From: Bill Veghte
Sent: Tuesday, September 10, 2002 8:42 PM
To: Ben Fathi
Subject: FW: Office 11 and .NET server

Sigh... We can't say we didn't try. Full bore ahead on shell implementation.

Can you give me the versioning 101 lesson and how TOPS and VMS compare since I am ignorant...

Thanks!

-----Original Message-----
From: Jim Allchin
Sent: Monday, September 09, 2002 7:22 PM
To: Bill Veghte
Subject: RE: Office 11 and .NET server

Unfortunately, Steven has a number of very good points. First, we really need versioning -- not timewarp. Second, having the snapshots show up in folders doesn't fit the open paradigm easily. In short, I would give up. It is great that we have the server apps working in concert so we are in good shape there. But, I think the shell will have to do for now.I

The key here is that WE NEED VERSIONING -- a la VMS or better yet TOPS 10/20.

jim

From: Bill Veghte
Sent: Sun, Sep 08, 2002 12:28 PM
To: Jim Allchin

Here is the last go-around. Love your thoughts on if you have a second.

-----Original Message-----
From: Steven Sinofsky
Sent: Friday, September 06, 2002 11:50 AM
To: Ben Fathi
Cc: Antoine Leblond; Bill Veghte
Subject: RE: Office 11 and .NET server

Thanks.

I really think this is all a good reason why there can't be a solid entry point in the app—there is just too much stuff to try to jam into a small area with no user affordances. The shell is really a better solution for customers here

This reminds me of the "search" button paulma made us jam onto the file open dialog in office 95—this was because we had added search to the server (original index server). All the button did was bring up the browser but what everyone wanted was search to be *in* file open. It was pretty clear that we could not do a better job of a search UI within a subset of the screen with no user interaction points—google proves this today. I think the same challenge is here—we can't put the shell inside a small window that has no user interface other than right click. This is by design—in DOS we tried to build a whole shell in file open, but not any more. There's no magic answer to the lack of UI affordances in or the modal state of file open—this is by design.

I really feel like we have the optimal experience when there is good shell integration. I just don't know how to expose the feature otherwise since users have to pull the information and those calls are costly to the server as you said.

-----Original Message-----
From: Ben Fathi
Sent: Friday, September 06, 2002 11:31 AM
To: Steven Sinofsky
Cc: Antoine Leblond; Bill Veghte
Subject: RE: Office 11 and .NET server

inline

-----Original Message-----
From: Steven Sinofsky
Sent: Friday, September 06, 2002 10:26 AM
To: Ben Fathi
Cc: Antoine Leblond; Bill Veghte
Subject: RE: Office 11 and .NET server

Yep I understand

The challenge is what event would trigger this recovery? With a crash, what we do (simplified) is just set a bit on boot and clear the bit on a clean File Exit. If you don't exit cleanly then we detect this at next boot and present all the recovered files.
Understood I agree that timewarp is user driven. I was just trying to decouple the interface from the file open menu to reduce confusion.
The timewarp scenario is a user driven event that does not count on a fatal exit for our apps—it counts on a fatal mistake from the end-user.

If there is a way of telling if a person has recently connected over SMB to a timewarp server then we could present a File Recover from Server menu that would just spawn the shell user interface
Yes, you can tell if the server/share has timewarp enabled. We also have an API that enumerates all the existing snapshots for a given share and another one (more expensive) that tells you all the unique versions of the file in previous snapshots. Shell uses this to reduce the clutter in the UI by listing only unique versions instead of all available versions.
I think the challenges are:

- This is a user driven event
- The UI is not a "command" but needs to present a ui with more than one step, so integration into a context menu produces a messy nested user experience (file open underneath the timewarp steps)

This is probably the hardest part to solve. I'm no UI expert, but I'm sure you guys come up with a good solution. ☺

- I'm not totally clear on the experience for access over HTTP and how that would work

