

Title: Case Sensitivity versus Case Preservation in CIFS Server (Samba)

Author: Don McCall, Senior Support Engineer, Hewlett-Packard GSE

Introduction

The differences between PC and UNIX file systems are legion: how big they can be, what attributes they support, what characters are legal for file and directory names, etc. Samba does a truly heroic job of handling these differences. One way it does that is by recognizing and handling the difference between an operating system that is case sensitive (Unix), and one that simply preserves case (Microsoft Windows9x, WinNT, and Windows2000). In this article I'd like to examine this difference.

Let us take for our study four filenames:

FILEONE.TXT
filetwo.txt
Filethree.txt
FiLeFoUr.TxT

The first file (FILEONE.TXT) is all uppercase.

The second file (filetwo.txt) is all lowercase.

The third file is a special 'case' whose first letter ONLY is uppercase. The fourth file (FiLeFoUr.TxT) is a truly 'mixed' case, representative of the myriad case permutations a file name could take on.

In Unix, if we used the touch command to create these four filenames, we would get exactly what we requested, i.e., a set of files in the directory whose names exactly matched the case we typed in, as we see here from the output of the UNIX 'll' command:

```
· rw-r-r-- 1 root  sys      0 Dec 19 15:17 FILEONE.TXT
· rw-r-r-- 1 root  sys      0 Dec 19 15:18 FiLeFoUr.TxT
· rw-r-r-- 1 root  sys      0 Dec 19 15:18 Filethree.txt
· rw-r-r-- 1 root  sys      0 Dec 19 15:18 filetwo.txt
```

Note that the Unix command 'll' doesn't alter this presentation – the filenames are shown just as they are represented in the file system. If this statement seems obvious to you, please keep it in mind for when we discuss Windows Explorer, and its graphic representation of filenames in directories, or as they are referred to there, folders.

Now let's move on to another operating system, Windows. For this discussion, we will be using a Windows XP client, with a NTFS file system. Let us again create our four example files, this time using the Windows Explorer graphical user interface (GUI), and the pull down menu File/New.../Text Document .

We type in (over the default name "New Text document.txt" that Windows NT Explorer favors for text documents) "FILEONE.TXT" and hit return. In older versions of Windows, for instance Windows NT Workstation - the file we just created would appear as "Fileone.txt", NOT "FILEONE.TXT". This is just a Windows NT Explorer visual aesthetics change however; if you go into the 'command prompt' window, and actually do a 'dir' in that directory, you would see that

the file DOES actually show up as all uppercase - FILEONE.TXT, just as you typed it in. Windows XP does not have this peculiarity, and FILEONE.TXT shows up all in uppercase both in the GUI and the command prompt window.

The command prompt window is a useful tool to look at what is going on UNDERNEATH the GUI interface that Windows shows by default. We will use it again in our study.

Back again in the Windows Explorer GUI, we create our next example file, typing in "filetwo.txt" all lowercase and hit return. Explorer actually shows us the filename just as we typed it, "filetwo.txt". And the 'dir' command in the command prompt window agrees.

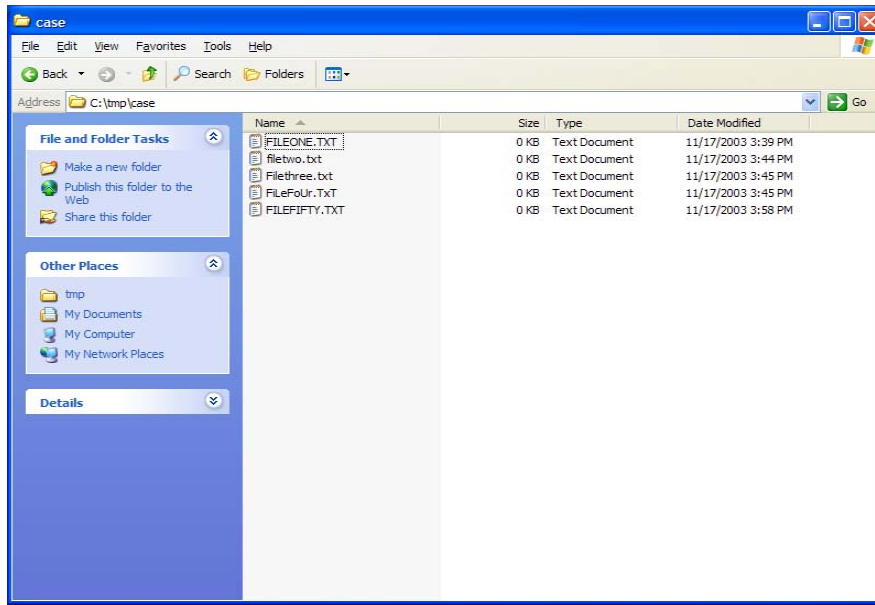
Let's move on to our third example file, "Filethree.txt". This filename differs from the others in our study in that this is not a standard 8.3 filename - (8.3 being a curiosity from the days when Microsoft operating systems did not have a concept of 'long file names', and all files were required to be no more than 8 characters long, plus an optional .xxx, where xxx represented some file 'type' meaningful to specific applications). This filename contains NINE letters before the "." Let's see how Explorer handles this:

