What is the problem with Bare Metal Restore Using Windows/SBS 2000 Restore from Backup?

Draft V 1.0 7/12/02

This is not an official MS technote, heck, it not officially anything. This document has not been reviewed or approved by MS or anyone. This may have some controversial ideas expressed and may even harbor outright errors. This is a compilation of information to use as you see fit, not a substitute for smart management of your own situations.

If you have recommendations, corrections, or additions, please be kind enough to forward them to me in an HTML format or plain text. Include any technotes you have reason to believe would help. I'm not really looking to start a newspaper here or become an investigative reporter. I'm trying to avoid some of us posting the exact same stuff over and over again.

SBSideas@cfisolutions.com

Overview

The short answer: A restore from backup that uses a file by files restore results in some folders/file paths being different that when they were backed up originally to tape. If the change in the new Short Filename (SFN) assigned is coincidentally related to a registered program that references the original SFN assigned in the registry, the application is broken now. This happens on every installation of SBS 2000. Only a partition level copy will preserve the SFNs because only copying the partition table with the SFNs embedded will result in the identical system following the restore, and no broken SBS applications.

My initial review indicates that on all default manner SBS installations, the following application folders with registered applications associated are broken:

Microsoft Backoffice Integration
Microsoft ISA Server
Microsoft Windows Media Player
Microsoft SQL Server

If any portion of Microsoft Office 2000 is installed on the SBS server, it will become broken. This is really by all summary considerations, nothing trivial, this is a major problem!

The entire problem results from the restore of the previous file structure according to Alphanumeric order by folder, instead of the historical based “natural order of installation” since the initial configuration of the system as a whole. This is a complex problem. It is not unique to SBS 2000, it is present in Windows 2000, and in fact the root issues originated in Windows 95 design features for Long Filename generation to Short Filenames. The problem is brought to light more significantly by the more recent change in SFN design starting with Windows 2000, and due to the increased use of Long Filenames in the actual Application Programs that install in the “Programs Files” folder, among other places.

Documented References to this Problem

For a detailed explanation in plain language, review the remainder of this document.

Here's an MS document that covers this concisely:

Q240240 - Programs Do Not Work After Restoring Computer With Backup

In a curious twist of fate, the revelations of Q240240 lead to the logical conclusion that the following KB is both rudely inadequate, and woefully understating a serious problem that would be encountered with virtually any use of it as it reads as of the time this document is now being written. Stated differently, the following KB will lead you to believe that you can restore a server to new hardware using a file-by-file restore from a different machine, and that's going to fail for all the reasons raised by the topic of the document you are now reading:

How to Move a Windows 2000 Installation to Different **Hardware** (Q249694)

Veritas covered this issue long ago, but in the most peculiar way.

There has been a well acknowledge article produced by Veritas, maker of Backup Exec as well as the NT Backup program that ships within Windows 2000, which recommends against using their Intelligent Disaster Recovery (IDR) feature with SBS. This article was released not long after SBS 2000 began shipping, but on the surface, the explanation it provided left the reasonable impression that Veritas IDR didn't work. In fact, Veritas was being "generous" to say only in a limited manner that you should "restore your SBS 2000 server" in the normal manner recommended by the manufacturer. While this was a simple way to document their problem and resolution, it really still concealed the point that there is in fact no way to use NT Backup in a restore method that doesn't have exactly the same faults as the Veritas IDR. In that context, the IDR is as effective as NT Backup or native Backup Exec restore methods, or stated differently, all of these produce equally broken restores for the same reason. The break is caused by issues due to the design of Windows 2000 handling of SFNs during a file-by-file restore, nothing more. Whether or not Veritas has had any roll in omitting attention to or propagating the flawed features related to file-by-file restore model they themselves wrote the embedded modules for on behalf of Microsoft, this remains a determination someone else will need to arrive upon.

Here is the link to the document, and the entire document as of this moment is inserted below:

http://seer.support.veritas.com/docs/235745.htm

After an Intelligent Disaster Recovery of a Windows 2000 Small Business Server system is performed, COM objects are failing with initialization failures such as "Snap in failed to initialize" when the SBS Administrators console is launched.

