Using Terminal Server.

This article we're looking at working remotely with Paradox. It's somewhat of a case study, as I'll be focusing on using Windows Terminal Server as a means of serving up a Paradox system to a number of remote users. This study uses a technique that was both convenient, because of the framework being used, and also required, because only Paradox 9 SP3 was available at the time of implementation, not the SP4 version that was released at the time the release took place. Re-implementation into Paradox 10 proceded with literally no hitches whatsoever.

The company that is the subject of this study has a package of subsidised benefits (health care insurance, etc.). The company has a head office in one large city and about 30 branches around the east coast of the U.S.A. Up until last year the company had physically sent out a database package to each remote branch each year, consisting of a copy of Paradox, a program to allow the users to enter their benefit choices, and a small database, containing just those employees at that branch. One employee at each branch was designated the "Benefits Administrator" and ran the program. When all was done, the whole database was zipped up and returned to head office on a floppy. Once received, the data was collated and used to update the main personnel database for the new year.

Finally it was decided that this state of affairs could not continue, not only for the growing number of reliability reasons, but also for the general inconvenience caused to all concerned. The whole of the HR and Payroll system was written in Paradox, so it seemed reasonable to create a new benefits enrollment package in Paradox also, keeping the data in one environment throughout. However, we could not accept the continued idea of the exchanges of data on floppies, etc., so we decided to use the Terminal Server abilities of Windows 2000. Terminal Server is intended to assist in reducing the Total Cost of Ownership (TCO) of an application system by eliminating the actual installation of the application at remote sites, which may be spread all over the globe.

The Hardware.

As many people find themselves, our MIS department seems to be in a permanent state of transition, so what we were able to obtain, for our ten-user licence for Windows 2000 Terminal Server, was a 350 MHZ Pentium II Micron Millennium, with an 8 GB IDE hard drive and a 10 Mb/s Ethernet card. We pushed the memory to 396 MB, the limit for the motherboard, which gave us about 40 MB of RAM per projected simultaneous user. Having used Paradox 9 successfully on a laptop with just 48 MB, I hoped that this might just be enough memory.

The Operating System

Onto this machine we installed just Windows 2000, Windows 2000 SP1, along with the Novell client system for it to work on our Novell network. Then the Windows 2000 Terminal Server licence ten-pack and WinZip were installed. Backup of the application was envisaged being a simple copy of the database directories when no users were logged in, followed by zipping up the data and storing it off the machine. This has proven to be a very effective technique.

Windows 2000 Terminal Server manages to provide a number of users with a slice of a Windows 2000 machine by running a number (ten in our case) of “virtual machines” (VMs) in parallel. Each VM appears to be a perfectly normal copy of Windows 2000. The C.P.U. switches from VM to VM on a time-slice basis - so many milliseconds per VM. All processing is done at the server, and the resultant screen image is transmitted to the client for display.

Client software is available for a number of different operating systems. Our users use Windows 98, so the interface to the Windows 2000 system within the client window was very familiar. This is, in many ways, a good aspect of the Windows Everywhere theology. However, for support personnel, it proves very difficult at times to get a user on the other end of a phone line to do what they are asked in the correct environment ! Very often they end up doing something in the Windows 98 machine, instead of on the Windows 2000 VM in the Client window.

This problem of confusion was exacerbated by the fact that the application was created with resolutions of 800x600 in mind, which is just exactly what most of the remote sites use. Therefore the client software often takes up the whole screen and still has scroll-bars, so, very often, the only Start Bar visible is the local Windows 98 one !

Based on this experience, we tried changing the background appearance of the Windows 2000 machine. However, we found that doing that forced a very large overhead in the local user environment of the Windows 2000 system, and as we were looking at a total of 150 user identities, we realised that we would run out of physical disk space ! This is a side effect of the method of determining the logon identity of the user for the application, and, had we known about it earlier, could have been easily avoided. However, by the time we realised our error, the application was already written and in use.

Finally, we took the usual precautions against opportunistic locking, for both client and server, on the server machine. The same settings apply for Windows 2000 as do for Windows NT 4.

