

**SQL Server 2008 Upgrade Technical Reference Guide**

**By Microsoft Corporation**

**Acknowledgements:**

**Contributing writers from Microsoft**: Arvind Rao, George Huey, Richard Waymire, Siva Harinath, Edward Melomed, Deepika Mistry, Fernando Caro, Goldie Chaudhuri, Max Verun, Vijay Tandra Sistla, Tom Michaels, Justin Erickson, Devendra Tiwari, Jingwei Lu, Fernando Azpeitia Lopez, Ketan Duvedi, Lukasz Pawlowski, David Noor, Matt Masson, Karandeep Anand

**Contributing writers from Solid Quality Mentors**: Ron Talmage, Aaron Johal, Steven Abraham, Allan Hirt, Herbert Albert, Antonio Soto, Greg Low, Joe Webb, Craig Utley, Dejan Sarka, Larry Barnes, Pablo Ahumada

**Technical reviewers from Microsoft**: Rebecca Laszlo, Saket Suman, Paul Mestemaker, Vishal Anand, Leo Giakoumakis, Alejandro Hernandez Saenz, Tom Michaels, Bob Ward, Lindsey Allen, Sanjay Mishra, Umachandar Jayachandran, Mike Wachal, Richard Tkachuk, Donald Farmer, Ritu Kothari, Rakesh Parida, Prash Shirolkar, Dave Sell, Craig Guyer, Denny Lee, Peter Scharlock, Yinyin Gao, Rahul Sakdeo, Eliza Tobias, Hajnalka Sarvari

**Contributing editors from Solid Quality Mentors**: Kathy Blomstrom

**Contributing editors from Microsoft**: Jen Witsoe, Suzanne Bonney, Megan Bradley, Tresy Kilbourne, Bronwyn McNutt

Microsoft gratefully acknowledges the contributions from numerous additional writers, reviewers and editors involved in this document, and also those who developed the SQL Server 2005 Upgrade Technical Reference Guide.

**Published:** November 2008

**Applies to:** SQL Server 2008

**Summary:** A successful upgrade to SQL Server 2008 should be smooth and trouble-free. To achieve that smooth transition, you must plan sufficiently for the upgrade, and match the complexity of your database application. Otherwise, you risk costly and stressful errors and upgrade problems.

Like all IT projects, planning and then testing your plan gives you confidence that you will succeed. But if you ignore the planning process, you increase the chances of running into difficulties that can derail and delay your upgrade.

This document covers the essential phases and steps to upgrade existing instances of SQL Server 2000 and 2005 to SQL Server 2008 by using best practices. These include preparation tasks, upgrade tasks, and post-upgrade tasks. It is intended to be a supplement to SQL Server 2008 Books Online. It is not intended to supersede any information in SQL Server Books Online or in the Microsoft Knowledge Base articles. The reader will notice many links to SQL Server Books Online topics and Knowledge Base articles. In all such cases, the information in this document is included to provide the context you need to decide whether to spend the time to read the linked article. If there are any discrepancies between this document and a linked article, the linked article is assumed to be more accurate.

© 2008 Microsoft Corporation. All Rights Reserved. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

**Copyright**

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2008 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, Excel, Intellisense, Internet Explorer, Outlook, SharePoint, SQL Server, Visual Basic, Visual Studio, Windows, Windows NT, Windows PowerShell, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

**Table of Contents**

[1 Upgrade Planning and Deployment 17](#_Toc215243591)

[1.1 Introduction 17](#_Toc215243592)

[1.2 Feature Changes in SQL Server 2008 17](#_Toc215243593)

[1.3 Preparing to Upgrade 17](#_Toc215243594)

[1.3.1 Upgrade Strategies 18](#_Toc215243595)

[1.3.2 Backward Compatibility 31](#_Toc215243596)

[1.3.3 Upgrade Tools 33](#_Toc215243597)

[1.3.4 SQL Server 2008 Setup 37](#_Toc215243598)

[1.3.5 Allowable Upgrade Paths 43](#_Toc215243599)

[1.3.6 Application and Connection Requirements 48](#_Toc215243600)

[1.3.7 Plan for Backups 50](#_Toc215243601)

[1.3.8 Upgrading Both Windows and SQL Server 50](#_Toc215243602)

[1.3.9 Upgrading Multiple Instances 52](#_Toc215243603)

[1.3.10 Upgrading Very Large Databases 52](#_Toc215243604)

[1.3.11 Upgrading High Availability Servers 53](#_Toc215243605)

[1.3.12 Minimizing Upgrade Downtime 53](#_Toc215243606)

[1.4 Developing an Upgrade Plan 54](#_Toc215243607)

[1.4.1 Treat the Upgrade as an IT Project 54](#_Toc215243608)

[1.4.2 Minimize Variables Involved in the Upgrade 57](#_Toc215243609)

[1.4.3 Create Upgrade Checklists 60](#_Toc215243610)

[1.4.4 Test the Upgrade Plan 60](#_Toc215243611)

[1.4.5 Develop Acceptance Criteria and Rollback Steps 63](#_Toc215243612)

[1.5 Post-Upgrade Tasks 63](#_Toc215243613)

[1.5.1 Integrate the New Instance into Its New Environment 64](#_Toc215243614)

[1.5.2 Determine Application Acceptance 64](#_Toc215243615)