We type in "Filethree.txt". Explorer shows us just what we would expect, "Filethree.txt". Does the command prompt 'dir' command agree? Indeed it does. [Apology to all you MS wizards out there who know that I have glossed over the fact that Windows XP has ALSO created a special '8.3' name to go along with this filename, which you can view from the command prompt by using the special 'dir /X option - but that's a topic for another paper.]

Now our final example file - a ReAlly mixed up case file. We again use Windows XP Explorer, and again, both the Explorer and the command prompt representation of the file is just as we would expect, "FiLeFoUr.TxT

So, all in all, not a bad track record; except for the case of older Windows NT Explorer versions choosing to display filenames all in "UPPER" case as the more visually pleasing "Upper" case, we seem to get what we ask for.

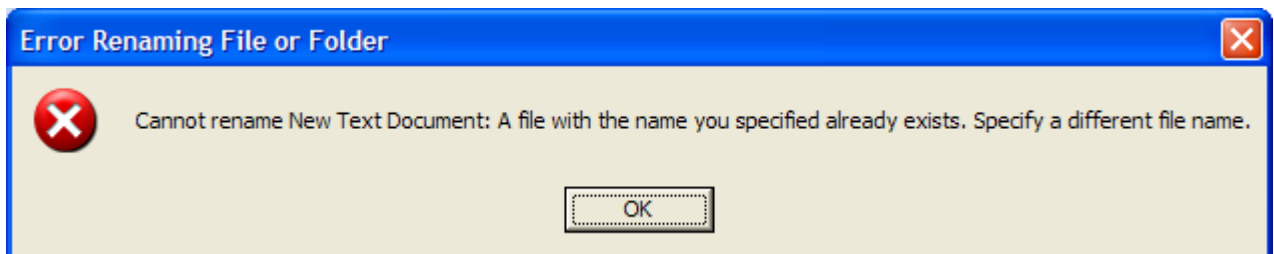
Speaking of filenames in all UPPER case; we forgot that Windows treats files that conform to the "8.3" specification differently than it does to 'long' filenames. Let's see what happens if we create an all UPPER case filename that does NOT conform to the "8.3" specification. We create (using Explorer) a file named "FILEFIFTY.TXT" (note the 9 characters before the ".") and in this case, both the older Windows NT Explorer and the Windows XP GUI are in complete agreement - the file is represented just as we asked, "FILEFIFTY.TXT".



Windows is Case Preserving, not Case Sensitive

Ok, I've laid all this groundwork not to disparage Windows case handling abilities, but to illustrate a point. Windows 'preserves' case. IT is NOT 'case sensitive'. Internally, it matters not at all to Windows XP whether you named the file FILE.txt, File.txt, file.txt, or FiLe.TxT. They're all the same name to Windows. While they are displayed (for the most part, as we have discussed above) in whatever mixed case you type them in with, they are all stored internally in the same case.

To verify this, we have a directory with the file FILEFIFTY.TXT from our previous discussion. If we try to create (using Explorer) a file named FileFifty.txt in this same directory, we will FAIL. Explorer responds that "A file with the name you specified already exists. Specify a different filename":



But could that just be a problem with the 'GUI' interface? Lets attempt the same thing in the command prompt window. If we use the 'dir' command to list a specific file, for instance:

```
dir FILEFIFTY.TXT
```

you will get the listing for FILEFIFTY.TXT
But if you type in

```
dir FileFifty.txt
```

or

```
dir FILEFifty.TXT
```

or

dir filefifty.txt
or ...