The Application System

We used Paradox 9, Paradox 9 SP3, and version 5.11 of the Borland Database Engine (BDE). We could (just) have used Paradox 9 SP4, which is more adjusted to the needs of Windows Terminal Server, but our application was just days away from being brought into use when the service pack was made available, and we had no chance to test the system to ensure stability. At the end of this article are a number of sections that deal with later versions of software.

The application that ran consisted of just twelve forms, but 56 tables. The total size of the application, including the skeleton that provided menus and security, etc., was about 140 MB, including all the data for about 1000 people. All this was placed onto the C: drive of the Terminal Server system

The other point about Paradox is that it needs a **Network Directory** for the `PdoxUsrs.Net` file. Normally, one would place this in some special directory on the shared drive where all the data is kept, so that all users can be assured of being able to "see" it. However, with this application, different remote sites had different drive letter mappings, so this was impractical. As a result, we just created a new directory (`PdoxCntl`) on the C: drive of the Terminal Server machine and put it there. This means that our network has two Network Directories, but as no applications ever use tables "owned" by both, we encounter no problems. One should always think carefully about how applications interact before doing this, however.

So far, so good - for a single-user system. However, we were going to have to deal with up to ten instances of Paradox running concurrently, and the user base numbers about 150 (most local; plus one for each remote branch site). This means that we can have up to ten Windows 2000 systems running at the same time, each hosting a Paradox system. In fact, because of the way that Terminal Server Client looks to the user, as mentioned above, it's very easy for people with little experience to minimise an application and then start it again, thinking that it is no longer running. In fact, we've seen some users with three or more instances of Paradox in their virtual machine, and totally unaware of the fact !

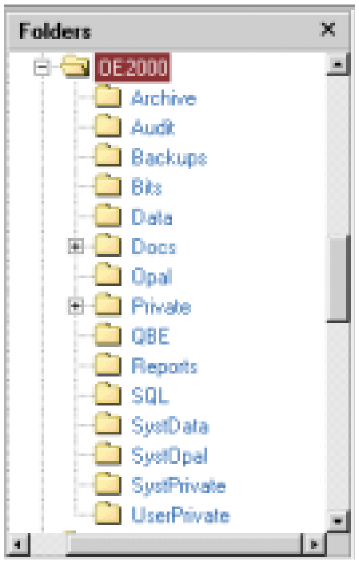

Figure 1 Application Layout.

To solve this problem with the Paradox Private directory, we created a directory structure as shown in Figure 1.

The directory `OE2000` is known as the **Application Root**. For convenience, this is also the location of the alias `:work:`, but, as this alias is never used, `:work:` could be elsewhere.

Held in the Application Root are three files that start the application:

- `starter.vbs`. VBScript to start Paradox
- `starter.ssl`. Starter script to create environment
- `starter.ini`. Text file holding definition of environment.

In this case, all the other subdirectories of the application are actually physically below the

Application Root. This, of course, is not necessary; any of the directories could be located on other drives or across the network.

