File Transfer Basics

An overview of file content types, and how Rumpus processes them.

Every computer file is, in reality, merely a series of numeric values, stored in series as a single object. Whether those values represent text (in ASCII format, for example), an image (such as a JPEG or GIF) or any other type of content, all files require some sort of interpretation. What the file represents, and how the data in the file should be interpreted, depends on the file's content type, which we'll refer to as the file type.

As you can see, a file's content type is extremely important, since processing an image file with an audio processor would end in a meaningless result, for example. So, managing file types is a key component of most operating systems, and must be handled correctly when transferring files between computers.

Type And Creator Codes

The original mechanism used to manage file content types on the Mac OS is using a 4 digit type code, along with another 4 digit creator code. These codes are registered with Apple to ensure their uniqueness, and are associated with every Mac file as part of each file's "Finder Info" (which also includes such elements as file modification date, icon display information, etc.). The File Type represents the content type of the file, such as "TEXT", "JPEG", and so on. The Creator Code tells the Mac OS what application originally created the file, or at least what application should be used to view the document. These 4 digit codes are what has allowed the Mac OS to automatically launch the correct application when you double-click a file for years.

Filename Extensions

Other common operating systems, including Windows and Unix, rely on the filename to convey the file's content type, using the convention of a dot followed by a few characters. This is known as a filename extension, with common examples being ".html", ".txt", ".gif", and so on. Filename extensions are, in at least some ways, less elegant than the original Macintosh solution, but are widely supported and do offer some advantages.

Because Mac OS X is based on Unix, filename extensions are now the primary mechanism used to determine file types, though type and creator codes are also supported by the OS X Finder, and remain very useful for determining the best application to process a given file.

File Types In Rumpus

It is important to realize that Rumpus does essentially no processing of the files it transfers. All files transferred through Rumpus are treated as the anonymous blob of numbers that the file really is (with the exception of ASCII mode transfers, see below for details). This is no different than if you were to copy the file from one machine to another using any LAN file server or even floppy disk. Transferred files are simply identical copies of the original file that was uploaded or downloaded.

There are reasons, however, why you may want Rumpus to recognize various file types. For uploaded files, it is best to set the type and creator codes of each file received and saved by Rumpus, so that the file can be correctly processed on the server. And when files are served via the Web File Manager, Rumpus needs to send the file content type to the browser so that the browser can correctly interpret and display (or save) the file contents.

Rumpus comes preset to recognize and process dozens of file types, and others can be easily added. In the Rumpus control application, open the "File Types" window to modify or add new types to Rumpus' list of known file types. This list allows Rumpus to set the content type and creator code of uploaded files (based on the filename extension) and to specify the correct content type for Web browsers (again, based on the filename extension). For details, see the help window in the Rumpus control application.

ASCII/Binary File Transfers

The FTP protocol specifies 2 distinct types of file transfer, ASCII and Binary. Binary transfers are much more common, and are used to transfer an exact duplicate of a file either to or from the server. ASCII mode transfers, by contrast, are used to send plain 7-bit ASCII files with platform specific line endings. Some operating systems use a carriage return to denote the end of a line of text (such as the Mac OS), while others use a line feed (Unix, for example). Windows, notably, uses both. When a file is sent in ASCII mode, the client and server convert line endings so that, regardless of the platform of the client and server, the file transfer will result in a copy that has the correct line endings for the local system. Since the traditional Mac OS line ending is different from that of Unix (and therefore, OS X), Rumpus gives you the choice of how line endings should be saved. See the "Save ASCII Uploads With Unix Line Endings" checkbox on the "Encoding" tab of the "FTP Settings" window for details.

Resource Forks

There is one other, very significant, unique aspect of traditional Mac OS files, called the resource fork. While files on most operating systems are made up strictly of a single series of numbers, Mac files contain 3 distinct components, the data fork (which is essentially analogous to the complete file on other operating systems), the resource fork, and the Finder information. The Finder info is described

briefly above, and includes a series of values used by the Mac OS Finder to describe how the file should be displayed and processed by the system. The resource fork is more complex, and is essentially a database of information used by applications in handling the file. For example, a JPEG image may have the raw JPEG data stored in it's data fork, but thumbnail images in various sizes stored in it's resource fork for easy access and processing. Another example is application files, which store GUI window representations, icons and other graphics, and even executable code in various resources within the resource fork of the file.

When a generic, non-platform-specific file is transferred via FTP, only the file data is transferred. A Mac file that is transferred to a non-Mac computer will lose it's resource fork, because the other computer has no concept of what a resource fork is. When a Mac file is transferred to a Mac server, it should retain it's resource fork and Finder info, but FTP provides no method of sending this additional information.

The solution is to encode the Mac file, including it's data fork, resource fork and Finder info into a single data stream for transfer across the Internet. The standard encoding method is called MacBinary, and allows Mac files transferred across the Internet (or any single stream mechanism) to retain their "Mac-ness".

Rumpus fully supports MacBinary, and will automatically encode and decode files on the fly, as will popular FTP clients like Fetch and Interarchy. When files are transferred using the MacBinary encoding, the file resource fork and Finder information will therefore be retained, making a true and exact copy of the original Macintosh file. Note, however, that the client must specify that MacBinary encoding should be used. If files don't appear to be transferred correctly, check to make sure that the client is set to send the files using MacBinary mode transfers. This is a common upload option in most Macintosh FTP client applications.

Conclusion

When files transfers result in corrupt or unusable files, the likely cause is that the system simply doesn't know how to process the file. In this case, contact the client and ask that they send files with MacBinary encoding specified, and confirm that the file has a name with an appropriate extension. Also, check the Rumpus "File Types" list and make sure that an entry exists for the file type being uploaded.

When files served by the Rumpus Web File Manager are displayed or stored incorrectly by Web browsers, you will also need to check the "File Types" list and confirm that a suitable entry exists. In this case, the "content-type" is of primary importance.

As always, if you continue to have trouble, please contact Maxum technical support at "support@maxum.com". Even if this article hasn't lead to a specific resolution to your problem, a basic understanding of the issues involved will make it easier for us to help you find a solution.