Posting Acceptor Operations Guide

Microsoft Site Server

Microsoft Corporation

Contents

Welcome

Microsoft® Posting Acceptor is a server add-on tool that Web content providers can use to publish their content using HTTP Post (RFC 1867). After installing Posting Acceptor on your Web server running Windows NT® Server, Windows NT Workstation, or Windows® 95, you will be able to provide a hosting service for users wanting to post Web content to your server. Resources are provided to assist with Posting Acceptor installation, configuration, and troubleshooting.

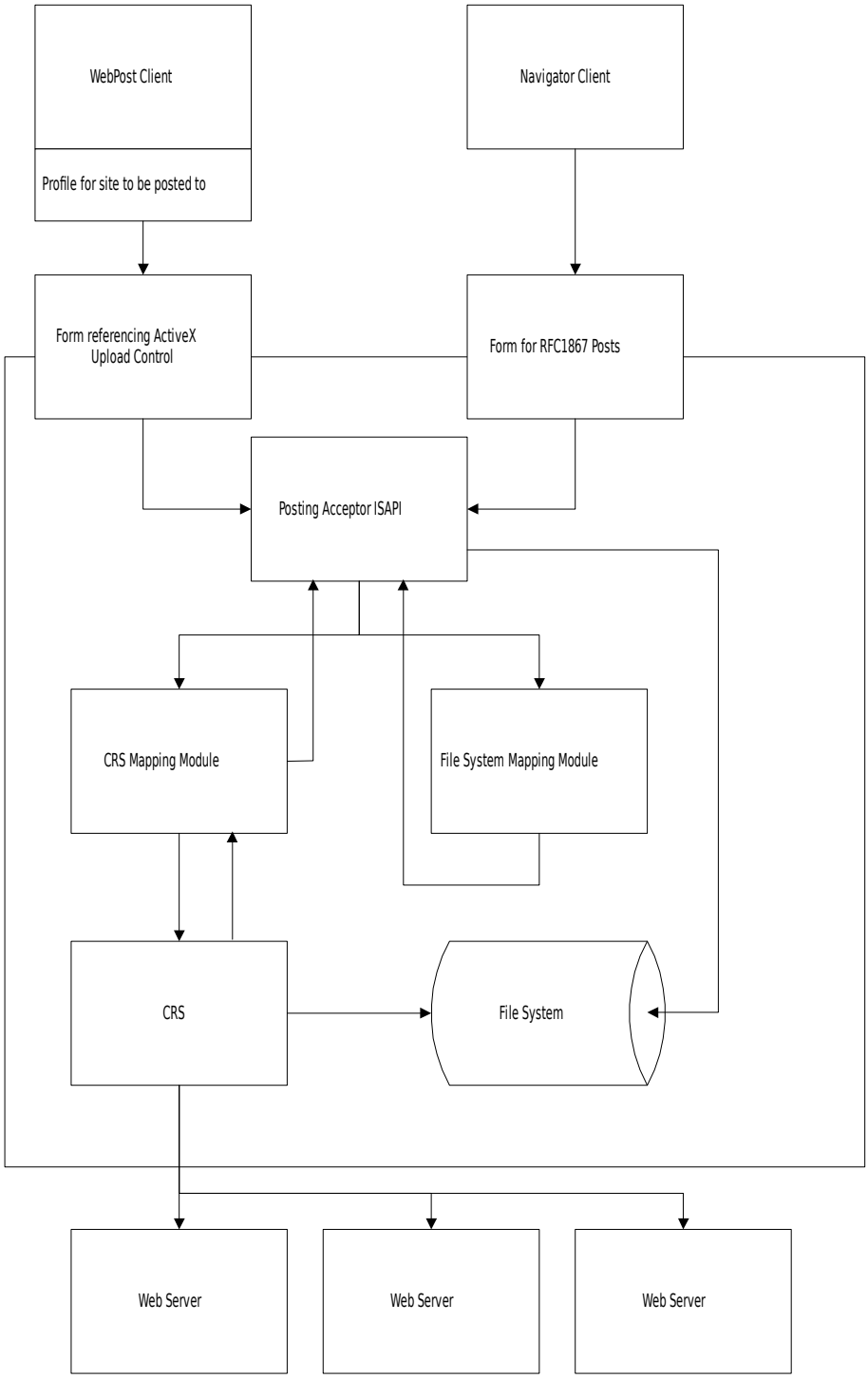
What Posting Acceptor Does

Posting Acceptor allows Microsoft Internet Information Server (IIS), Microsoft Peer Web Services, and Microsoft Personal Web Server to accept Web content from Microsoft Web Publishing Wizard/API and Netscape Navigator 2.02 or later through any standard HTTP connection. In conjunction with Microsoft Content Replication System (CRS), Posting Acceptor can also distribute content to multiple servers simultaneously.

How It Works

Posting Acceptor provides a receiver for Microsoft IIS to accept files from clients using the HTTP multipart/form-data method to post their Web content. Clients using this posting method include the Web Publishing Wizard/API 1.5 Microsoft ActiveX™ Upload control and Netscape Navigator 2.02 or later. Posting Acceptor can also provide the same functionality to a cluster of servers with content managed through CRS supporting the Web Publishing Wizard/API 1.5 client. In this way, Microsoft Web Publishing Wizard/API and any HTTP Post–compliant clients, can post content through a standard firewall.

For example, suppose an Internet service provider (ISP) decides to host Web pages for its users. The ISP simply installs Posting Acceptor on its Web server and creates a virtual directory called “users.” Users are then able to post their content to the server and retrieve it from a URL such as http://servername/users/username. A graphic representation of these relationships is provided in the following illustration.


WebPost Client
Navigator Client
Profile for site to be posted to
Form referencing ActiveX Upload Control
Form for RFC1867 Posts
Posting Acceptor ISAPI
CRS Mapping Module
File System Mapping Module
CRS
File System
Web Server
Web Server
Web Server

Conventions

The following text formats are used throughout this document.

Convention	Meaning
Bold	Indicates the actual commands, words, or characters that you type in a dialog box or at the command prompt.