Starting The Application - The VBScript Part

In order to start each instance of Paradox with a unique Private directory, we wrote a short piece of VBScript to wrap around Paradox. It would set up the private directory and then start Paradox, specifying the private directory on the command line. The essentials of the VBScript are as follows:

```
'various declarations
'.........................The base set of Paradox Private objects
strPrivate = "C:\OE2000\Private"
strUserPrivate = "C:\OE2000\UserPrivate"
'.........................'create objects we'll need
set shell = createObject("WScript.Shell")
set fso = createObject("scripting.fileSystemObject")
'.........................obtain the Volatiles
set env = shell.Environment("Volatile")
'.........................get the Volatile "NWUSERNAME"
strUsrEnv = env.item("NWUSERNAME")
`                               error trapping code ....
btnCode = shell.popup(strUsrEnv, 30, "Open Enrollment")
strLocalPrivate = strPrivate & "\" & strUsrEnv
'.........................There shouldn't be a folder yet ...
for iLoop = 1 to 10
   if not fso.folderExists(strLocalPrivate) then
      exit for
   else
      strLocalPrivate = strLocalPrivate & "a"
   end if
next iLoop
btnCode = shell.popup("Please wait while your environment is loaded", _
                     5, "Open Enrollment")
fso.CopyFolder strUserPrivate, strLocalPrivate
if not fso.folderExists(strLocalPrivate) then
   btnCode = shell.popup("Unable to create Private Directory", _
                     10, "Open Enrollment", vbOk)
   WScript.quit
end if
'.........................
strProgram = "C:\Program Files\Corel\WordPerfect Office _
                     2000\programs\pdxwin32.exe "
strSwitch = " -c -m -q -p"
strPrivate = strLocalPrivate & " "
strWork = "  -wC:\OE2000  "
strObject = " starter.ssl"
'.........................
strStartString = chr(34) & strProgram & chr(34) & strSwitch & _
                     strPrivate & strWork & strObject
'.........................
shell.Run strStartString, SW_SHOWNORMAL, True
'.........................
fso.deleteFolder strLocalPrivate, True
btnCode = shell.popup("Closing down ...", 5, "Open Enrollment")
```

Note that in lines `031`, `034`, `038` and `044` the trailing “ _” signifies a line-wrap to VBScript.

Line `013` Line `013` obtains a copy of the environment from the OS and makes it available to the VBScript code. Line `015` extracts the value of the item in the environment called `NWUSERNAME`. This item is always available if you are logged in to a Novell system. Lines `016-021` (omitted) trap for a missing name, tell the user to log on to Novell first, and exit. Line `022` creates the specification for the private directory from the standard private directory concatenated with the NetWare user name held in `strUsrEnv`. Lines `024-030` check that this is not already in use and, if it is, add a letter `a` to the end of it and try again. Eventually they obtain a unique name and then use it.

The reason for this loop is partially that we were getting people logging in more than once by accident, and we wanted them to be able to get at the information. An “old” copy of the app hanging around really doesn’t affect the data, and can easily be killed off.

VB error However, we did run into a more serious problem, and that is that VBScript does not always succeed in getting the correct value from the volatile area when it asks for it ! We found that if a user with Windows Administrator privileges was logged on to the system, especially logged on to the server directly, other people would often obtain the administrator’s name, or, sometimes, that of the previous person to log on to the Novell net ! For this reason we do not complain if we find that the supposedly unique name is already in use, but change it and try again. This is a workaround, not a cure, unfortunately, but we didn’t find any assistance in the VBScript references.

Line `032` This line creates the private directory, and the next four check that it was successful.

`038-044` These lines create the various parts of the command line for starting Paradox

Line `046` This is the VBScript line to start Paradox. The final `True` indicates that the VBScript interpreter should wait until the program is finished before continuing. When it does, it deletes the private directory.

Wrapping Paradox in a VBScript shell like this means that any cleaning up after Paradox has finished can be done by the script, regardless of whether Paradox finished normally or was crashed, which is very convenient for system support.

Starting The Application - The Paradox Part

We use a well proven system where all our applications start with a starter script that establishes the alias mappings for the application and then play an application-specific script that controls logging in and which starts things like libraries before launching the desktop form. This makes building an application rather easy, as all one has to do is concentrate on getting the inside of a form or report correct; the outside tends to take care of itself. As this system has been in use for some five years it is well debugged, and has been found to work just as well under Terminal Server as it does when Paradox is running on a local system. Obviously, we were very happy to discover this !

Problems - Tweaking Paradox

We tested the application with three users from MIS and it appears, with 20-20 hindsight, that they were much too kind to it. On the very first day, within about an hour of the system going live, we had our first major problem. This was that a user pressed a button on a form to launch another form, and the second form halted. Paradox reported that it was unable to load the form because a certain table was locked. This was puzzling, since checking the form showed that the table was only used for populating a drop-list when the form was opened, and never again. The user cited in the error message as having the table locked was, by the time we arrived on the scene, not even logged on. This happened on several times, and each time the user could click on the Ok button of the error message and then Paradox would freeze. One then had to use the Windows 2000 Task Manager to kill the Paradox process and restart it; very often matters would then proceed normally.