Symptom:

After an Intelligent Disaster Recovery of a Windows 2000 Small Business Server system is performed, COM objects are failing with initialization failures such as "Snap in failed to initialize" when the SBS Administrators console is launched.

Solution:

Dear Valued VERITAS Customer,

VERITAS has recently discovered an issue* that could prevent successful Disaster Recovery operations of a Microsoft Small Business Server 2000 system using the VERITAS Intelligent Disaster Recovery Option. This is not to be construed that your data is at risk, only the ability to automate the recovery of the Microsoft Small Business Server 2000 operating system during an Intelligent Disaster Recovery operation. VERITAS is advising users NOT to use the Backup Exec <IDR> function for automated recovery of systems running Microsoft's Small Business Server 2000 v5.0. (Backup Exec's <IDR> operation for automated recovery of a system running Microsoft SBS 4.5 remains unaffected).

In Microsoft Small Business Server 2000, this issue is caused by a combination of:

- directory short names being used in the Registry for COM object registration
- non-unique generation of directory short names
- · installation of all 2000 family products in the "~\Program Files" directory

This combination results in COM object failures because the directory short name locations that are registered for specific COM object DLLs are not the same after a disaster recovery. This is because the directory creation sequence during disaster recovery is not identical to the original directory creation sequence, which will then cause the directory short name generation to be different from the directory short name registered in the restored registry.

This issue appears to be most prevalent during a disaster recovery of an SBS system because of the numerous Microsoft components that are installed in the "~\Program Files" directory as "Microsoft <product name>", however, this can affect non-SBS system recovery as well.

This issue does NOT affect the ability to fully protect or restore a system running Microsoft Small Business Server 2000 using Backup Exec. A manual disaster recovery of the Microsoft Small Business Server 2000 including Exchange 2000 (if applicable) and SQL 2000 (if applicable) can be performed by re-installing the SBS 2000 operating system including Exchange 2000 (if applicable) , SQL 2000 (if applicable), and performing a manual disaster recovery.

For complete step-by-step manual disaster recovery procedures, please see the related articles at the bottom of this document or review these chapters in the Backup Exec for Windows NT and Windows 2000 Administrator's Guide.

- Backup Exec Administrator's Guide Chapter 9 for Manual Disaster Recovery Procedures for Windows 2000 Systems
- Backup Exec Agent for Exchange Server (if applicable)
- · **Backup Exec Agent for SQL Server (if applicable)**

*** This issue is under review by Microsoft and VERITAS , and when resolution becomes available, VERITAS will modify this TechAlert to include the resolution and notice will be sent to all customers who have subscribed to the VERITAS product notification service.**

Why does this SFN problem Occur?

When you look at a drive partition, say Drive C:, you could open a command prompt at the root of C, and then run the following command:

Dir C: /s

That will run from the root of C: down through all subdirectories and show you all files, folders and subfolders that exist (unless they have the hidden attribute enabled, but that's not significant to the point I'm making).

What you will normally see is the folder tree and files displayed in alphanumeric sequence, as they exist at this point. Literally, you see them sorted in alphanumeric order. You could instead run the command:

Dir C: /s /oe

That will sort the display by file extension. No particular value in that, but it demonstrates that the order I sort the tree isn't really going to affect the fact that I still see all the same files listed all the way down the tree from the root of C:

What happens then if you run this command:

Dir C: /s /od

....is you see the files in the order of "last date modified", again for all files in the tree.

The problem is, there isn't a record or attribute maintained that indicates the "order in which the files arrived into this folder", what we have previously stated as the "natural order of creation" in the folder. This is simply not recorded _and_ it can't be reliably deconstructed either!

It's not just that you would need to know every file and the arrival sequence into the folder, you would also need to know any files that exited from the folder that _might_ affect the SFN naming both by the order in which it arrived as well as by the timing of when it left!

This is really messy to explain. A brief explanation of how SFN is generated will help...some.

SFN is the Short Filename that meets the 8.3 convention for a filename originally established in DOS perspective. Windows 95 established the first MS OS that allowed a longer filename than the so-called 8.3 convention. Typically, we would represent a template of 8.3 as follows:

filename.ext

8-characters of filename, 3 characters of extension to result in an 8.3 filename template.