Italic	Indicates a placeholder for information or parameters that you must provide. For example, if the procedure asks you to type *filename*, you must type the actual name of a file.
`Monospace`	Represents examples of screen text or entries that you might type at the command line or in initialization files.

Understanding Posting Acceptor

The following section provides an overview of the components, functionality, and concepts that make up Posting Acceptor operation.

Posting Acceptor Components

Two sample pages are provided with the Posting Acceptor server add-on: one designed to accept multi-part/form-data method posts from Netscape Navigator, and one to provide the ActiveX Upload control for Microsoft Internet Explorer users. You can modify these pages to fit your needs.

The Active Server Pages (ASP) file (upload.asp) detects the type of incoming browser and routes the user to the appropriate sample page. If the user has a browser that supports ActiveX controls, he or she is routed to the page containing the ActiveX Upload control. If it is not yet installed, the user's system is automatically updated with the control. The following table describes the functions of the components, including sample pages, that make up Posting Acceptor.

Component	Description	Function
Posting Acceptor ISAPI cpshost.dll	ISAPI application; core component of Posting Acceptor	Receives posted files from the content providers and saves them to the disk according to the TargetURL.
CRS Mapping Module crsmapr.dll	A module responsible for mapping TargetURLs into physical locations on the hard disk	Queries CRS for all MapURLs and compares them with the TargetURL. If a match is found, physical location on the hard disk is reported back to Posting Acceptor.
File System Mapping Module Built-in cpshost.dll	A module responsible for mapping TargetURLs into physical locations on the hard disk	Queries IIS (or PWS) for a mapping between the TargetURL and a physical location on the hard disk.
ActiveX Upload control flupl.cab	An ActiveX control that when embedded in a Web page can be used to post content files	Users can drag and drop files and folders or double-click the control to select files and folders to upload to your Web server.
Sample page for clients using HTTP (for example, Netscape Navigator) to post content uploadN.asp	An Active Server Pages (ASP) file containing a form with HTTP Post (RFC 1867)–specific fields in which to post files	Can be used by any HTTP Post (RFC 1867) Web browser (for example Netscape Navigator) to upload files to your server.
Sample page for clients using the ActiveX Upload control	An Active Server Pages (ASP) file containing the	Enables browsers supporting ActiveX to