The immediate answer was to remove the cited table from the data model of the form and populate the drop-lists with code using tCursors or an SQL command. However, this just caused the error to happen on another table. We removed four such tables from the form. In addition, looking at the problem in parallel with altering the form, we changed the Paradox network settings. We pulled the Refresh Rate down from 60 seconds to one second, and pushed the Retry Period up from one second to five.

This meant that our copies of Paradox were refreshing their data very frequently. However, as the network was totally local, the data and the instances of Paradox all being on the same machine, this doesn’t matter in the slightest. The advantage is that changes are seen immediately and database latency was reduced.

Lengthening the retry period would appear, from the Help text, to mean that our lockout period would be even longer than before. However, even with three seconds, it appeared infinite, so we assumed that the time-out was somehow affected by the intervention of the Terminal Server VM switching, meaning that applications within the VM couldn’t really count properly.

After introducing these changes, which may only be supplemental to the one of removing all the effectively-unused tables from the form’s data model, all went smoothly.

Printing

We all know that printing has produced a number of problems for users of Paradox 8 & 9, so we started out prepared for the worst. The first problem, that of printing Landscape instead of Portrait, never seemed to appear. Instead, we had the rather odd effect of getting a Portrait output on some LaserJet 5L machines and a Landscape one, as intended, on other HP printers. Since the Portrait version looks fine, because we were in the process of converting the Landscape layout to Portrait anyway, we have left matters as they are. However, we did encounter a few problems with printing that required a little ingenuity to dispell.

No printing at all

Our first problem was that, under some circumstances, printing would result in no execution of the print function in the code at all. We have code that happens to hide the form as it prints (a small "feature", soon to be removed); on these occasions the form was never hidden, indicating that the print command was never being triggered within Paradox. Purely by chance, we discovered that printing to a screen image, rather than directly to the printer, would work, and that printing directly to the printer thereafter would work reliably. Therefore we immediately created a method that was called by any form with a Print button, that opened a report, hidden, on-screen just as the form is opened, and then immediately closes it. Another workaround, and something for the Corel staff to ponder.

Printing to the wrong place

The other major problem we had with printing was that of finding out where to print ! Paradox, you will recall, is running in a VM at head office, whereas the user is sitting maybe 2000 miles away, and wants a print-out *there*, not at head office. Which printer should Paradox use ? All seemed to be fine until we got one early user complaining that Paradox was printing to the fax machine ! We found that a VM enumerates the printers for you, including all the common printers that are visible to the host server, and also any that the local machine can see, so that each site can, potentially, see a different list. A user with Administrator privileges can "see" all the local printers.

A local printer is distinguished from others by the use of the string "`Session n`" appended to its name, where "n" is the Terminal Server session number or VM number. For ordinary users, because they only have one printer connected locally, we adopted the simple expedient of enumerating the printers, searching each string for the substring "Session", and making that printer the current printer with the built-in method `printerSetCurrent()`. For users who might have multiple printers attached, we presented a small form with all the possible local printers to choose from.

Printers

Printers themselves, of course, are a whole other problem ! Most of the remote offices use HP printers of some sort or another, and we have found that making them connect to HP LaserJet Series II printer drivers cures almost all problems. However, we found that some sites using Lexmark printers just couldn't be persuaded to print correctly, and we were forced to purchase HP machines for them. The only alternative would have been to have arranged to print the report to PostScript on the server, convert it to Acrobat there, and email it to the site. This was tested and found perfectly feasable, but we really considered this a technique of last resort. The problem with Lexmark printers, by the way, appears to be a known problem for Windows 2000 Terminal Server, so you should expect to encounter it.

Other Items Of Interest

Configuration Files

Sometimes, on ending a Paradox session, the user would encounter three error messages stating that Paradox was unable to write back to the two toolbar `.cfg` files in `C:\Program Files\ Corel\WordPerfect Office 2000\programs\config`. We watched this for some days and concluded that being unable to do that (because some other user was doing it at the time) had absolutely no effect on the functioning of either Paradox or the application, so we asked our users to ignore it, putting the blame on Terminal Server !