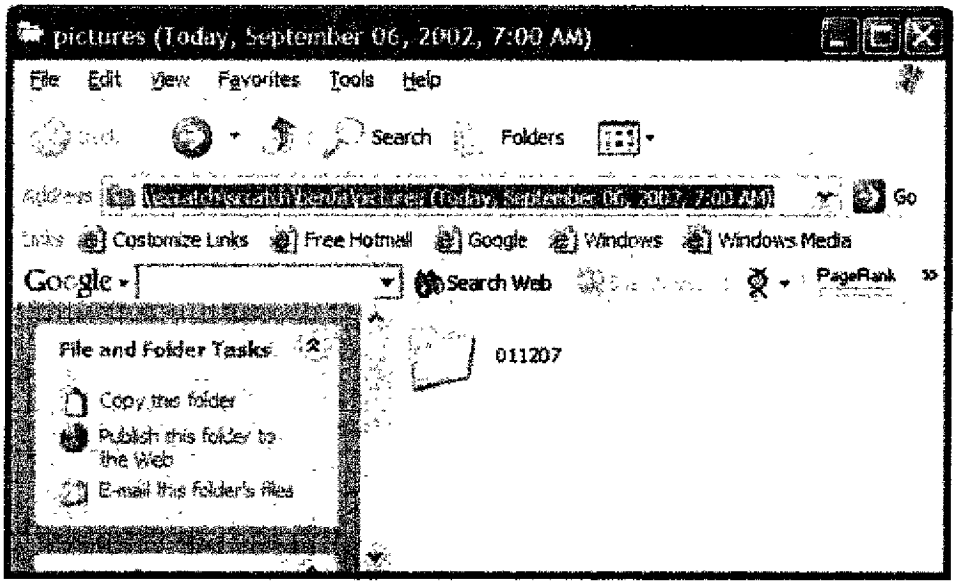
Currently, this only works across SMB.

- In the event of a crash, the recovery information locally is going to be more recent and more predictable (up to the last keystroke).

Correct, but the user doesn't necessarily always want the latest version He might have screwed up the file with his last set of edits and wants to go back to what he had last night.

- Deleted files – how are these presented to the user in file open?

The snapshots are presented as folders, so the user can navigate them just like a regular folder. The shell annotates the title bar and the address bar with the date when the snapshot was created.



What else?

The snapshots are read-only, so you can open them, but can't write to them. The shell offers a couple of options (Copy to, Revert) so the user can easily write the contents back into a writable file in his "My Documents" folder.

Ben

-----Original Message-----
From: Ben Fathi
Sent: Friday, September 06, 2002 9:24 AM
To: Steven Sinofsky
Cc: Antoine Leblond; Bill Veghte
Subject: RE: Office 11 and .NET server

I had a wacky idea in the shower this morning...

Instead of integrating timewarp into the file open menu and thereby confusing end-users, how about using it in a way similar to the recovery menu that comes up if I start up word the first time after it crashes? I'm not sure what this is called, but it's the pane that shows old versions of files and asks if I want to restore one of them. What I'm proposing is a different menu item (say File->Recover) that would bring up a similar menu if Timewarp is enabled on the server. This would

mean we don't touch the file/open menu and the experience would be very similar to the current shell UI.

Ben

-----Original Message-----
From: Steven Sinofsky
Sent: Tuesday, July 09, 2002 9:09 PM
To: Ben Fathi
Subject: RE: Office 11 and .NET server

Right. Servers don't use files, only apps use files. The block level stuff makes total sense for people that don't use files but use blocks.

-----Original Message-----
From: Ben Fathi
Sent: Tuesday, July 09, 2002 9:01 PM
To: Steven Sinofsky; Bill Veghte
Cc: Antoine Leblond
Subject: RE: Office 11 and .NET server

This was originally a server feature. Point-in-time snapshots were meant as an enabling feature for high-end storage management scenarios, hence the comments below about server app cooperation. We've spent a lot of effort coordinating with all the vendors I mentioned below (SAN vendors, backup vendors, server apps). It was only exposed as a client visible feature recently. We've been working with the STS guys for over a year now, but not the rest of the office team.