Well, you can work out all the permutations yourself but the bottom line is that ALL of these permutations will return the entry for the ONE file FILEFIFTY.TXT.

So, as you can see the Windows file system allows for only ONE case permuted filename to exist at a time in a single directory. Unlike Unix, where you can have in the same directory the files named:

fileone.txt
Fileone.txt
FILEONE.txt
FiLeOnE.TxT

all existing at once, in Windows this is impossibility. That's the difference between a file system that PRESERVES case, and one that is actually CASE SENSITIVE.

And that brings me to what I really want to discuss, which is how CIFS Server (Samba) deals with this.

As you would expect from an application that was grown to bridge the gap between the Unix and Windows worlds, it is very flexible. This is both good and bad - with flexibility comes responsibility, and sometimes not a little confusion. There are four configuration options that Samba provides to allow one to define its behavior when dealing with matters of 'case':

preserve case = (yes/no)
short preserve case = (yes/no)
default case = (upper/lower)
case sensitive = (yes/no)

Case Preservation In CIFS Server (Samba)

The first three options define, in essence, how a filename will be written to the Unix file system underneath Samba. These options loosely correspond to how Samba will PRESERVE case. "preserve case" and "short preserve case" both do the same thing; the first in the case of NON 8.3 filenames, and the second specifically for filenames conforming to the older 8.3 DOS file naming conventions. If these options are set to "yes" (the default), then a file will be saved with the case as it is presented by the client. That is, if you create a file on a Samba share from Windows XP explorer with the name "FiLeNaMe.TxT", a Unix 'll' of the file will show that its name is indeed "FiLeNaMe.TxT".

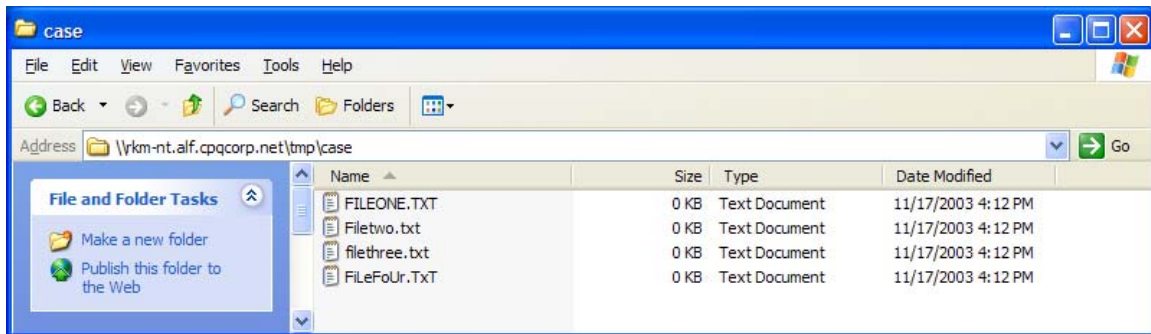
The "default case" option defines how a filename will be saved if either of the 'preserve case' options are set to "no". If "default case = lower" (which is the default) then the effect is the same as if you had "preserve case = yes". HOWEVER, if "default case = upper" then when "preserve case = no" a file will always be saved using all UPPER case letters, regardless of how the client 'presents' it. That is, the file we create in explorer named "FiLeNaMe.TxT", will actually be saved as "FILENAME.TXT" when we look at it with the Unix 'll' command.

To illustrate, let's go back to our four 'example' files, and look at what actually happens. Let's take the Samba defaults first:

```
preserve case = yes
short preserve case = yes
default case = lower
```

Using Windows XP Explorer interface, lets create our four files on a Samba share. In Explorer we type in the four file names:

```
FILEONE.TXT
Filetwo.txt
filethree.txt
FiLeFoUr.TxT
```



A Unix 'll' command will show us:

```
· rwxr-r-- 1 ddmc users 0 Dec 19 16:40 FILEONE.TXT
· rwxr-r-- 1 ddmc users 0 Dec 19 16:41 FiLeFoUr.TxT
· rwxr-r-- 1 ddmc users 0 Dec 19 16:40 Filetwo.txt
· rwxr-r-- 1 ddmc users 0 Dec 19 16:41 filethree.txt
```

Note that with the CIFS Server (SAMBA) defaults in place, we get the results we would expect.