[1.5.3 Troubleshooting an Upgrade 64](#_Toc215243616)

[1.5.4 Decommission and Uninstall After a Side-by-Side or New Hardware Upgrade 65](#_Toc215243617)

[1.6 Considerations for Upgrading without a DBA 65](#_Toc215243618)

[1.7 Conclusion 67](#_Toc215243619)

[1.8 Additional References 68](#_Toc215243620)

[2 Management and Development Tools 69](#_Toc215243621)

[2.1 Introduction 69](#_Toc215243622)

[2.2 Feature Changes in SQL Server 2008 Management and Development Tools 69](#_Toc215243623)

[2.2.1 SQL Server Management Studio 2008 Changes 70](#_Toc215243624)

[2.2.2 Business Intelligence Development Studio2008 Changes 73](#_Toc215243625)

[2.2.3 SQL Server Configuration Manager 75](#_Toc215243626)

[2.2.4 Reporting Services Tools 76](#_Toc215243627)

[2.2.5 DTS Tools 76](#_Toc215243628)

[2.2.6 SQL Server Agent 77](#_Toc215243629)

[2.2.7 Maintenance Plans 80](#_Toc215243630)

[2.3 Preparing to Upgrade 80](#_Toc215243631)

[2.3.1 Deprecated Features 80](#_Toc215243632)

[2.3.2 Discontinued Functionality 82](#_Toc215243633)

[2.3.3 Breaking Changes 84](#_Toc215243634)

[2.3.4 Behavior Changes 84](#_Toc215243635)

[2.3.5 Upgrade Tools 84](#_Toc215243636)

[2.3.6 64-bit Considerations 85](#_Toc215243637)

[2.3.7 Known Issues and Workarounds 85](#_Toc215243638)

[2.4 Upgrading from SQL Server 2000 86](#_Toc215243639)

[2.4.1 Tools Replacement 86](#_Toc215243640)

[2.4.2 Tools Connectivity 87](#_Toc215243641)

[2.4.3 SQL Server Agent Jobs 87](#_Toc215243642)

[2.5 Upgrading from SQL Server 2005 88](#_Toc215243643)

[2.5.1 Tools Replacement 88](#_Toc215243644)

[2.5.2 Tools Connectivity 89](#_Toc215243645)

[2.5.3 Project Files 89](#_Toc215243646)

[2.6 Post-Upgrade Tasks 90](#_Toc215243647)

[2.7 Conclusion 90](#_Toc215243648)

[2.8 Additional References 90](#_Toc215243649)

[3 Relational Databases 92](#_Toc215243650)

[3.1 Introduction 92](#_Toc215243651)

[3.2 Relational Database Configurations 92](#_Toc215243652)

[3.3 Upgrade Considerations 92](#_Toc215243653)

[3.3.1 Full-Text Search 93](#_Toc215243654)

[3.3.2 What Can Be Upgraded? 94](#_Toc215243655)

[3.3.3 What Cannot Be Upgraded? 95](#_Toc215243656)

[3.4 In-Place Upgrade vs. Side-by-Side Upgrade 96](#_Toc215243657)

[3.4.1 In-Place Upgrade 96](#_Toc215243658)

[3.4.2 Side-by-Side Upgrade 97](#_Toc215243659)

[3.5 Evaluating Potential Upgrade Issues 102](#_Toc215243660)

[3.5.1 Deprecated Features 102](#_Toc215243661)

[3.5.2 Discontinued Functionality 102](#_Toc215243662)

[3.5.3 Breaking Changes 103](#_Toc215243663)

[3.5.4 Behavior Changes 105](#_Toc215243664)

[3.6 Preparing for an Upgrade 106](#_Toc215243665)

[3.6.1 Preparing for an In-Place Upgrade 106](#_Toc215243666)

[3.6.2 Preparing for a Side-by-Side Upgrade 111](#_Toc215243667)

[3.7 Performing an Upgrade 112](#_Toc215243668)

[3.7.1 Performing an In-Place Upgrade 112](#_Toc215243669)

[3.7.2 Performing a Side-By-Side Upgrade 112](#_Toc215243670)

[3.8 Post-Upgrade Tasks 114](#_Toc215243671)

[3.8.1 In-Place Upgrade 114](#_Toc215243672)

[3.8.2 Side-by-Side Upgrade 114](#_Toc215243673)

[3.8.3 General Post-Upgrade Tasks 114](#_Toc215243674)

[3.8.4 Use Plan Hints 115](#_Toc215243675)

[3.8.5 Important Information About Query Plans 116](#_Toc215243676)

[3.9 Connecting Client Applications to SQL Server 2008 116](#_Toc215243677)

[3.10 Conclusion 117](#_Toc215243678)

[3.11 Additional References 117](#_Toc215243679)

[4 High Availability 118](#_Toc215243680)

[4.1 Introduction 118](#_Toc215243681)

[4.2 Preparing to Upgrade 118](#_Toc215243682)

[4.2.1 In-Place Upgrade 118](#_Toc215243683)

[4.2.2 Side-by-Side Upgrade on the Same Server or Cluster 119](#_Toc215243684)

[4.2.3 Side-by-Side Upgrade to a Separate Server or Cluster 120](#_Toc215243685)

[4.2.4 Decommissioning and Disabling the Original Instance or Database in a Side-by-Side Upgrade 120](#_Toc215243686)