Component	Description	Function
to post content uploadX.asp	ActiveX Upload control (flupl.cab).	post files and folders to your server via the ActiveX Upload control (flupl.cab).
Active Server Pages (ASP) upload.asp	An Active Server Pages (ASP) file that determines the type of client browser and refers it to the appropriate content-uploading form, either uploadN.asp or uploadX.asp	If the browser supports ActiveX controls, it is automatically referred to uploadX.asp when it hits this page. If the browser does not support the controls, it is referred to uploadN.asp
ASP PostInfo file postinfo.asp	The PostInfo file that is retrieved by WebPost to automatically configure client software for posting content	The PostInfo file contains the PostingURL and the TargetURL that WebPost uses to post content to your server.
Postinfo.eg file	A file located in your www root directory	Postinfo.eg is an “example” file of a line that needs to be added to default.htm so that WebPost can locate the "PostInfo" file. Setup adds this line to your default.htm (home) file.

Background Functions

The following section provides a summary of functions performed by Posting Acceptor and the concepts behind those functions. This includes authenticating content publishers, processing a content post after it is received, managing content reposting failure, and creating mapping modules.

Built as an ISAPI application, Posting Acceptor accepts HTTP-based “POST” requests that contain a TargetURL for the incoming content. Posting Acceptor then parses the URL to a base that matches a hosted site, which is determined by querying the configured mapping modules until one succeeds. The remainder of the URL is assumed to be subdirectories of the matched/hosted URL. The incoming files are then deposited into the appropriate directory selected by the module.

Posting Acceptor takes content posts from any client conforming to the rules outlined in the HTTP multi-part/form-data method (RFC 1867), including Netscape Navigator (with the use of a form on the server). Further posting options are also supported if the client code is Microsoft WebPost API based, for example, Web Publishing Wizard and any other applications using the ActiveX Upload control.

Authenticating Publishers

Authentication of publishers is handled by any mechanism that is exposed by Windows NT on the server that matches the authentication method supported by the client-side software; for example, Basic/NTLM/MSN/DPA (Distributed Password Authentication) or Microsoft Membership System.

Note

Posting Acceptor specifically disallows anonymous connections.

IIS and PWS enumerate supported authentication methods and negotiate with the client to find a match. You can control who is allowed to post to a given site by establishing accounts for those users on the server. You can also control which directories those users can post to by applying access control lists (ACLs) to the various destination directories for that site. For more information about applying ACLs to a destination directory, see the Windows NT and Windows 95 user guides.

Your client's publishing software can authenticate against your Web server to upload their files using Microsoft Membership System or any other authentication mechanisms supported by Microsoft Internet Information Server (IIS), Peer Web Services, and Personal Web Server.

If you have installed a Security Certificate on your Web server, your publishers are able to upload their files securely via HTTPS (Hypertext Transfer Protocol Secure) assuming that the client supports SSL (Secure Sockets Layer).

Processing After Receiving Posts

If you want to perform additional processing after a post is received, Posting Acceptor can call a secondary post-processing URL with all the form data except the contents of the posted files uploaded from the client. In place of the content is a list of locations and sizes of the files posted to the server. You are able to edit the PostInfo file in order to specify the post-processing URL.

If a PostInfo file is not supplied, the option of specifying a post-processing URL is exposed in the WebPost API. You can also add it to the HTTP (uploadN.asp) sample page. The post-processing URL is specified as a part of the PostingURL after the PUBLISH keyword.

As part of the repost, the acceptor sends URL-encoded form fields of well known names, which list the locations where each of the uploaded files was saved. This form data is appended to any form data that already resides in the request. For each of the files uploaded and saved, the following will be passed on to the repost URL:

```
FileName=page
FilePath=c:\test
FileSize=32
FileExtention=htm
FileName=...
```

PostInfo File

The PostInfo file contains several definitions for the WebPost client, one of those definitions being the posting URL, which by default points to Posting Acceptor. The PostInfo file is used by the WebPost Publishing Wizard only. The WebPost API on the client's machine figures out where to post content by information it receives from the PostInfo file.