Now, let's change just ONE thing at a time, and observe the results.

```
preserve case = no <change>
short preserve case = yes
default case = lower
```

We remove and recreate the four files in the same manner as above, and our trusty Unix 'll' command shows us:

```
· rwxr-r-- 1 ddmc users 0 Dec 19 16:45 filefour.txt
· rwxr-r-- 1 ddmc users 0 Dec 19 16:45 fileone.txt
· rwxr-r-- 1 ddmc users 0 Dec 19 16:45 filethree.txt
· rwxr-r-- 1 ddmc users 0 Dec 19 16:45 filetwo.txt
```

Interesting! It is understandable that 'filethree.txt' is being converted to lower case; after all it is not an 8.3 filename, so it should fall under the auspices of the 'preserve case = no' option. But what about the other three? They are all good conformant 8.3 filenames. Why was case NOT preserved? Apparently, 'short preserve case' is dependent on 'preserve case'. That is to say, in order for 'short preserve case = yes' to work, 'preserve case = yes' must be set.

Ok, let's move on to the case where we specify BOTH
preserve case = yes
short preserve case = no
default case = lower

In explorer, when we create (with the default name) New Text Document.txt, the UNIX ll command shows us:

```
· rwxr-r--  1 ddmcc  users          0 Dec 19 17:10 New Text Document.txt
```

You can see that with this NON 8.3 conformant filename, case was preserved.

When we create FileWW.txt in Explorer it becomes fileww.txt, as we would expect:

```
· rwxr-r--  1 ddmcc  users          0 Dec 19 17:11 fileww.txt
```

Since this file name conforms to 8.3 conventions (i.e. is a 'short' file name), the 'short preserve case' option is used, and the 'default case' of lower is used to create the filename. Thus this 'short' file name has NOT had it's case preserved on the Unix system.

Now another test:

```
preserve case = yes  
short preserve case = no  
default case = upper
```

With these options, creating "HeresALongFileName.TxT" in Explorer, yields "HeresALongFileName.TxT" in our Unix listing. Case has been preserved. Creating "ShrtFiLe.TxT" in Explorer yields "SHRTFILE.TXT" in our Unix listing. Case has NOT been preserved:

```
# ll  
total 0  
-rw-rw-rw-  1 root    sys          0 Nov 17 16:24 HeresALongFileName.TxT  
-rw-rw-rw-  1 root    sys          0 Nov 17 16:24 SHRTFILE.TXT
```

This is good - we told Samba to preserve case for non 8.3 filenames, and it did. We told samba to use default case = upper for 8.3 filenames, and indeed, it converted our mixed case "ShrtFiLe.TxT" to the all uppercase "SHRTFILE.TXT" file name.

I hope this may go a long way towards explaining how Samba 'preserves' case or not, depending on some pretty flexible configuration options.

Case Sensitivity in CIFS Server (Samba)

But how is this going to affect our clients, when they start LOOKING for files? Well, remember that all Windows operating systems (WinNT, Win98, Win2k, etc) PRESERVE case but are not case SENSITIVE. One result of this is a somewhat laissez-faire attitude in applications and the

operating system itself in trying to FIND a file of a specific name. A program could, for instance CREATE a file named FileName.TXT, and then when it next opened the file, could refer to it as filename.txt, (note the case difference) and still expect to find it.

This presents certain problems when you sit atop a TRUE case sensitive operating system like Unix. FileName.TXT and filename.txt could BOTH be present in the same directory; which one does the client really want?

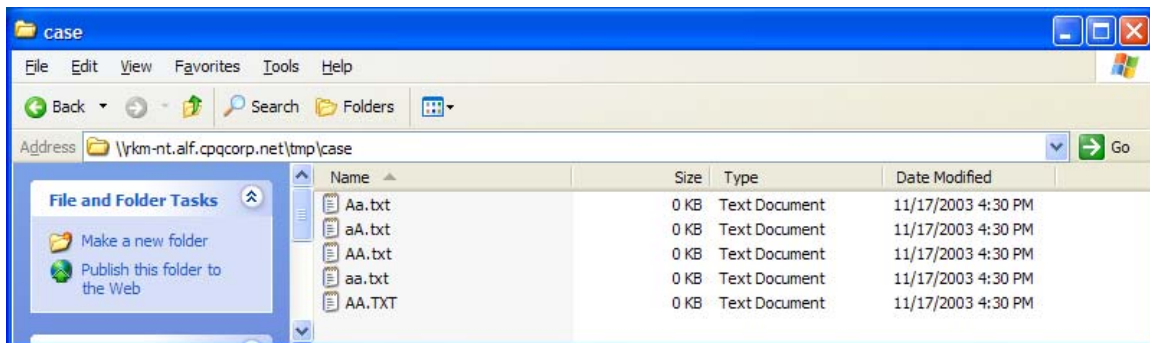
The answer is found in the 'case sensitive' configuration parameter. The other parameters defined how we SAVE filenames when we create files through Samba. This parameter determines the rules we follow when we try to RESOLVE a filename given us by a client, and it the crux of whether that client gets WHAT IT EXPECTS or not.