[4.2.5 Methods for Side-by-Side Upgrades to a Separate Server or Cluster 120](#_Toc215243687)

[4.2.6 Which SQL Server Upgrade Method Should You Use? 122](#_Toc215243688)

[4.3 Minimizing Downtime 122](#_Toc215243689)

[4.3.1 Prepare for SQL Server 2008 122](#_Toc215243690)

[4.3.2 Devise an Upgrade Plan 123](#_Toc215243691)

[4.3.3 Test the Upgrade Plan 124](#_Toc215243692)

[4.3.4 Prepare Servers and Instances for SQL Server 2008 124](#_Toc215243693)

[4.4 Upgrading Failover Clusters 128](#_Toc215243694)

[4.4.1 Feature Changes in SQL Server 2008 Failover Clustering 128](#_Toc215243695)

[4.4.2 Instance IDs and Failover Clustering 129](#_Toc215243696)

[4.4.3 Windows Server 2008 and SQL Server 2008 Failover Clustering 129](#_Toc215243697)

[4.4.4 Considerations for Upgrading a SQL Server 2000 Failover Cluster to SQL Server 2008 132](#_Toc215243698)

[4.4.5 Considerations for Upgrading a SQL Server 2005 Failover Cluster to SQL Server 2008 133](#_Toc215243699)

[4.4.6 Upgrading a SQL Server 2000 or SQL Server 2005 Failover Cluster to SQL Server 2008 133](#_Toc215243700)

[4.4.7 Additional References for Clustering Upgrades 147](#_Toc215243701)

[4.5 Upgrading Log Shipped Databases 147](#_Toc215243702)

[4.5.1 Feature Changes in SQL Server 2008 Log Shipping 147](#_Toc215243703)

[4.5.2 Log Shipping Upgrade Scenarios 148](#_Toc215243704)

[4.5.3 Upgrading from SQL Server 2000 to SQL Server 2008 Log Shipping 149](#_Toc215243705)

[4.5.4 Upgrading from SQL Server 2005 Log Shipping to SQL Server 2008 Log Shipping 154](#_Toc215243706)

[4.5.5 Upgrade log shipping with a role change 155](#_Toc215243707)

[4.5.6 Upgrade log shipping without a role change 156](#_Toc215243708)

[4.5.7 Upgrading with Multiple Secondaries 158](#_Toc215243709)

[4.5.8 Additional References for Upgrading Log Shipping 158](#_Toc215243710)

[4.6 Upgrading Mirrored Databases 159](#_Toc215243711)

[4.6.1 Feature Changes in SQL Server 2008 Database Mirroring 159](#_Toc215243712)

[4.6.2 In-Place Upgrade – High-Performance Mode 159](#_Toc215243713)

[4.6.3 In-Place Upgrade – High-Safety Mode 161](#_Toc215243714)

[4.6.4 Side-by-Side Upgrade to a New Server 161](#_Toc215243715)

[4.6.5 Additional References for Upgrading with Mirrored Databases 162](#_Toc215243716)

[4.7 Upgrading Replicated Databases 162](#_Toc215243717)

[4.7.1 Feature Changes in SQL Server 2008 Replication 162](#_Toc215243718)

[4.7.2 Snapshot Replication 164](#_Toc215243719)

[4.7.3 Merge Replication 168](#_Toc215243720)

[4.7.4 Transactional Replication 173](#_Toc215243721)

[4.7.5 Additional Information for Upgrading Replicated Databases 178](#_Toc215243722)

[4.7.6 Upgrade Considerations for SQL Server Analysis Services 179](#_Toc215243723)

[4.7.7 Methods for New Hardware and Side-by-Side Upgrades 179](#_Toc215243724)

[4.8 Upgrading Failover Clusters 180](#_Toc215243725)

[4.8.1 Considerations for Upgrading a SQL Server 2000 Failover Cluster to SQL Server 2008 181](#_Toc215243726)

[4.8.2 Considerations for Upgrading a SQL Server 2005 Failover Cluster to SQL Server 2008 181](#_Toc215243727)

[4.8.3 Additional References for Clustering Upgrades 181](#_Toc215243728)

[4.9 Conclusion 181](#_Toc215243729)

[4.10 Additional References 181](#_Toc215243730)

[5 Database Security 182](#_Toc215243731)

[5.1 Introduction 182](#_Toc215243732)

[5.2 New Security Features 182](#_Toc215243733)

[5.2.1 New Configuration Tools 183](#_Toc215243734)

[5.2.2 Configuring Services and Connections 183](#_Toc215243735)

[5.2.3 Service Account Security 185](#_Toc215243736)

[5.2.4 Configuring Features 186](#_Toc215243737)

[5.2.5 Metadata Visibility 187](#_Toc215243738)

[5.3 Preparing to Upgrade 187](#_Toc215243739)

[5.3.1 Deprecated Features 188](#_Toc215243740)

[5.3.2 Discontinued Features 189](#_Toc215243741)

[5.3.3 Breaking Changes 190](#_Toc215243742)

[5.3.4 Behavior Changes 191](#_Toc215243743)

[5.3.5 Pre-Upgrade Security Tasks 194](#_Toc215243744)

[5.4 Upgrading to SQL Server 2008 196](#_Toc215243745)