Long Filename (LFN) has always used a familiar technique to abbreviate an LFN in a SFN compatible format. Most all of us paying any attention quickly committed to memory the manner that SFN abbreviates LFN. In almost all cases, it was the first 6-characters plus ~1, then ~2, then next ~3 as the files were created. This assumes the extensions are all the same.

Microsoft determined that with Windows 2000, there were performance motivations to alter this plan just slightly, however, I need to emphasize that even in altering the SFN convention, the problem we now have existed before, and it exists still because of how SFN works, they only changed a subtle point with a subtle alteration on the result that still leaves the system broken.

BTW, you can reveal and illustrate this at home in the comfort of your own crying chair by doing the following things:

1. You have to be at a Windows 2000 computer
2. You add to your User or System Environment the variable "DIRCMD=/x"

    This will let you open a command prompt windows and do "DIR" commands that show parallel columns of the SFN and LFN as you work through these steps.

3. Create a new empty folder by any name, and using the very same the LFN names I show, add them into a folder by either using Explorer | Create new Textfile...and use that name, or at a cmd prompt you can simply use: echo > LongFileName1.txt

    Repeat that to create the other names indicated by adjusting the number accordingly in the filename.

4. When it comes time to simulate the delete and restore of the files, simply create another new folder on the same partition, and copy the files as a group or by whatever method (COPY at a CMD prompt works) so that they are added into a new folder that is empty before the copy process is started.

Windows 2000 uses the same convention generally for the first 4 files in a folder that have otherwise identical 8.3 abbreviated names. Assume that we have a filename like LongFileName1.txt, and then follow it with LongFileName2.txt, LongFileName3.txt, LongFileName4.txt and LongFileName5.txt, etc. The SFNs generated produce:

    LONGFI~1.TXT
    LONGFI~2.TXT

LONGFI~3.TXT
LONGFI~4.TXT
LO0CAB~1.TXT
LO00BB~1.TXT

You are probably surprised about the last two! You probably didn't realize this change had been made in Windows 2000. After the 4th file, Windows 2000 goes from:

xxxxx~#.ext

...where the # is 1, 2, 3 or 4, but then after instead to...

xxABCD~1.ext

...where the ABCD are generated at the time the file is created, and always followed by ~1, at least as far as I've been able to determine is the way the algorithm works. It is not a random algorithm character generation. You should find that if you create the file I just illustrated at your own computer in your own empty folder, they should look just like that, including the 5th and 6th filenames matching exactly. I don't know what the algorithm is, but it's supposedly predictable and repeatable. Some additional technique is applied to deal with non-legal characters such as spaces, and punctuation characters that are legal in LFN, but not in SFN.

We aren't in trouble yet, we are about to see what gets us in trouble.

Suppose I have the following files existing in the folder in question:

LONGFI~1.TXT	LongFileName1.txt
LONGFI~2.TXT	LongFileName2.txt
LONGFI~3.TXT	LongFileName3.txt
LONGFI~4.TXT	LongFileName4.txt
LO0CAB~1.TXT	LongFileName5.txt
LO00BB~1.TXT	LongFileName6.txt

Now, I delete the first and third files from the folder. What remains is:

LONGFI~2.TXT	LongFileName2.txt
LONGFI~4.TXT	LongFileName4.txt
LO0CAB~1.TXT	LongFileName5.txt
LO00BB~1.TXT	LongFileName6.txt

Now, I again recreate one additional filename as LongFileName3.txt, and keep in mind, this is literally the name of the file I'm creating, here is the result of looking at the files that will be in the folder now:

LONGFI~1.TXT	LongFileName3.txt

LONGFI~2.TXT	LongFileName2.txt
LONGFI~4.TXT	LongFileName4.txt
LO0CAB~1.TXT	LongFileName5.txt
LO00BB~1.TXT	LongFileName6.txt

Notice that LongFileName3.txt adopted the first available SFN position? Two important points: The files that were not altered still have the same SFN. However, I illustrated that by sort order of SFN, with the LFN shown on the right, if you were to display this instead by LFN order (which is what normally shows with the DIR command anyway), you see this:

LONGFI~2.TXT	LongFileName2.txt
LONGFI~1.TXT	LongFileName3.txt
LONGFI~4.TXT	LongFileName4.txt
LO0CAB~1.TXT	LongFileName5.txt
LO00BB~1.TXT	LongFileName6.txt

You may now see this coming. Okay, here's where the bomb gets dropped. Now do a tape backup of this folder, delete the folder, then restore these files back from tape file by file, and they will be recreated in the folder in LFN alphanumeric order, but watch what happens to the SFN for the entire set:

LONGFI~1.TXT	LongFileName2.txt
LONGFI~2.TXT	LongFileName3.txt
LONGFI~3.TXT	LongFileName4.txt
LONGFI~4.TXT	LongFileName5.txt
LO00BB~1.TXT	LongFileName6.txt

Yeowww!!! Each of the first four filenames restored in Alphanumeric order now has a totally different SFN assigned, yet the 5th filename is the same as before, even though it's actually created as the 5th file in the folder now, whereas it was originally the 6th file into the folder!

An Analysis of What Just Happened

Okay, class. What did we just learn?

1. Files are recreated when they restore based upon the LFN name because that's all that is stored and processed by the file by file create process, this because the SFN is generated dynamically based upon the current contents of the folder.
2. SFNs resolve collisions based upon using the first available SFN "slot", including taking an SFN previously occupied by a file that was created, and has later been deleted.
3. Natural create order for files would need to include an understanding of not only the SFN and LFN of the file in one instance, but also for the exact "slot" conditions available at the time each collision is being resolved at the time each new file is being introduced to a folder where a collision is going to occur.
4. It's not simply a case of Alphanumeric order or date order for the files that exist at the time you make a file by file backup that will impact the SFNs, it the <> difference between what results from the "point in time" natural creation order which you then compare to the "at time of file by file restore" creation order by alphanumeric LFN at that time.
5. Files, which never were removed from a folder, but arrived other than first where an SFN collision was resolved, can still have a different or same SFN assigned, simply depending upon if they are recreated in a "slot" order that establishes the same "slot" assignment condition for the first 4 collision slots.
6. Files which by any means arrive in any create order, but after all of the first four collision slots are full will be created with an xxABCD~1.ext SFN that will be the same regardless of the order of creation order beyond 4th slot.
7. Deleting any file/folder in one of the first four SFN collision slots can potentially impact the SFNs for all other files/folders that would be generated at a later restore point. This impact would be represented by its position "n", where $1 < n < 5$, and the number of files that would be affected is equal to: $5 - n$.

Summary Conclusions

1. Unless you know both the SFN and the LFN assigned to a file by file backup, you have no possible way to reliably determine after a file by file restore if you have restored the files back into in a known good/matching condition or not whether or not you restore to empty folders, much less folders with some existing content that results in SFN naming collisions

2. SFN generated names are generally of no consequence for data files because most programs don't refer to their own data files by SFN because it's unreliable, but also not a condition you would meet expect in poor programming designs that don't seem to be prevalent.

3. SFN generated names are critically dangerous to breaking applications when those applications store the SFN rather than LFN *path* in the registry!!!

4. For a Windows 2000 computer first installed from scratch, you can evaluate the most significant exposure of SFN path issues by searching your registry for the following four txt items:

   ~1
   ~2
   ~3
   ~4

   The reason this evaluates the problem is that in all SFN path storage, only the ~1 through ~4 string is consistently present, since after ~4 being assigned to the 4th slot file, all further files are assigned ~1 as the 7th and 8th characters.

5. For a computer running Windows 2000 which was originally installed with Windows NT, or some other version of OS which allows for an in-place upgrade to Windows 2000, it's possible that other SFN entries could be present that met the previous SFN generation rules that would have created a ~5, ~6 based SFN, and in no case can that SFN be generated in Windows 2000 even by accident.

6. You can't force an SFN to be a certain value by naming the LFN to match the SFN. You can match an SFN with the LFN for that file/folder instance and this would work in matching pre-existing registry entries that had stored the SFN because there's really no way the program reading the registry entry with a ~1 ending on the file could know if that was an SFN or LFN. However, if you attempt to file an SFN slot using the ~1 concept for the LFN, it doesn't behave as expected. Besides, the technique is of little merit unless you knew all the SFNs you needed, and if you knew that, you probably would still have trouble recreating all the matches, and naming them for SFNs is just as likely to also break some other LFN references in some other stored reference!

