based sites. This can create a number of problems including:

- Running out of disk space on the server
- upload large files only to find the user is not authorized
- upload complete files only to find that a user has exceeded his space allocation
- in IIS/ASP.NET based systems, the ASP.NET worker process can actually reset itself or run into performance problems when receiving very large files

ajmUpload solves these problems by using an Authorization PlugIn that is called before any file is uploaded. The PlugIn has access to key information and can authorize or cancel the upload as needed. The ajmUpload installation provides a stub PlugIn that you can use to create your own Authorization PlugIn as well as two samples that demonstrate UserID/Password authentication. Each of these is explained below.

Writing the PlugIn

This discussion on writing and debugging the Authorization Plug-In assumes you have Visual Studio .NET installed and some experience in developing .NET programs. Although this is

not absolutely required, it certainly makes development much easier. The Plug-In stub program shown below is installed in the **Samples** directory. It contains the basic elements needed by the Plug-In.

```
Imports Utility

<Serializable()> _
Public Class ajmPlugIn
    Inherits MarshalByRefObject
    Implements IajmPlugIn

#Region " Properties - Do Not Modify"

    Dim mMsg As String          'Message displayed to uploader if the upload request
    '                            is refused. You should enter a descriptive reason
    '                            here if the Authorize Function returns false. Use
    '                            HTML tags as needed.
    Dim mIPAddress As String    'Readonly IP address of the uploader
    Dim mContentLength As Long  'Total length of the data being uploaded. This includes
    '                            the sum of the lengths of all files being uploaded plus
    '                            the length of any header information being uploaded.
    '                            Header length may vary, but you can estimate 80 bytes/file
    Dim mFileSequence As Integer '0 based sequence of the file being uploaded. 0 = 1st file
    '                            uploaded, 1 = 2nd file uploaded, etc.
    Dim mFileName As String     'Name of the file being uploaded
    Dim mFileLength As Long     'Actual length of the last file uploaded
    Dim mUploadDir as String     'Name of the directory where the file will be saved
    Dim Request As New Hashtable 'All form variables you placed on the upload form

    'Class constructor
    Public Sub New()

    End Sub

    'This function is called before each file is uploaded. Return True to
    'accept the upload and false to cancel the upload. All form variables
    'are found in the 'Request' collection and can be accessed using the form
    'variable name you assigned on the upload page.
    '
    '    e.g. Dim szUserID as string = Request("UserID")
    '
    Public Function Authorize() As Boolean Implements IajmPlugIn.Authorize
        Return True
    End Function

End Class
```

Authorization Plug-In Stub

The sample is written in VB.NET, but can be ported to C# fairly quickly. When porting, please note that you need a reference to **Utility.dll** found in the installation directory – usually C:\Program Files\ajm Software\ajmUpload. This dll contains the interface IajmPlugIn which is used to insure that all required methods and properties are defined in the PlugIn. The class must be named **ajmPlugIn** and must be contained in the **ajmUpload** Namespace. Additionally, you must give the PlugIn Assembly a strong name. You may use the key pair provided in

the file **sgnkey.snk** located in the Samples\ajmPlugIn directory. The stub and sample PlugIn projects are already configured to use this key pair. The PlugIn contains several variables a single Function called **Authorize**. All of the variables except **mMsg** are ReadOnly and changing their values will have no effect on the operation of ajmUpload. The **Authorize** function is called before any file is uploaded. If you are uploading more than one file on a single HTML page, the **Authorize** function will be called multiple times – once for each file uploaded.

Each of these local variables are available when the **Authorize** function is executed.

- **mIPAddress** is the readonly IP address of the uploader in dotted decimal notation
- **mContentLength** is the Total length of the data being uploaded. This includes the sum of the lengths of all files being uploaded plus the length of any header information being uploaded. Header length may vary, but you can estimate 80 bytes per file. The HTTP Upload Protocol does not require individual file lengths or counts, so the actual number of files being uploaded and their individual lengths are not known until the upload has completed.
- **mFileSequence** is the 0 based sequence of the file being uploaded. 0 = 1st file uploaded, 1 = 2nd file uploaded, etc.
- **mFileName** is the name of the file being uploaded
- **mFileLength** is the actual length of the last file uploaded. Note that this field is not the length of the file about to be uploaded. It is useful only when more than one file is being uploaded from the same page. It's value is 0 for the first file being uploaded.
- **mUploadDir** is the name of the directory where the file will be saved
- **Request** is a collection of form variables you placed on the upload form. It is a HashTable and variables can be retrieved using the HTML form variable name. (**IMPORTANT** please read the section on creating HTML for the PlugIn below).

The **Authorize** function should evaluate the information presented and authorize the upload by returning **True** or cancel the upload by returning **False**. Note that if multiple files are being uploaded, returning **False** will cancel the current file being uploaded as well as any subsequent files being uploaded in this session. Files already received will not be affected. For example, if a user uploads three files and you accept the first file but cancel the second file, the first file will be accepted and

saved to disk. The second and third files will be cancelled. If you do cancel an upload, place a descriptive reason in the **mMsg** field. This reason will be displayed in the Progress Window and you may use valid HTML tags in this field. For example, if you need to split the reason across two lines, use the **
** tag as demonstrated here:

```
mMsg = "User " & Request("UserID") & " Upload cancelled" & _  
      "<br>Quota exceeded."
```

Note that, due to limitations in the protocol the main browser window will display a very general error message (i.e. – Internet Explorer displays a ‘Server not Found’ message) if you cancel the upload and your user will not be transferred to the Completion Page.