Memory Allocation

As I mentioned at the beginning of this article, we ran this system with up to ten concurrent users on a machine with just 396 MB of RAM. This is almost certainly too little memory for ten copies of Paradox, and we found at times that the whole system just started to suddenly slow down to the point when displaying the Start menu took up to a minute. At this point we normally took the route of finding a couple of users who were not actually in Paradox and terminating their VMs. Invariably, the machine then picked up again. We think that about 60 MB per user would be a more appropriate minimum for using Paradox.

Printer Redirection

This is the Windows Terminal Server name for getting print output from a program on the host to appear on a local printer at the client's machine. We experienced an ongoing problem where some event would cause the Windows 2000 kernel to declare that print redirection services were abandoned, but we could never discover just what it was that was causing it. We could see (in the Event Log) print jobs happen (successfully) and be purged, and immediately after that we found the Print Redirection Failed event. We set the system to restart Print Redirection whenever it was found to be halted, which got around the problem, but are still looking for the real cause of the problem.

Conclusions

We entered this development with a small amount of worry, because Corel had stated that Paradox 9 SP3 was not tested against Terminal Server, but that SP4 would be approved for it. However, time was against us and we discovered, to our relief, that many of our fears were totally groundless. Paradox systems can, with just a small amount of modification, be run on Terminal Server systems, and this provides a very easy and cost-effective way of distributing the Paradox front-end to a database to a lot of people in a short period of time.

Our thanks must go to Dan Alder for his support of Paradox, and also to Hanno van Pelt, from Guatemala, who has also used Paradox with Terminal Server and who provided some much-needed reassurance at a time of high pressure.

Two Years Later

Two years have now passed. The Open Enrollment system has been used three times and has become "the norm" to such an extent that users are now grumbling about the few shortcomings that it has, such as the screen-size problems.

In these two years another program, a Payroll & HR Data Entry system (SDE), has been added to the same server. SDE shares the main HR database with another Paradox system that is hosted on a Novell server in-house, so it has to share the Paradox Net File held off the Terminal Server. Converting the OE program to do this caused no problems whatsoever. This additional program is used every day, 5 days a week, throughout the year.

Windows Problems

- One user only reports problems with her Terminal Server Client instance freezing, which is frustrating, both from a software engineering point of view and also from the point of view of having a dissatisfied customer.
- Installing SP3 onto the Terminal Server machine eliminated a lot of problems. In addition, we have found that if an administrator installs the drivers for a printer (but not the printer itself), then Windows will not find that driver and automagically install it when the user with the printer logs on, thus meaning that we can have a number of different printers available.
- Win2KTS SP3 also appears to have cured a lot of other printing problems, including the one where we needed to bring up a blank report before ever starting a real one.
- Lexmark printers appear to be moderately acceptable now.
- Printer redirection failures are now about 1% of their original levels.

So, Microsoft appears to have improved Windows 2000 Terminal Server with their Service Packs.

Paradox Setup

Since Paradox 9 SP4, Paradox on Windows 2000 and Windows XP machine has respected the new `Documents and Settings` directory structure, placing its default `work` and `priv` directories there. However, we have good reasons for wishing to have our directories elsewhere, so we have ignored these settings.

As before, `:work:` is the Application Root of the project, and `:priv:` is a directory specially created and filled for the occasion by the VBS wrapper. Now, the actual name of the directory for `:priv:` has no connection with the user, because it doesn't need to ! This means that we don't need to know who has invoked the instance of Paradox; we create a new subdirectory, fill it from SystPrivate, and then use it as the private directory for the new instance of Paradox.

Other Systems

Over the last two years I have performed test transfers of four other complete systems to the Terminal Server, and tested them out. In no case was the required change to the application anything more than changing the alias definitions in the Starter.ini file, that the applications read to see where to find things.

This, I believe, is a good testament to the abilities of Paradox as a database to perform with a Windows Terminal Server system.