Ben

-----Original Message-----
From: Steven Sinofsky
Sent: Tuesday, July 09, 2002 8:53 PM
To: Ben Fathi; Bill Veghte
Cc: Antoine Leblond
Subject: RE: Office 11 and .NET server

I didn't even know about the feature until billv asked about file open a few weeks ago. I'm not sure anyone else did either. Perhaps there is a breakdown in communicating the opportunities for the apps, at least to Office?

We'll definitely look at this for Longhorn. The complexities of touching save code for us are mind-boggling. This is our most high risk area.

-----Original Message-----
From: Ben Fathi
Sent: Tuesday, July 09, 2002 8:28 PM
To: Steven Sinofsky; Bill Veghte

Cc: Antoine Leblond
Subject: RE: Office 11 and .NET server

I'll schedule some time with Antoine to go over the details of how this works in the current implementation Please let us know if you'd like to be involved

Bottom line is that the architecture provides a simple mechanism for apps (e.g., outlook and powerpoint just as much as Exchange and SQL) to get involved in snapshots. We'd love to work with the office team to make sure the appropriate bits are getting saved to the disk and are presented to the user for retrieval.

Ben

-----Original Message-----
From: Steven Sinofsky
Sent: Tuesday, July 09, 2002 5:05 PM
To: Ben Fathi; Bill Veghte
Cc: Antoine Leblond
Subject: RE: Office 11 and .NET server

Ah—the difference is that our RAM is not a "buffer" it represents the work that a person has explicitly not saved. We can't use this metaphor because our autosave goes to a different location and does not overwrite the previous "saved file".

Of course STS works because the file is not open on the server. However, a shared document (all apps allow multiple writers now) will not work.

I haven't seen how this will not be in an inconsistent state. So I'm a bit more worried than I used to be.

Antoine and I have been talking about this. It would make sense for the two of you to talk just to make sure we're not going to get in a pickle over this. I of course might totally misunderstand but I am super worried about the idea of "corrupt" files getting created.

-----Original Message-----
From: Ben Fathi
Sent: Tuesday, July 09, 2002 4:42 PM
To: Steven Sinofsky; Bill Veghte
Subject: RE: Office 11 and .NET server

The snapshot is done at the block level, underneath the file system. It creates a new virtual file system that can be mounted or enumerated (by shell, for example). The resulting file is seen by the OS as a different file on a different (read-only) file system, so apps can open it without affecting the original file. The shell UI allows you to open, drag and drop, or revert to this old version.

The crash analogy is there because some apps don't cooperate with the snapshot model. The apps that care (STS, AD, SQL, Exchange, etc) do cooperate and you get a fully consistent version of the file.

Simple scenario on when this would be used: I worked on a presentation all last week (and the server has been taking hourly or daily snapshots of it for me all along). Ten minutes before the

meeting, I edit the file and delete some slides by mistake, then hit "save". Using the shell UI, I can just recall the last snapshot version and get back to the correct version of the file. Our competitors (e.g., NetApp) offer this feature today on their file servers and it is by far the most requested feature from IT organizations and end-users. It gets the admin out of the backup/restore loop and significantly reduces file server TCO. I agree that it's not as good as explicit file versions in some cases, but it is far better than what we have today and the entire industry (from storage hardware vendors to backup apps to SAN vendors to app developers like SQL and Oracle) are working with us to get this delivered for .Net

Would it help if we set up a one hour presentation and demo on this?

Ben

-----Original Message-----
From: Steven Sinofsky
Sent: Tuesday, July 09, 2002 3:02 PM
To: Ben Fathi; Bill Veghte
Subject: RE: Office 11 and .NET server

How does this get done—if the file is open it is not in a consistent state. The file needs to be closed before it can be reopened by Office. I guess it is the same as a crash but then you can lose more work than you gain back. Using the simulation of a crash analogy is a very poor user experience. It looks more like an error than a "I'd love to get back version n-3 of my document".

The alternative is to go to the autobackup which is there by default and consistent with the user setting.

We're just confused over how this will work in practice and when someone is supposed to use this?