Creating HTML for the PlugIn

There is an important consideration when creating HTML for the PlugIn. Most browsers return form variables in the order in which they appear in the HTML. This means that any variables you need in the Authorization PlugIn **must** appear before any **'input type=file'**. For example, consider the following HTML fragment:

```
<form action="" method="post" enctype="multipart/form-data" name="form1">  
  <p>  
    <input type="file" name="file"><br>  
    UserID:<input type="text" name="textfield"><br>  
    Password:<input type="password" name="textfield2">  
  </p>  
</form>
```

Note that the file appears first, so the browser would send the entire file to the server before transmitting the UserID and Password variables. This means that the Authorization PlugIn would not see those fields. Use the following fragment to send the file:

```
<form action="" method="post" enctype="multipart/form-data" name="form1">  
  <p>  
    UserID:<input type="text" name="textfield"><br>  
    Password:<input type="password" name="textfield2"><br>  
    <input type="file" name="file">  
  </p>  
</form>
```

Note that the file now appears after the UserID and Password fields. If you require any fields to appear below the files, you can use layers to place the form variables in front of any files and make them render in the appropriate place on the page. For example, this fragment:

```
<form action="" method="post" enctype="multipart/form-data" name="form1">
<div id="Layer1" style="position:absolute; width:356px; height:59px; z-
index:1; left: 15px; top: 65px;">
  <p>UserID:<input name="UserID" type="text" id="UserID"><br>
  Password:<input name="Password" type="password" id="Password"></p>
</div>
<div id="Layer2" style="position:absolute; width:200px; height:31px; z-
index:2; left: 15px; top: 20px;">
  <input type="file" name="file">
</div>
</form>
```

would place the UserID and Password beneath the file field, however, since it appears before the file field in the HTML, the browser transmits these fields before the file is sent.

Debugging the Plugin

The debugging procedure described here requires that ajmUpload is installed on the development machine. Before starting Plugin development, open the ajmUpload configuration program and click on the Upload tab. Make sure the **Auth Plugin** parameter points to your DLL and click the **Test** option. **Test** insures that you may recompile your program without shutting down ajmUpload. When the appropriate configuration options are set, debugging the Plugin is accomplished using the following steps:

- Start ajmUpload
- Start Visual Studio.NET
- Load your Plugin project. You may use the Plugin stub program by navigating to the Samples directory – usually

C:/Program Files/ajm
Software/ajmUpload/Samples

and loading **ajmPlugin.vbjproj**.

- Set a breakpoint in the first executable line of the **Authorize** function
- Attach a debugger to the ajmUpload process by clicking **Debug...Processes** on the Visual Studio menu. Find and select the **ajmUpload** process in the list of available processes and click the **Attach** button.
- Start an upload – make sure your test upload page is configured properly. Configure and use one of the upload pages in the samples directory as needed.

Execution will stop at your breakpoint and you may debug your program normally. Please note the **Receive Timeout**

parameter may cause the upload to terminate prematurely if you spend too much time debugging, so you may want to set that parameter to an appropriate value while debugging.

Using the Sample Plugins

Two sample Plugins are provided with ajmUpload in addition to the Authorization Plugin stub program. The samples have two components – the actual Plugin dll (and associated source code) and a sample web page that uses the dll. The first sample is found in the Samples directory – normally

C:/Program Files/ajm
Software/ajmUpload/Samples/ajmPlugin/Sample1.

It demonstrates the use of a UserID and Password field on the upload page itself. To use this sample, load the project found in the Sample1 directory and compile it. Please make sure that all references are intact before compiling. The **Utility.dll** assembly is found in the program directory, usually

C:/Program Files/ajm Software/ajmUpload.

Make sure that ajmUpload is running and configured to use the Authorization Plugin found in the Sample1/bin directory. It's called ajmAuth1.dll. Move upload.htm from the Sample1 directory to your web server and make sure that su.js is configured properly as discussed in the **Quickstart** section. Start your web browser and navigate to upload.htm on your server.

If you require a login to reach the upload page and don't want your user to re-enter a UserID and Password, see the second sample Plugin found in

C:/Program Files/ajm
Software/ajmUpload/Samples/ajmPlugin/Sample2

It places the UserID and a Password hash in hidden fields on the upload page. To use this sample, load the project found in the Sample1 directory and compile it. Please make sure that all references are intact before compiling. The **Utility.dll** assembly is found in the program directory, usually

C:/Program Files/ajm Software/ajmUpload.

Make sure that ajmUpload is running and configured to use the Authorization Plugin found in the Sample2/bin directory. It's called ajmAuth2.dll. Move upload.htm from the Sample2 directory to your web server and make sure that su.js is configured properly as discussed in the **Quickstart** section. Start your web browser and navigate to upload.htm on your server.

Getting Technical Support

Support

Technical Support is available from our website at:

<http://www.ajmsoft.com/ac/support.html>

or via email at:

support@ajmsoft.com

See our website for additional contact options.

Technical support is free for ninety days after purchase. After 90 days, support is available on a per incident basis. Please see our website for current pricing. There is no charge for email support.