[5.4.1 In-Place Upgrade 196](#_Toc215243746)

[5.4.2 Side-by-Side Upgrade 196](#_Toc215243747)

[5.5 Post-Upgrade Security Tasks 197](#_Toc215243748)

[5.5.1 Post-Upgrade Security Testing 197](#_Toc215243749)

[5.6 Conclusion 198](#_Toc215243750)

[5.7 Additional References 198](#_Toc215243751)

[6 Full-Text Search 199](#_Toc215243752)

[6.1 Introduction 199](#_Toc215243753)

[6.2 Preparing to Upgrade 199](#_Toc215243754)

[6.2.1 Deprecated Features 199](#_Toc215243755)

[6.2.2 Discontinued Functionality 200](#_Toc215243756)

[6.2.3 Breaking Changes 201](#_Toc215243757)

[6.2.4 Behavior Changes 201](#_Toc215243758)

[6.2.5 Running Upgrade Advisor 202](#_Toc215243759)

[6.2.6 Preparing for a Possible Rollback 203](#_Toc215243760)

[6.2.7 Additional Preparation Steps When Upgrading from SQL Server 2000 204](#_Toc215243761)

[6.3 Upgrading a Full-Text-Enabled Database 204](#_Toc215243762)

[6.3.1 In-Place Upgrade 204](#_Toc215243763)

[6.3.2 Side-by-Side Upgrade 205](#_Toc215243764)

[6.4 Post-Upgrade Tasks 209](#_Toc215243765)

[6.4.1 Using Customized Noise Word Files from a Previous SQL Server Version 209](#_Toc215243766)

[6.5 Conclusion 209](#_Toc215243767)

[6.6 Additional Resources 210](#_Toc215243768)

[7 Service Broker 211](#_Toc215243769)

[7.1 Introduction 211](#_Toc215243770)

[7.2 Feature Changes 211](#_Toc215243771)

[7.3 Preparing to Upgrade 212](#_Toc215243772)

[7.3.1 Disk Space Requirements 212](#_Toc215243773)

[7.3.2 Upgrade Tools 212](#_Toc215243774)

[7.3.3 64-bit Considerations 213](#_Toc215243775)

[7.4 Upgrading from SQL Server 2005 213](#_Toc215243776)

[7.4.1 In-Place Upgrade 213](#_Toc215243777)

[7.4.2 Side-by-Side Upgrade 214](#_Toc215243778)

[7.5 Post-Upgrade Tasks 215](#_Toc215243779)

[7.5.1 Restoring Settings 215](#_Toc215243780)

[7.5.2 Routing Changes 215](#_Toc215243781)

[7.5.3 Implementing Conversation Priorities 215](#_Toc215243782)

[7.6 Conclusion 216](#_Toc215243783)

[7.7 Additional References 216](#_Toc215243784)

[8 Transact-SQL Queries 217](#_Toc215243785)

[8.1 Introduction 217](#_Toc215243786)

[8.2 Preparing to Upgrade 217](#_Toc215243787)

[8.2.1 Backup and Rollback Plan 217](#_Toc215243788)

[8.2.2 Deprecated Features 218](#_Toc215243789)

[8.2.3 Discontinued Functionality 220](#_Toc215243790)

[8.2.4 Breaking Changes 223](#_Toc215243791)

[8.2.5 Behavior Changes 225](#_Toc215243792)

[8.2.6 Upgrade Tools 237](#_Toc215243793)

[8.2.7 64-Bit Considerations 237](#_Toc215243794)

[8.2.8 Known Issues and Workarounds 237](#_Toc215243795)

[8.3 Upgrading from SQL Server 2000 or SQL Server 2005 242](#_Toc215243796)

[8.3.1 In-Place Upgrade 242](#_Toc215243797)

[8.3.2 Side-by-Side Upgrade 242](#_Toc215243798)

[8.4 Post-Upgrade Tasks 243](#_Toc215243799)

[8.5 Conclusion 243](#_Toc215243800)

[8.6 Additional References 243](#_Toc215243801)

[9 Notification Services 244](#_Toc215243802)

[9.1 Introduction 244](#_Toc215243803)

[9.2 Feature Changes 244](#_Toc215243804)

[9.3 Preparing to Upgrade 244](#_Toc215243805)

[9.3.1 Stop the Notification Services Instances 244](#_Toc215243806)

[9.3.2 Back Up the Instance Data 245](#_Toc215243807)

[9.3.3 Install the Prerequisites 245](#_Toc215243808)

[9.3.4 Install the Notification Services Components 246](#_Toc215243809)

[9.4 Upgrading from SQL Server 2000 246](#_Toc215243810)

[9.5 Upgrading from SQL Server 2005 247](#_Toc215243811)

[9.5.1 In-Place Upgrade 247](#_Toc215243812)

[9.5.2 Side-by-Side Upgrade 248](#_Toc215243813)

[9.6 Post-Upgrade Tasks 250](#_Toc215243814)

[9.7 Conclusion 251](#_Toc215243815)

[9.8 Additional References 251](#_Toc215243816)

[10 SQL Server Express 252](#_Toc215243817)

[10.1 Introduction 252](#_Toc215243818)

[10.2 Feature Changes 252](#_Toc215243819)

[10.3 Preparing to Upgrade 253](#_Toc215243820)