Repost Failures

In the case of repost failure, a warning message embedded inside the HTML repost is sent to the user. A repost failure is marked as a warning, not an error. A program that only checks for errors, such as WebPost, is not aware that the repost failed. If the repost succeeds, anything returned from the repost is sent back to the client. The acceptor then becomes transparent, meaning that it does not send anything back on its own. If no content is returned from the repost, the acceptor sends a warning message back to the client.

Mapping Modules

The mapping module is a self-registering server that creates the appropriate entry in the Posting Acceptor's mapping modules registry during registration.

Inside the registry, Posting Acceptor expects to find DWORD values, names of which are the Class IDs of mappers, and the values of which are either 1 (enable) or 0 (disable). For example, the entry for the Microsoft Content Replication System (CRS) mapping module looks like this:

```
{66BE7352-83A0-11D0-A317-00C04FD7CFC5}:REG_DWORD:0x1
```

The acceptor queries each of the installed mapping modules until one of them succeeds and a physical location is returned to the acceptor. If no mapping module succeeds, the acceptor defaults to querying IIS for the physical location of the given URL. If the default also fails, the acceptor returns an error message to the client.

Posting Acceptor can be configured with one or several mapping modules to route the incoming files to the desired location. When more than one mapping module is available, you can control where the user's content is routed depending on user permissions and setup specifications in the mapping modules.

Note

Mapping module keys are located in HKLM\Software\Microsoft\WebPost\Acceptors\CPSHost\Mappers\

Setting Up Posting Acceptor

The following information lists the things you should know before installing Posting Acceptor, as well as procedures to follow as you install and configure Posting Acceptor. Before installing Posting Acceptor, note the minimum hardware and software requirements for the platforms that Posting Acceptor supports.

Hardware and Software Minimum Requirements

Posting Acceptor runs on the following platforms using either the Intel *x*86 or DEC Alpha processor. To successfully install Posting Acceptor, ensure the following minimum requirements are met.

Category	Windows NT Server	Windows NT Workstation	Windows 95
Hardware	486/33 or higher Intel processor or Alpha RISC processor compatible with Windows NT 16 MB of memory (64 MB recommended) 10 MB of hard disk space plus space for users' uploaded files	486/33 or higher Intel processor or Alpha RISC processor compatible with Windows NT 16 MB of memory (64 MB recommended) 10 MB of hard disk space plus space for users' uploaded files.	386DX or higher Intel processor (486/33 or higher recommended) 8 MB of memory (24 MB recommended) 10 MB of hard disk space plus space for users' uploaded files.
Software	IIS 3.0 or later Active Server Pages With or without Content Replication System	Peer Web Services 3.0 or later Active Server Pages No Content Replication System support	Personal Web Server 1.0a or later Active Server Pages* No Content Replication System support

* Active Server Pages (ASP) is not required, but will allow for automatic routing of the posting client to the proper page for a particular client tool. Microsoft IIS and PWS 3.0 provide ASP in their respective installations.

Installing Posting Acceptor

During Posting Acceptor installation, the ISAPI, mapping modules, sample forms, and PostInfo file are deposited by default into the correct virtual directories. You are required to insert the meta-tag into your default home page. As you install Posting Acceptor, you may accept the default directory locations for component installation or specify other directory locations.

- **To install Posting Acceptor**

  1. From the root directory, double-click the PostAcc folder. Double-click Posting Acceptor.exe and accept the license agreement.
  2. In the dialog box, enter the physical path to the virtual root of your Web server. You may select the default, or choose another path. Click OK. If you select another directory, you must edit all Active Server Pages (.asp) files and adjust the paths accordingly.
  3. In the dialog box, enter the directory where you want to install Posting Acceptor. This must be a virtual root of your Web server that has execute permissions. You may select the default, or choose another path. Click OK. If you select another directory, you must edit all .asp files and adjust the paths accordingly.
  4. Click OK to finalize Posting Acceptor installation. Reboot if necessary.
  5. Run your editor on your default home page. Insert the meta-tag found in the postinfo.eg file into your default home page. Inside the meta-tag, modify the path to your PostInfo file to suit your needs.