-----Original Message-----
From: Ben Fathi
Sent: Tuesday, July 09, 2002 2:43 PM
To: Bill Veghte
Cc: Steven Sinofsky
Subject: RE: Office 11 and .NET server

Snapshots are taken from the entire file system, at the block level There is an architecture in place with "writers" for each app that cares to add them to make sure the data is consistent on disk when the snapshot is taken. Examples of this include Exchange, SQL, AD, registry, and the FS itself, so what you get includes what was in RAM at the time the snapshot started (since the FS is told about the snapshot and flushes its buffers during the snapshot process). We do snapshot open files as well as closed files.

In the case of the example you gave below, the recovery procedure is the same as what it would be for a system that crashes, with the additional advantage that nothing in RAM is lost (i.e., we are "crash consistent" if the app doesn't participate in the snapshot). As far as I know, all Office apps can recover from this gracefully. The "oops" scenario is exactly what we can fix and have demonstrated. The gotcha is that you don't get an "undo" of the last set of changes before the save, but the file as it was at the time of the last snapshot. Yes, you may lose some work, but you'll get a reasonable version of the file that you can restart from. The alternative would be to go to the last set of backup tapes and ask an administrator to restore the file.

Ben

-----Original Message-----
From: Bill Veghte
Sent: Tuesday, July 09, 2002 2:27 PM
To: Ben Fathi
Subject: FW: Office 11 and .NET server

He has a very valid point. I want to get a reply back on this thread today so we keep things moving.

-----Original Message-----
From: Steven Sinofsky
Sent: Tuesday, July 09, 2002 11:04 AM
To: Bill Veghte
Subject: FW: Office 11 and .NET server

Hey...I'm still really stumped on how this will work in practice.

As far as I can tell, this will checkpoint only closed files (otherwise the files would be in an inconsistent state since the file would be represented by committed pages+RAM). The problem is we never close files until the user explicitly closes them. So there isn't really anything that this can do for the common case of "oops I was editing, hit save, and now I want the old one back [undo goes away after a save]" This seems more like the old style—before opening a file make a backup copy.

Are we confused? A few of us have been talking about this.

-----Original Message-----
From: Steven Sinofsky
Sent: Tuesday, July 09, 2002 9:36 AM
To: Bill Gates
Cc: Anoop Gupta (RESEARCH); Bill Veghte
Subject: RE: Office 11 and .NET server

BillV and I have been talking about this.

The feature is very cool, but it is not manifested in a way that I think lends itself to file open support. The "go back in time" is not to a specific time or to the "last saved version" but to some arbitrary (configurable) time when the system snapshotted it. On VMS, for example, every file save operation creates a new version so the end-user has a clear concept of when back in time really is.

As a result, I think the best place for this will really be in the shell since there will need to be some non-trivial UI around the feature (in the shell task pane). I think you're going to want to be careful about overwriting the existing one—in fact this should just open up as a whole new file. You're going to want to see all the time and date information, etc. The direction for file open has been less "details" view and more about thumbnails and big icons (presumably the previous version would not be a radically different thumbnail).

For example, it isn't really clear to me what we would put in file open? I suggested just having the files show up as additional IShellView items, but I think chrisg explained that this is not how the enumeration is implemented (we would pick this up if it were the case since we use the Windows controls).

In our conversation I think we concluded that there isn't an obvious mechanism for inclusion into file open the way the feature was implemented and that the UI in the file system's work pane might make the most sense.

-----Original Message-----
From: Bill Gates
Sent: Tuesday, July 09, 2002 9:28 AM
To: Steven Sinofsky
Cc: Anoop Gupta (RESEARCH); Bill Veghte
Subject: Office 11 and .NET server

.NET server has one feature we plan to make a big deal about – this feature where you can go back in time to get files that are deleted.

One thing I don't understand as well as I should is whether Office 11 can or does expose the ability to get at these files.

I assume the standard Windows Open dialog is being updated to do this so there is a clear guideline on how Office should expose it in its open dialog.

In fact someone could even make a case we should update the Office open dialog for previous versions of Office.

This might be too complicated but at least it seems like we should be able to show off how Office11 takes advantage of this feature.

If this hasn't been done I know it is late but I think having this one thing to brag about to show off .NET would be worth looking into.