[10.3.1 SQL Server Express and MSDE Limitations 255](#_Toc215243821)

[10.3.2 SQL Server Express and MSDE Feature Support 255](#_Toc215243822)

[10.3.3 Deprecated Features 256](#_Toc215243823)

[10.3.4 Discontinued Functionality 256](#_Toc215243824)

[10.3.5 Breaking Changes 256](#_Toc215243825)

[10.3.6 Behavior Changes 256](#_Toc215243826)

[10.3.7 Upgrade Tools 257](#_Toc215243827)

[10.3.8 64-Bit Considerations 257](#_Toc215243828)

[10.3.9 System Requirements for SQL Server 2008 Express 258](#_Toc215243829)

[10.3.10 Known Issues and Workarounds 260](#_Toc215243830)

[10.4 Upgrading from SQL Server 2000 (MSDE) 261](#_Toc215243831)

[10.4.1 Number of Instances of MSDE 261](#_Toc215243832)

[10.4.2 MSDE Installation Method 261](#_Toc215243833)

[10.4.3 MSDE Language 262](#_Toc215243834)

[10.4.4 In-Place Upgrade 262](#_Toc215243835)

[10.4.5 Side-by-Side Upgrade 263](#_Toc215243836)

[10.4.6 Performing Scripted Upgrades 265](#_Toc215243837)

[10.5 Upgrading from SQL Server 2005 Express 266](#_Toc215243838)

[10.5.1 In-Place Upgrade 266](#_Toc215243839)

[10.5.2 Side-by-Side Upgrade 266](#_Toc215243840)

[10.6 Post-Upgrade Tasks 268](#_Toc215243841)

[10.7 Upgrading to Other Editions of SQL Server 2008 268](#_Toc215243842)

[10.7.1 Upgrading to SQL Server 2008 Workgroup 269](#_Toc215243843)

[10.7.2 Upgrading to SQL Server 2008 Standard Edition 270](#_Toc215243844)

[10.8 Conclusion 270](#_Toc215243845)

[10.9 Additional References 270](#_Toc215243846)

[11 Analysis Services 271](#_Toc215243847)

[11.1 Introduction 271](#_Toc215243848)

[11.2 Preparing to Upgrade 271](#_Toc215243849)

[11.2.1 In-Place Upgrade vs. Side-by-Side Upgrade 271](#_Toc215243850)

[11.2.2 Determining and Evaluating Potential Upgrade Issues 272](#_Toc215243851)

[11.2.3 Issues Preventing an Upgrade 273](#_Toc215243852)

[11.2.4 Deprecated Features 273](#_Toc215243853)

[11.2.5 Discontinued Functionality 273](#_Toc215243854)

[11.2.6 Breaking Changes 274](#_Toc215243855)

[11.2.7 Behavior Changes 279](#_Toc215243856)

[11.2.8 64-bit Considerations 279](#_Toc215243857)

[11.3 Upgrading from SQL Server 2000 280](#_Toc215243858)

[11.3.1 In-Place Upgrade 281](#_Toc215243859)

[11.3.2 Side-by-Side Upgrade 289](#_Toc215243860)

[11.3.3 Redesigning Databases for SSAS 2008 295](#_Toc215243861)

[11.4 Upgrading from SQL Server 2005 297](#_Toc215243862)

[11.4.1 In-Place Upgrade 298](#_Toc215243863)

[11.5 Performing Post-Upgrade Tasks 301](#_Toc215243864)

[11.6 Conclusion 301](#_Toc215243865)

[11.7 Additional References 302](#_Toc215243866)

[12 Data Mining 303](#_Toc215243867)

[12.1 Introduction 303](#_Toc215243868)

[12.2 Data Mining Features 303](#_Toc215243869)

[12.3 Preparing to Upgrade 305](#_Toc215243870)

[12.3.1 Deprecated Features 305](#_Toc215243871)

[12.3.2 Discontinued Functionality 306](#_Toc215243872)

[12.3.3 Breaking Changes 306](#_Toc215243873)

[12.3.4 Behavior Changes 307](#_Toc215243874)

[12.3.5 Running Upgrade Advisor 307](#_Toc215243875)

[12.4 Upgrading from SQL Server 2000 310](#_Toc215243876)

[12.4.1 In-Place Upgrade 310](#_Toc215243877)

[12.4.2 Side-by-Side Upgrade 313](#_Toc215243878)

[12.4.3 Post-Upgrade Tasks 314](#_Toc215243879)

[12.5 Upgrading from SQL Server 2005 318](#_Toc215243880)

[12.5.1 In-Place Upgrade 319](#_Toc215243881)

[12.5.2 Side-by-Side Upgrade 321](#_Toc215243882)

[12.5.3 Post-Upgrade Tasks 323](#_Toc215243883)

[12.6 Conclusion 328](#_Toc215243884)

[12.7 Additional References 329](#_Toc215243885)

[13 Integration Services 330](#_Toc215243886)

[13.1 Introduction 330](#_Toc215243887)

[13.2 Preparing to Upgrade to SSIS 2008 330](#_Toc215243888)

[13.2.1 Preparing to Upgrade from DTS 331](#_Toc215243889)

[13.2.2 Deprecated Features 335](#_Toc215243890)

[13.2.3 Discontinued Functionality 336](#_Toc215243891)