Note

It is not required that you add meta-tag information to any virtual root directory other than the default home page of your Web server. Setup automatically adds that line for you. You are required to edit the line to point to your PostInfo file if you do not use the Posting Acceptor default. (/scripts/default.asp)

Configuring Posting Acceptor

An unlimited number of configurations are possible when setting up Posting Acceptor. You have the option of either selecting the Posting Acceptor defaults or editing the configuration information to fit your needs.

Elective configurations include selecting the TargetURL and PostingURL to which users post, editing the Posting Acceptor sample pages, creating mapping modules, using Posting Acceptor with Microsoft Content Replication System (CRS), configuring for outstanding posts, and performing additional processing after a post is received.

Selecting TargetURL and PostingURL

Depending on the content provider's choice of client browser, you will modify different Active Server Pages (.asp) files to specify your TargetURL and PostingURL. For more information about interacting with content providers, see "What to Tell Content Providers" in the Posting Acceptor Administration section of this document.

- **To select the TargetURL and PostingURL for WebPost API clients**
  - Modify the postinfo.asp file.

- **To select the TargetURL and PostingURL for Netscape Navigator and other HTTP clients**
  - Modify the uploadN.asp file.

Configuring with Sample Pages

To quickly get you up and running with Posting Acceptor, two sample pages are provided in the product. One sample page, UploadX.asp, maps to users posting content with the WebPost API, and the other sample page, uploadN.asp, maps to users posting content with Netscape Navigator and other HTTP client browsers. For more information about sample pages, see the section "Posting Acceptor Components" in this document.

- **To configure Posting Acceptor using the WebPost API and Netscape Navigator/HTTP sample pages**
  - You may use both sample pages (uploadX.asp and uploadN.asp) as they currently exist in Posting Acceptor, or edit each sample page source file to suit your needs. With the HTTP sample page, you can also copy its forms directly into your existing Web pages.

Creating Mapping Modules

The mapping module is a self-registering server that creates the appropriate entry in the Posting Acceptor's mapping modules registry during registration. You have the option to accept the default mapping module or create other modules to suit your needs. For more information about mapping modules, see "Mapping Modules" in the "Background Functions" section of this document and sample code provided in the WebPost Software Developers Kit (SDK).

A mapping module must be an in-process server that supports the **IMapper** interface defined as follows:

```
const IID IID_IMapper = \
 {0x66BE7351,0x83A0,0x11D0,{0xA3,0x17,0x00,0xC0,0x4F,0xD7,0xCF,0xC5}};
```

This interface supports one additional method besides the standard ones:

```
HRESULT STDMETHODCALLTYPE GetLocation(
/* in */LPCTSTR szUrl,          // Destination URL ("TargetURL"variable)
/* in */LPCTSTR szUsername,     // User uploading files
/* out */   LPSTR szDestination,    // Physical disk location buffer
/* in */DWORD dwDestinationLength); // Length of the buffer
```

(For the acceptor, the length of the buffer will always be MAX_PATH.)

Note

You can enable or disable a mapping module by setting its value in the registry to 1 (enable) or 0 (disable).

Using Posting Acceptor with Microsoft Content Replication System

A mapping module for Content Replication System (CRS) is installed automatically when you install Posting Acceptor. If you have CRS installed on your system, you can use it to replicate content posted to your server or to other servers.

- **To use Posting Acceptor with CRS**
  - Set up an automatic project with a MapURL pointing to the content directory of your project. With the TargetURL being a superset of the MapURL, posted content is saved in the content directory via the CRS mapping module.

For example, if the TargetURL were http://server/content/html, the MapURL would be http://server/content.

For more information on setting up automatic projects, see the *CRS Operations Guide*.

Example Scenario Using CRS with Posting Acceptor

The following information is a detailed example of how to use CRS with Posting Acceptor.