7. To perform an Authoritative restore an AD Domain Controller, you must restore the original registry, and the entire System State in the process. When you restore the System State, you restore the Software portion of the registry. Therefore, you restore the registry entries that, when present, contain any SFN path entries which are preserved based upon the original installation order of those applications and registry updates, but any files that are restored by a file by file creation process have the SFN generated by real-time alphanumeric order at the time of the restore, nor the time of the backup, and neither would necessarily represent what the original SFN path that was stored in the registry to start with.

8. The only documented and supported techniques to avoid or correct SFN restore problems is to either:
    a) Restore a partition as an image, including the partition table that contains the SFNs assigned.
    b) Restore by any convenient method, then reinstall all applications that are impacted by revised SFN file and folder paths, assuming you are able to determine which applications are dependent upon "time of installation SFN" stored paths held in the registry, or other configuration files that remain static following a file restore (internal references in a .ini file or database format, for instance)

9. In the case of SBS 2000, the likely method of bare metal restore required would be:

        1) Install OS from scratch.
        2) Install your tape backup software.
        3) Restore your disaster recovery compatible backup including AD, programs and data.
        4) Reinstall all applications that are impacted by revised SFNs generated by the difference of alphanumeric creation order as compared to the original "time of creation instances" generated for the SFNs.

Impact upon an SBS 2000 Bare Metal Restore

A reasonable estimate for this could easily mean that if your tape backup requires 4 hrs to complete per night, you should allow for a file-by-file recovery process to involve:

1. Whatever time is required to establish the required hardware conditions to do a restore.
2. 1 hr. to install Windows 2000 from scratch, including the required backup media drivers and backup program application.
3. 4 hrs. to restore from tape.
4. 1-2 hrs. to evaluate and correct by reinstallation of all SFN impacted applications, plus creation of, at a minimum, a new full System State disaster recovery backup, ERD, Repair directory, and any related preparations you would take at that time.

Alternatively, if you have the means to perform "Partition Image" backups (which include a snapshot in time of the partition table which is also restored), you could use that in one of two manners:

1. If the partition backup represents the entire point in time condition that you want to restore, you can simply restore that and you will be 100% whole with no additional steps.
2. If your partition backup represents a time prior to the condition you want to restore, _however_ it represents no substantially different condition of the installed application SFN file/folder conditions, you could use that to create the same "baseline" of SFN condition. Once the historically consistent SFN file/folder tree is back in place, a file-by-file restore which overwrites the existing files/folder on that partition will retain the SFN even as the files are updated to more current contents!

To clarify that last point, which is quite important and valuable, file by file restores that overwrite existing files will retain the SFN which the file to be overwritten has already been assigned. The vast majority of the SFN risk is associated with *folders*, not files, but regardless, if the files/folders of concern are simply being revised, but not supplemented, your overwrite will retain the correct SFNs to match the registered applications and the file by file restore will have an extremely high probability to be correct.

To use this idea of "restore a baseline image", then "update to recent backup with revised file by file restore" as your recovery process, you would need to ensure that you revise your "partition image" backup if you should make a substantial revision to the file/folder path. Examples of this would be, but not limited to: installing, reinstalling or uninstalling an application that uses LFN files or folders. Even if there were errors that were missed by an older partition image, it would most likely limit the scope of the problem dramatically so that very few applications would be impacted by the SFN restore problems.

Appendix

What follows is an excerpt from a Microsoft Technet document related to the Windows NT Resource Kit explaining many disk storage related matters. For a simpler presentation in this context, the link is provided to see the original document in it's entirety, but only the relevant part has been quoted below. It explains a similar, but more technical example like the original one I presented above:

http://www.microsoft.com/TechNet/prodtechnol/ntwrkstn/reskit/diskdesc.asp

Chapter 17 - Disk and File System Basics

[Refer to the section head as follows]

Generating and Viewing Short Filenames

Filenames on Windows NT platforms can be up to 256 characters, and can contain spaces, multiple periods, and special characters that are illegal in MS-DOS filenames. These long filenames use the 16-bit Unicode character set. Windows NT makes it possible to access files with long names from other operating systems by automatically generating an MS-DOS-readable (eight-plus-three) name for each file. This way, files are accessible over a network by computers using the MS-DOS, Windows 3.1*x*, and OS/2 operating systems, as well as by computers using Windows NT and Windows 95 operating systems.