[13.2.4 Breaking Changes 336](#_Toc215243892)

[13.2.5 Behavior Changes 337](#_Toc215243893)

[13.2.6 Upgrade Tools 337](#_Toc215243894)

[13.2.7 Coexistence with Previous Versions 345](#_Toc215243895)

[13.2.8 64-bit Considerations 346](#_Toc215243896)

[13.2.9 Data Providers 349](#_Toc215243897)

[13.2.10 Failover Clustering 350](#_Toc215243898)

[13.2.11 Known Issues and Workarounds 350](#_Toc215243899)

[13.3 Upgrading from SQL Server 2000 356](#_Toc215243900)

[13.3.1 Migrating DTS Packages to SSIS 356](#_Toc215243901)

[13.3.2 Comparing DTS and SSIS Functionality 356](#_Toc215243902)

[13.3.3 Running the DTS Migration Wizard 359](#_Toc215243903)

[13.3.4 DTS Migration Examples 372](#_Toc215243904)

[13.4 Upgrading from SQL Server 2005 403](#_Toc215243905)

[13.5 Post-Upgrade Tasks 413](#_Toc215243906)

[13.5.1 DTS Post-Migration Tasks 413](#_Toc215243907)

[13.5.2 SSIS 2005 Post-Upgrade Tasks 414](#_Toc215243908)

[13.6 Conclusion 420](#_Toc215243909)

[13.7 Additional References 420](#_Toc215243910)

[14 Reporting Services 421](#_Toc215243911)

[14.1 Introduction 421](#_Toc215243912)

[14.1.1 Reporting Services Configurations 421](#_Toc215243913)

[14.1.2 Reporting Services Editions 423](#_Toc215243914)

[14.1.3 Upgrade Considerations 424](#_Toc215243915)

[14.1.4 In-Place Upgrade vs. Side-by-Side Upgrade 426](#_Toc215243916)

[14.1.5 Preparing to Upgrade 427](#_Toc215243917)

[14.1.6 Important Reporting Services 2000 Configuration Files 428](#_Toc215243918)

[14.1.7 Storing Configuration Settings 429](#_Toc215243919)

[14.1.8 Deprecated Features 430](#_Toc215243920)

[14.1.9 Discontinued Functionality 431](#_Toc215243921)

[14.1.10 Breaking Changes 431](#_Toc215243922)

[14.1.11 Behavior Changes 436](#_Toc215243923)

[14.1.12 Upgrade Tools 441](#_Toc215243924)

[14.1.13 64-bit Considerations 442](#_Toc215243925)

[14.1.14 Known Issues and Workarounds 443](#_Toc215243926)

[14.1.15 Backup and Rollback Plan 444](#_Toc215243927)

[14.2 Upgrading from SQL Server 2000 445](#_Toc215243928)

[14.2.1 In-Place Upgrade 445](#_Toc215243929)

[14.2.2 Side-by-Side Upgrade 451](#_Toc215243930)

[14.3 Upgrading from SQL Server 2005 460](#_Toc215243931)

[14.3.1 In-Place Upgrade 460](#_Toc215243932)

[14.3.2 Side-by-Side Upgrade 461](#_Toc215243933)

[14.4 Troubleshooting a Failed Upgrade 462](#_Toc215243934)

[14.5 Post-Upgrade Tasks 463](#_Toc215243935)

[14.5.1 Deploying Custom Extensions and Assemblies 465](#_Toc215243936)

[14.5.2 Verifying Configuration Files 466](#_Toc215243937)

[14.5.3 Uninstalling SSRS 2000 or SSRS 2005 467](#_Toc215243938)

[14.6 Conclusion 468](#_Toc215243939)

[14.7 Additional Resources 468](#_Toc215243940)

[15 Other Microsoft Applications and Platforms 469](#_Toc215243941)

[15.1 Introduction 469](#_Toc215243942)

[15.2 Microsoft Windows Small Business Server 2008 469](#_Toc215243943)

[15.2.1 Preparing to Migrate to Windows SBS 2008 469](#_Toc215243944)

[15.2.2 Migrating to Windows SBS 2008 470](#_Toc215243945)

[15.2.3 Post-Migration Tasks 471](#_Toc215243946)

[15.3 Microsoft Office Communications Server 471](#_Toc215243947)

[15.3.1 Preparing to Upgrade to OCS 2007 R2 471](#_Toc215243948)

[15.3.2 Upgrading OCS 2007 R2 to Use SQL Server 2008 472](#_Toc215243949)

[15.3.3 Rollback Options and Tools 473](#_Toc215243950)

[15.4 Microsoft Office SharePoint Server 2007 473](#_Toc215243951)

[15.4.1 Preparing to Upgrade MOSS 2007 to SQL Server 2008 473](#_Toc215243952)

[15.5 Microsoft System Center 2007 474](#_Toc215243953)

[15.5.1 Operations Manager: Management Pack for SQL Server 2008 475](#_Toc215243954)

[15.5.2 Data Protection Manager 475](#_Toc215243955)

[15.6 Microsoft Dynamics 477](#_Toc215243956)

[15.7 Conclusion 477](#_Toc215243957)

[15.8 Additional References 477](#_Toc215243958)

[16 Appendix 1: Version and Edition Upgrade Paths 478](#_Toc215243959)

[17 Appendix 2: Upgrade Planning Deployment and Tasks Checklist 481](#_Toc215243960)

**Overview**

A successful upgrade to SQL Server 2008 should be smooth and trouble-free. To achieve that smooth transition, you must devote plan sufficiently for the upgrade, and match the complexity of your database application. Otherwise, you risk costly and stressful errors and upgrade problems.