In the example, the Internet service provider (ISP) wants users to post files to an intermediate machine instead of posting directly to the live WWW servers. The ISP has a posted file dropped into a directory that is running an Automatic Mode CRS project. User posts are then replicated to the end WWW servers.

The intermediate system is called stage01.mynet.com and the WWW server is called www.mynet.com. All systems are running Windows NT Server 4.0 and IIS 3.0.

On www.mynet.com, the ISP has configured the following IIS virtual directory:

/users maps to: d:\inetpub\users

On stage01.mynet.com, the content directory for dropping of files is e:\uploads.

User accounts on stage01.mynet.com must be in the CRS Users group.

CRS Setup

On stage01.mynet.com, create the following project:

```
crs addproj stage e:\uploads www.mynet.com /automatic /fastmode /mapurl
http://www.mynet.com
```

On www.mynet.com, create the following project entry:

```
crs addproj stage d:\inetpub\wwwroot\users
```

On stage01.mynet.com, start the crs project.

```
crs startrep stage
```

Posting Acceptor Setup

This example assumes that the posting acceptor sample has been installed on the system.

On www.mynet.com, add the following line to the default page of root.

```
<META name="postinfo" content
http://stage01.mynet.com/scripts/postinfo.asp>
```

On stage01.mynet.com, create a postinfo.asp file with this content:

```
<% Response.Buffer = True %>
Version=1.5
[WebPost.PostWPP]
PostingURL="http://<%= Request.ServerVariables("SERVER_NAME")
%>/scripts/cpshost.dll?PUBLISH"
BaseURL="http://<%= Request.ServerVariables("SERVER_NAME") %>/users/<%=
Request.ServerVariables("LOGON_USER") %>"
```

Note

Stage01.mynet.com, turns off anonymous access so that the postinfo.asp file can determine LOGON_USER.

Configuring for Outstanding Posts

To keep the system from being overwhelmed by excessive amounts of content posts, for example, the results from a denial of service attack, Posting Acceptor has implemented limitations on two posting values. The limitations are placed on the total number of outstanding posts and on the maximum post duration, both of which are adjustable in the registry. For more information, see “RegKey,” in the “Posting Acceptor Administration” section of this document:

- **MaximumOpenTransactions**. This value controls maximum outstanding posts. (Default is 200 posts.)
- **OpenTransactionsTimeout**. This value controls maximum outstanding post duration. (Default is 600 seconds or five minutes.)

- **To configure for outstanding posts**
  - In a registry editor, open up the registry key, HKLM\Software\Microsoft\ WebPost\Acceptors\CPSHost. Adjust the value limitations to suit your needs.

Specifying a Post-Processing URL

If you want to perform additional configuring after a post is received, Posting Acceptor is able to call a secondary or "post-processing" URL with all the form data except that of the posted files uploaded from the client. In place of the content is a list of locations and sizes of the files posted to the server. You are able to edit the PostInfo file if you wish to specify a post-processing URL. For more information about the PostInfo file, see "PostInfo File," in the "Background Functions" section of this document.

- **To specify a post-processing URL for WebPost API clients**
  - Modify the PostInfo file for WebPost.

- **To specify a post-processing URL for Netscape Navigator and other HTTP clients**
  - Modify the HTTP sample page (uploadN.asp) that contains the file upload form information.

Posting Acceptor Administration

This section contains information necessary to maintain optimal functioning of Posting Acceptor. This includes performance monitoring tables and other important information you must provide for users to help them post Web content properly.

Performance Monitoring

Posting Acceptor exposes the following Performance Monitor counters, Windows NT events, and registry keys to track the activity of multiple components and operations.

Performance Monitor Counters

For performance monitoring on Windows NT, the Posting Acceptor exports one object called “RFC 1867 Posting Acceptor.” This object contains the following counters.