By default, 'case sensitive' = no. This means that no matter HOW the windows operating system or application refers to a filename case wise , it will match the first filename that we stumble across that matches the requested filename REGARDLESS of case.

For instance, consider that you have five files in your samba directory:

Aa.txt (contains the ascii characters <Aa>)
aA.txt (contains the ascii characters <aA>)
AA.txt (contains the ascii characters <AA>)
aa.txt (contains the ascii characters <aa>)
AA.TXT (contains the ascii characters <AA.TXT>)

When you look at the directory via Windows Explorer, all five files will appear, with the unfortunate confusion that there will be TWO entries for Aa.txt (remember the older Windows NT Explorer would 'translate' an all UPPERCASE 8.3 name as if only the first character in the name were capitalized – here we use WinXP so this is not an issue).



Unfortunately, with 'case sensitive' = no, you have no way to tell Samba WHICH file you want to open; it doesn't matter which file you click on, samba is going to match it to the first file that has a caseless match for the characters you provide it. In our example above, for instance, we will ALWAYS get the text "AA.TXT" when we double click on ANY of the above filenames in the Windows Explorer window, because this version of the name is canonically first in the list. This is PROBABLY NOT the behavior you would desire.

Fortunately, changing 'case sensitive' to yes will actually do what you expect; Windows 'preserves' case, so when it sends the smb request to 'open' the file, it will send the name with the correct case. That is, the file aA.txt will be requested by the name aA.txt. And with samba operating in 'case sensitive' mode, it will look for an exact match, respecting case, for that filename, and return aA.txt to you. Similarly, if you double click on the Aa.txt filename, you will

get the Aa.txt version of the file (the one containing the characters <Aa>), NOT the AA.TXT version.

I trust you can see the potential for confusion and error here, but let me mention a couple of specific issues you may run into when you set case sensitive = yes:

1. Sloppy programming: If your program CREATES a file named Initialize.INI, it had better always try to open it as Initialize.INI. Not initialize.ini. Not INITIALIZE.INI. This is not a requirement for programs written to work on native Windows file systems, and so you may come across programs which mix case, and will fail when run on files saved to Samba shares with case sensitive = yes enabled.

2. Dos and WfW clients will probably NOT be able to run programs using files created by later clients (Win9x, NT) with the same program. Let me illustrate by example.

Say we have an application that has a version that runs on WfW, DOS, WIN98 and WinNT. The WIN98 and WinNT versions create and use a file named Startup.dat. The WfW and DOS use the same file, but of course they must refer to it as STARTUP.DAT (Dos and WfW understand 8.3 uppercase filenames only).

If you install this program on your various PC's, and decide that you want them to ALL use the same initialization file (Startup.dat), you might have the installation program locate that file on a Samba share accessible to all clients. But this won't work - if the file is named Startup.dat, the WfW and Dos clients won't see it. if the file is named STARTUP.DAT, then the Win98/NT clients won't see it.

Conclusion

A simple rule of thumb emerges: If the files on your share are going to be mainly created and accessed by Windows clients, leave the defaults alone. This will give you the behavior most similar to what an NT Client would expect from a 'real' Windows NT server. If you HAVE to have multiple filenames in the same directory that differ only by case (perhaps you have a UNIX application seeding these samba shares with files that will later be accessed by a Windows program for processing, for example) , change the 'case sensitive' option to yes. But recognize that this may cause some inexplicable behavior on the part of Windows client applications accessing that directory. Remember that these options can be applied on a share by share basis, and try to restrict your use of non default values ONLY to shares with files and applications that require this.