Like all IT projects, planning for every contingency and then testing your plan gives you confidence that you will succeed. But if you ignore the planning process, you increase the chances of running into difficulties that can derail and delay your upgrade.

This document covers the essential phases and steps involved in upgrading existing SQL Server 2000 and 2005 instances to SQL Server 2008 by using best practices. These include preparation tasks, upgrade tasks, and post-upgrade tasks.

For the purpose of this document, an “upgrade” is any type of transition from an earlier version of SQL Server to SQL Server 2008, a “server” is a computer that is Windows Server (either physical or virtual), and a “component” is one of the several relatively independent executables included within SQL Server, such as the Database Engine, SQL Server High Availability Solutions, SQL Server Analysis Services, SQL Server Integration Services, SQL Server Reporting Services, and SQL Server Notification Services.

This document is intended to be a supplement to SQL Server 2008 Books Online. It is not intended to supersede any information in SQL Server Books Online or in Microsoft Knowledge Base articles. The reader will notice many links to SQL Server Books Online topics and Knowledge Base articles. In all such cases, the information in this document is included to provide you enough context to decide whether you need to read the linked article. If there are any discrepancies between this document and a linked article, the linked article is assumed to be more accurate.

* Chapter 1 gives an overview of the technical issues and decisions that are involved in an upgrade to SQL Server 2008, as well as recommendations for planning and deploying an upgrade.
* Chapter 2 addresses issues related to upgrading to SQL Server 2008 Management Tools.
* Chapters 3 through 8 focus on upgrade issues for SQL Server relational databases.
* Chapter 9 addresses upgrading to SQL Server 2008 Express.
* Chapters 10 through 14 focus on upgrading to SQL Server 2008 Business Intelligence components: Analysis Services, Data Mining, Integration Services, and Reporting Services.
* Chapter 15 addresses the implications of upgrading to SQL Server 2008 for other Microsoft applications and platforms.
* Appendix 1 contains a table of allowed SQL Server 2008 version and edition upgrade paths.
* Appendix 2 contains an upgrade planning checklist.

After this document is published, Microsoft may post updates, corrections or clarifications in a companion Knowledge Base article. The reader is encouraged to review the [SQL Server 2008 Upgrade Technical Reference Guide KB Landing Page](http://msdn.microsoft.com/en-us/library/cc936623.aspx) (KB 936623) for any such updates.

# Upgrade Planning and Deployment

## Introduction

This first chapter lays out the general guidelines for planning a successful upgrade to SQL Server 2008. These guidelines include upgrade strategies, test and rollback considerations, and upgrade tools. This chapter also introduces you to the SQL Server 2008 Upgrade Advisor, a tool that analyzes legacy instances of SQL Server 2000 and SQL Server 2005 and flags potential problems that you must address before upgrading.

The remaining chapters in this guide provide technical detail about how to upgrade specific SQL Server components and scenarios.

## Feature Changes in SQL Server 2008

SQL Server 2008 contains improvements and additional features in almost every area of the product. In fact, any one of these improved features can be a compelling case for upgrading, depending on your need for high availability, performance, and additional functionality. Additionally, upgrading to the latest release of the product extends the Microsoft support life cycle to the maximum degree possible, according to the software support policy.

SQL Server 2008 new features and improvements fall into three categories:

* **Trusted**: SQL Server 2008 provides the highest levels of security, reliability, and scalability for your business-critical applications.
* **Productive**: SQL Server 2008 reduces the time and cost required to manage and develop database applications.
* **Intelligent**: SQL Server 2008 provides a comprehensive platform for delivering business intelligence (BI) solutions where users want them.

To better understand the SQL Server 2008 features that make upgrading helpful, see the [SQL Server 2008 Product Overview](http://go.microsoft.com/fwlink/?LinkID=105937) white paper.

## Preparing to Upgrade

To prepare for an upgrade, begin by collecting information about the effect of the upgrade and the risks it might involve. When you identify the risks up front, you can determine how to lessen and manage them throughout the upgrade process.

Upgrade scenarios will be as complex as your underlying applications and instances of SQL Server. Some scenarios within your environment might be simple, other scenarios complex. Start to plan by analyzing upgrade requirements, including reviewing upgrade strategies, understanding SQL Server 2008 hardware and software requirements, and discovering any blocking problems caused by backward-compatibility issues.

### Upgrade Strategies

An upgrade is any kind of transition from SQL Server 2000 or SQL Server 2005 to SQL Server 2008. There are two fundamental strategies for upgrading, with two main variations in the second strategy:

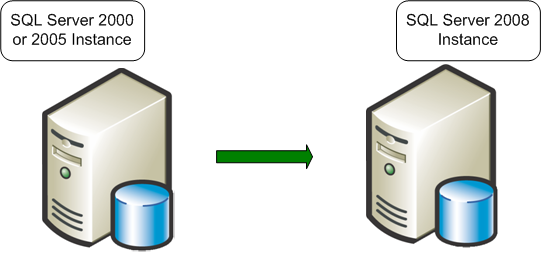
* **In-place upgrade:** Using the SQL Server 2008 Setup program to directly upgrade an instance of SQL Server 2000 or SQL Server 2005 to SQL Server 2008. The older instance of SQL Server is replaced.
* **Side-by-side upgrade: Using steps to move all or some data from an instance of SQL Server 2000 or SQL Server 2005 to a separate instance of SQL Server 2008.** There are two main variations of the side-by-side upgrade strategy:
  + **One server**: The new instance exists on the same server as the target instance.
  + **Two servers**: The new instance exists on a different server than the target instance.