By creating eight-plus-three filenames for files, Windows NT also enables MS-DOS-based and Windows-based 3.*x* applications to recognize and load files that have long filenames. In addition, when an application saves a file on a computer running Windows NT, both the eight-plus-three filename and long filename are retained.

Note Use caution with MS-DOS-based or Windows 3.*x*-based applications when running under Windows NT. With these applications, if you save a file to a temporary file, delete the original file, and rename the temporary file to the original filename, the long filename is lost. Any unique permissions set on that file are also lost.

If the long name of a file or folder contains spaces, be sure to surround the name with quotation marks. For instance, if you have a program called DUMP DISK FILES that you want to run from the Start icon and you enter the name without quotation marks, you will get an error message that says "cannot find the program DUMP or one of its components."

You must also use quotation marks when a path typed at the command line includes spaces, as in the following example:

```
move "c:\This month's reports\*.*" "c:\Last month's reports"
```

Use wildcards such as * and ? carefully in conjunction with the **del** and **copy** command prompt commands. Windows NT searches both long and short filenames for matches to the wildcard combination you specify, which can cause extra files to be deleted or copied.

To copy or move files with case-sensitive long filenames, it is safest to select the files using a mouse in Windows NT Explorer and My Computer. That way, you can clearly identify which files you want to copy or move.

Because both the FAT file system and the NTFS file system use the Unicode character set for their names, there are several illegal characters that MS-DOS cannot read in any filename. To generate a short MS-DOS-readable filename for a file, Windows NT deletes all of these characters from the long filename and removes any spaces. Since an MS-DOS-readable filename can have only one period, Windows NT also removes all extra periods from the filename. Next, Windows NT truncates the filename, if necessary, to six characters and appends a tilde (~) and a number. For example, each nonduplicate filename is appended with **~1**. Duplicate filenames end with **~2**, **~3**, and so on. Filename extensions are truncated to three or fewer characters. Finally, when displaying filenames at the command line, Windows NT translates all characters in the filename and extension to uppercase.

When there are five or more files that would result in duplicate short filenames, Windows NT uses a slightly different method for creating short filenames. For the fifth and subsequent files, Windows NT:

- Uses only the first two letters of the long filename.
- Generates the next four letters of the short filename by mathematically manipulating the remaining letters of the long filename.
- Appends ~1 (or another number, if necessary, to avoid a duplicate filename) to the result.

This method provides substantially improved performance when Windows NT must create short filenames for a large number of files with similar long filenames. Windows NT uses this method to create short filenames for both FAT and NTFS volumes.

For example, these are the long and short filenames for six files that you create in the order test 1 through test 6.

Long filename	Short filename
This is test 1.txt	THISIS~1.TXT
This is test 2.txt	THISIS~2.TXT
This is test 3.txt	THISIS~3.TXT
This is test 4.txt	THISIS~4.TXT
This is test 5.txt	TH0FF9~1.TXT
This is test 6.txt	THFEF5~1.TXT

However, when you create the files in the order shown in this table, you get the following short filenames.

Long filename	Short filename
This is test 2.txt	THISIS~1.TXT
This is test 3.txt	THISIS~2.TXT
This is test 1.txt	THISIS~3.TXT
This is test 4.txt	THISIS~4.TXT
This is test 5.txt	TH0FF9~1.TXT
This is test 6.txt	THFEF5~1.TXT

Windows NT displays the long names for folders and files. You can use Windows NT Explorer and My Computer to see the short name by selecting the file or folder and selecting **Properties** on the **File** menu.

From the command line, to see both the long and short filenames for each file in the folder, type the following command:

```
dir /x
```

Tip To display both long and short filenames automatically when using the **dir** command, use the System option in Control Panel to set the **dircmd** variable to the value **/x**.