Name	Explanation
Current Posts	The number of current posts being processed by the acceptor.
Unresolved Posts	The number of current posts awaiting commitment from the clients.
Maximum Posts	The maximum number of instantaneous posts over the lifetime of the acceptor.
Total Posts Received	The number of posts received up to this point.
Total Successful Posts	The number of successful posts received up to this point.
Total Re-posts Done	The number of reposts performed up to this point.
Total Files Received	The number of files received up to this point.
Total Failed Posts	The number of posts that failed so far.
Total Bytes Received	The number of bytes received so far.
Successful Posts/Sec	The rate of successful posts.
Re-posts/Sec	The rate at which reposts are done.
Posts Received/Sec	The rate at which posts are received.
Files Received/Sec	The rate at which files are posted.
Failed Posts/Sec	The rate at which posts are failing.
Bytes Received/Sec	The rate at which bytes are received.

Windows NT Events

Posting Acceptor logs a limited number of Windows NT events to notify you of failed posting or reposting attempts. The logs detail the following information:

- Who attempted to post.
- What URL was specified.
- What error or warning was encountered.

Event ID	0x2201
Meaning	Cannot move file to its final destination. Further processing is stopped.
Cause	NA
Solution	None

Event ID	0x2202
Meaning	TargetURL is invalid.
Cause	NA
Solution	None

Event ID	0x2203
Meaning	No valid files received.
Cause	NA
Solution	None

Event ID	0x2204
Meaning	Cannot initialize OLE libraries.
Cause	NA
Solution	None

Event ID	0x2205
Meaning	The query string is invalid.
Cause	NA
Solution	None

Event ID	0x2206
Meaning	This is your server and not the specified server in the TargetURL.
Cause	NA
Solution	None

Event ID 0x2207
Meaning File posted successfully.
Cause NA
Solution None

Event ID 0x2208
Meaning Reposting to URL failed.
Cause NA
Solution None

Event ID 0x2209
Meaning Reposting to URL successful.
Cause NA
Solution None

Event ID 0x220A
Meaning Reposting to URL successful. No content came back from the server.
Cause NA
Solution None

Event ID 0x220B
Meaning Your transaction is either invalid or has expired.
Please try reposting your files again.
Cause NA
Solution None

Event ID 0x220C
Meaning Your files cannot be posted at this time.
Cause There are too many open transactions.
Solution Please try reposting your files again later.

Event ID	0x2281
Meaning	Cannot move file to its final destination. Further processing is stopped.
Cause	The user may not have adequate permissions on the machine to move posted files to a final destination.
Solution	Check the connected user's permissions to see if they are allowed to create directories and files in the specified destination.

Event ID	0x2282
Meaning	TargetURL is invalid.
Cause	The TargetURL cannot be resolved**.**
Solution	Ask the user to specify the correct TargetURL. Also, confirm that you have correctly setup the URL in the postinfo.asp and upload.asp files.

Event ID	0x2283
Meaning	No valid files received.
Cause	The post contained no files. The user is sending posts that do not contain any valid (non-empty) files.
Solution	None

Event ID	0x2284
Meaning	Cannot initialize OLE libraries.
Cause	Initialization of OLE failed. This may be a bad operating system installation, or the connected user is not able to use OLE on this system.
Solution	None

Event ID	0x2285
Meaning	The query string is invalid.
Cause	The query string (part of the PostingURL) is invalid.
Solution	Tell the user to specify the correct PostingURL. Also, confirm that you have correctly setup the URL in postinfo.asp and upload*?*.asp files.

ID	0x2286
Meaning	This is your server and not the specified server in the targeted URL.
Cause	Someone is attempting to post files to a target server that is not yours.
Solution	Tell user to specify the correct TargetURL. Also, confirm that you have correctly setup the URL in postinfo.asp and upload*?*.asp files.

ID	0x2288
Meaning	Reposting to specified repost URL failed.
Cause	An attempted repost of the data to the Re-postURL failed.
Solution	Check the Re-postURL for accuracy.

ID	0x228A
Meaning	Reposting to specified Re-postURL successful. No content came back from the server.
Cause	The repost was successful. No HTML content came back from the server**.**
Solution	May be intentional. If not intentional, check the Re-postURL for accuracy.

ID	0x228B
Meaning	User specified transaction is either invalid or has expired.
Cause	An invalid session (transaction) ID was sent by the client. The client stayed disconnected for more than the allowed transaction timeout specified in the registry.
Solution	None

ID	0x228C
Meaning	There are too many open transactions to complete the user's request.
Cause	Too many users are trying to post simultaneously.
Solution	If your server has adequate space, increase the appropriate value limit in the registry.

