SQL Server 2008 Upgrade

Gotchas & Fixes



Summary:

In preparation for upgrading our SQL Server 2005 production database servers to SQL Server 2008, it was prudent and necessary to test the upgrade process and regression test the existing applications dependent on those servers. To accomplish this task I upgraded a series of development and test servers and in so doing found several “Gotchas” which I had to fix. I define “Gotchas” as documented or undocumented issues with the install process that either slowed the process down or stopped it altogether. The purpose of this paper is to relate my discovery of those Gotchas and how I resolved them.

The Environment:

Our current production environment has eight primary production database servers, and another half dozen outliers that I will not address here. Four of the eight servers are clustered using Windows Server 2003 R2 and SQL Server 2005 SP2. Each production server is in a two node configuration with their own shared SAN disk subsystem. The other four servers are BI servers that are stand-alone, but support each other’s functionality. There is only one x64 production server, and all other servers are x86 architecture. There are two domains, which are not trusted since one of the domains is the public facing web site and the other is the internal corporate domain. In support of the production servers are several dev and test servers, one of which is clustered.

The Process:

I began this process with interviews with the principal stakeholders, which I broke into four functional groups. These groups were Web/CRM/MOSS/BI, and a couple of special purpose outliers. I assigned the responsibility of regression testing the applications to each of the functional groups.

I used the upgrade procedure documented in the MSDN article, How to: Upgrade a SQL Server Failover Cluster Instance (Setup) <http://msdn.microsoft.com/en-us/library/ms191295.aspx> as my upgrade guide.

I upgraded the CRM dev and test servers first, and then the BI servers and finally the shared MOSS/Web test server, which was the only clustered non-production server.

The Gotchas:

1. Reporting Services required for the Upgrade Advisor.
2. Windows Installer v4.5 hotfix.
3. Visual Studio 2008 - SP1 required for any upgraded v3.5 .NET components.
4. Install packages need to be on local drive for SQL Native Client to upgrade to SP1.
5. MSCORSVW.EXE in use during SP1, forces a reboot.
6. Proper sequencing of actions to reduce the number reboot and failover events.
7. Developers required upgrade to SQL Server 2008 workstation tools.

1) Reporting Services (RS) required for the Upgrade Advisor

This is a logical gotcha that kinda makes me mad. Microsoft Best Practices and our corporate security guidelines state that you should not put Reporting Services on the same server as an Operational Data Store (ODS). The crux of the matter is that the SQL Upgrade Advisor (SUA) has to be on the server that it is scanning. It makes much more sense that the SUA be installed on a separate RS server and pointed towards the target server with a connection object. I went ahead and installed RS on the first dev server and then ran the SUA scan. 

The scan revealed nothing I didn’t already know. Secondly, the SQL upgrade program does a similar scan, which will reveal any show stoppers. I made a judgment call and decided not to run SUA on servers that don’t already have RS on them.

2) Windows Installer v4.5 hotfix

This gotcha is primarily for Windows 2003 R2, as Windows 2008 already has the latest installer. Before you install SUA or SQL upgrade, you have to install the latest Windows installer v4.5. This comes with the SQL install program, or it is available as a separate download in a windows hotfix (KB942288-v4). 

The installer will require a reboot of the server, regardless from which package you install it from. This becomes a timing issue and should be planned for.

3) Visual Studio 2008 - SP1 required for any v3.5 .NET components.

This gotcha didn’t stop the install but it did require a lot of rework. If Visual Studio 2008, or components that use Framework 3.5 (CRM 4.0 is one) are on the server to be upgraded, then Service Pack 1 for Visual Studio 2008 (VSSP1) needs to be installed before the SQL upgrade begins. 

When I upgraded the first dev server, it had Visual Studio 2008 already on it, but not VSSP1. The Integration Services (IS), Client Tools, Management Tools, BIDS and several other components did not get upgraded. I had to go back, install VSSP1, and then do full repair on the SQL 2008 instance before installing SP1. This cost me at least 4 hours.



4) Install packages need to be on local drive for SQL Native Client to upgrade to SP1. 

This next gotcha I had a hard time researching, but finally found a forum reference that gave me the clue. I was doing a network install from a shared drive to circumvent copying the upgrade packages to the local drives. The upgrade itself went without a flaw, but the Service Pack 1 install bombed completely. As it turns out when you do a network install, there is some versioning problem with the SQL Native client component, that I have yet to figure out, that stops the install of SQL 2008 Service Pack 1. The fix is to copy the upgrade packages to a local drive, do a complete repair, and then install SP1.

5) MSCORSVW.EXE active during SP1 install process, requires a reboot.

This one is very easy to fix. Do nothing for 10 minutes, and then restart the SP1 install. As it turns out the v3.5 Framework has to recompile the assemblies that are installed. The MSCORSVW program does this in the background after you do the SQL 2008 upgrade. The process takes about 10 to 15 minutes and then is done for good.

6) Proper sequencing of actions to reduce the number reboot and failover events.

I only had one clustered server in my test environment that was clustered. The install itself went without issues, but I hadn’t considered timing as well as might have. The passive node should be rebooted , upgraded and service packed before failing over to the other node. Once the primary node fails over to passive the process should be done to completion. I began the failover before upgrading to Service Pack 1, so I had to failover a second time to accomplish that. It’s a minor nit, but reboots and failovers can be minimized by careful planning.

7) Developers required upgrade to SQL Server 2008 workstation tools.

The upgrade of the dev/test servers was widely publicized, but of course the developers ignored it because it didn’t concern them. It didn’t concern them, until I upgraded the primary (clustered) test server and then it concerned all of them. Fortunately our Fortune 500 Company has a site license for SQL Server, so I simply steered all requests to the share drive with SQL Server 2008 Developers Edition. They were all savvy enough to know that you don’t need to install a local instance of SQL to get the client tools. If I were more proactive about it however, I would have communicated to them earlier about their need to upgrade their client tools, and of course some of them hadn’t applied VSSP1, so they had to do that before the SQL upgrade.

In Conclusion:

The lessons learned in practicing the SQL Server 2008 upgrade in our dev/test environment were invaluable in discovering the Gotchas that would have slowed down the production rollout. These steps are prudent and necessary whenever rolling out any major upgrade to the RDBS supporting all applications. Additionally, I had made the requirement that all the functional units did regression testing of their applications to ensure that there were no hidden Gotchas with their applications and SQL Server 2008. With only a couple of minor exceptions, there were no significant issues with the applications using the upgraded SQL database servers.

The benefits of this upgrade became apparent to all testers almost immediately, as they saw faster response times from SQL Server 2008 from their applications. I am preparing a whitepaper for the developers on the new features of SQL Server 2008 that they should consider for new development projects. The most anticipated features of SQL Server 2008 that I am looking forward to are the Resource Governor and backup compression on fly. I now have a test bed to quantify the real gains from these features that I can report to management, which ultimately justifies the time and expense of the SQL Server 2008 upgrade.