ID	0x22FC
Meaning	Transaction ID has expired.
Cause	The user never committed the transaction so it expired.
Solution	None

ID	0x22FD
Meaning	Unknown error was encountered.
Cause	An internal error occurred.
Solution	None

ID	0x22FE
Meaning	Unknown error was encountered.
Cause	NA
Solution	None

Registry Keys

Microsoft Posting Acceptor contains a single registry key: HKLM\Software\Microsoft\WebPost\Acceptors\CPSHost

The registry contains the two values that you can adjust to suit your needs. For more information, see the section "Configuring Outstanding Posts," in this document:

- **OpenTransactionsTimeout**. This controls maximum outstanding post duration. (Default is 600 seconds or five minutes.)
- **MaximumOpenTransactions:** This controls maximum outstanding posts. (Default is 200.)

What to Tell Content Providers Posting To Your Server

Depending upon the type of browser used by your content providers, you must furnish them with certain information to make sure that their files are successfully posted.

Posting Information for WebPost Clients

If a content provider is publishing to your server with the WebPost API, you must convey the name of the server to which content will be posted. If you do not specify your PostInfo file, you must also tell them the PostingURL and TargetURL. WebPost API performs a GET against the virtual root of the TargetURL server to determine the location of the PostInfo file on the server. The WebPost API then retrieves the PostInfo file and parses out the PostingURL and TargetURL if they were specified earlier. If either the PostingURL or the TargetURL were not specified, the content provider must enter that information manually in Web Publishing Wizard.

Posting Information for Netscape Navigator and Other HTTP Clients

If a user is publishing to your hosting server with Netscape Navigator or another HTTP posting method, content is posted through the form that is embedded in the uploadN.asp sample page and is provided to the user. The PostingURL is specified

by the action field in that form and the TargetURL is specified by the “TargetURL” form variable.

Note

This procedure is optional. You may accept the default, http://yourserver/users/username, or select another URL.

Troubleshooting

This section provides references to assist with any questions you may have about installing, configuring, and using Posting Acceptor.

Windows NT Users

If you are having problems after checking for possible configuration errors, client-side errors, and any Windows NT errors in the events log, please post e-mail to the Posting Acceptor newsgroup:

news://msnews-gw/microsoft.public.site.posting-acceptor

Windows 95 Users

If you are having problems after checking for possible configuration errors and client-side errors, please post e-mail to the Posting Acceptor newsgroup:

news://msnews-gw/microsoft.public.site.posting-acceptor

Glossary

An experimental protocol proposed for the Internet community. It is a suggested extension to HTML that would allow content providers to express file upload requests uniformly, and a MIME-compatible representation for file upload responses.

A self-registering server that creates the appropriate entry in the Posting Acceptor's mapping modules registry key during registration.

The PostInfo file is used by Web Post API and the ActiveX Upload control to automatically set up a client to publish content to your server. A sample PostInfo file (postinfo.asp) is provided in Posting Acceptor.

The following table provides additional details for the parameters in the sample PostInfo file.

Value	Description
Version	Must be 1.5 to indicate compatibility with this release.
Provider's GUID	Globally unique identifier.
PostingURL	URL to post to.
TargetURL	Target URL for the uploaded content.
Default page	Name of default page used on the server. For example, the IIS default is "default.htm."

PostingURL is used by content providers to post files. It is the full URL path to the Posting Acceptor, and appears as the following default unless otherwise specified:

http:\\your_server_name\path-to-PostingAcceptor*?*PUBLISH

Post-processingURL is part of the PostingURL. It can be created by appending the PostingURL to read:

http:\\your_server_name\path-to-PostingAcceptor*?*PUBLISH*?*http:\\Post-processingURL

The TargetURL is the final target for a content provider's uploaded files, as well as the URL they use to retrieve content.

The client side of the WebPost API contains the flupl.cab, Web Publishing Wizard, and any other application that uses ActiveX controls.