#### In-Place Upgrade

By using an *in-place* upgrade strategy, the SQL Server 2008 Setup program directly replaces an instance of SQL Server 2000 or SQL Server 2005 with a new instance of SQL Server 2008 on the same x86 or x64 platform. (An in-place upgrade requires that the old and new instances of SQL Server be on the same x86 or x64 platform, see the note in "Extended System Support (WOW64)" later in this chapter.) This kind of upgrade is called "in-place" because the upgraded instance of SQL Server 2000 or SQL Server 2005 is actually replaced by the new instance of SQL Server 2008. You do not have to copy database-related data from the older instance to SQL Server 2008 because the old data files are automatically converted to the new format. When the process is complete, the old instance of SQL Server 2000 or SQL Server 2005 is removed from the server, with only the backups that you retained being able to restore it to its previous state.

**Note**: If you want to upgrade just one database from a legacy instance of SQL Server and not upgrade the other databases on the server, use the side-by-side upgrade method instead of the in-place method.

Figure 1-1 shows the before and after states of an in-place upgrade.



**Figure 1-1:** In an in-place upgrade, SQL Server 2008 Setup replaces a legacy instance of SQL Server 2000 or SQL Server 2005 with a new instance of SQL Server 2008.

Note the following restrictions on an in-place upgrade:

* SQL Server 2008 Setup requires that all SQL Server 2000 and SQL Server 2005 components be upgraded together. SQL Server 2008 Setup will detect all the components of the instance to be upgraded and will require that they all be upgraded immediately. In other words, you cannot upgrade only an instance of the SQL Server 2005 Database Engine without also upgrading the Analysis Services component.
* A cross-platform, in-place upgrade from a 32-bit instance of SQL Server 2000 or SQL Server 2005 (x86) to a 64-bit instance of SQL Server 2008 (x64), or vice versa, is not supported. For more information, see the note in "Extended System Support (WOW64)" later in this chapter.

Here are the major steps that the SQL Server 2008 Setup program takes when you perform an in-place upgrade:

1. The SQL Server 2008 Setup prerequisites—Microsoft .NET Framework 3.5 Service Pack 1 (SP1) or a later version, SQL Server Native Client, and so on—are installed. The legacy instance databases continue to be available.
2. Setup checks for upgrade blocking issues, a small set of issues that will completely block an upgrade. If it finds any, Setup will list them. You must fix them and restart the upgrade process.
3. Setup installs the required SQL Server 2008 executables and support files.
4. Setup stops the legacy SQL Server service. At this point, the legacy instance is no longer available.
5. SQL Server 2008 updates the selected component data and objects.
6. Setup removes the legacy executables and support files in addition to the legacy SQL Server 2000 tools. SQL Server 2005 tools are not removed (see Chapter 2, "Management and Development Tools," for more information).

The new instance of SQL Server 2008 is now fully available. The legacy instance of SQL Server 2000 or SQL Server 2005 is now gone, and can be restored only from a backup if the need arises.

#### Side-by-Side Upgrade

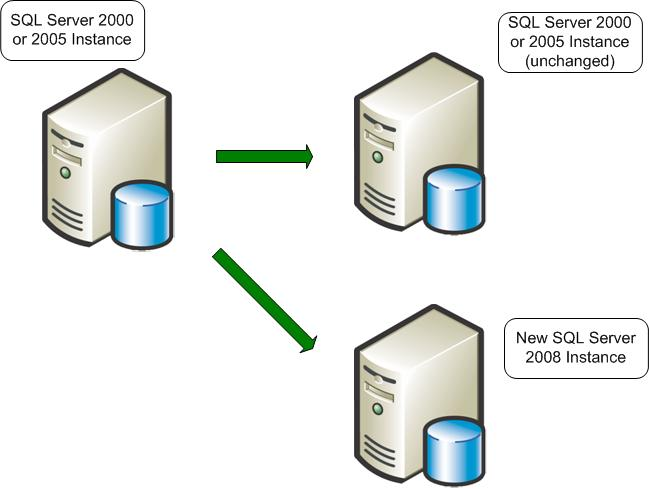
In a *side-by-side* upgrade, instead of directly replacing the older instance of SQL Server, required database and component data is transferred from an instance of SQL Server 2000 or SQL Server 2005 to a separate instance of SQL Server 2008. It is called a "side-by-side" method because the new instance of SQL Server 2008 runs alongside the legacy instance of SQL Server 2000 or SQL Server 2005, on the same server or on a different server.

There are two important options when you use the side-by-side upgrade method:

* You can transfer data and components to an instance of SQL Server 2008 that is located on a different physical server or on a different virtual machine, or
* You can transfer data and components to an instance of SQL Server 2008 on the same physical server

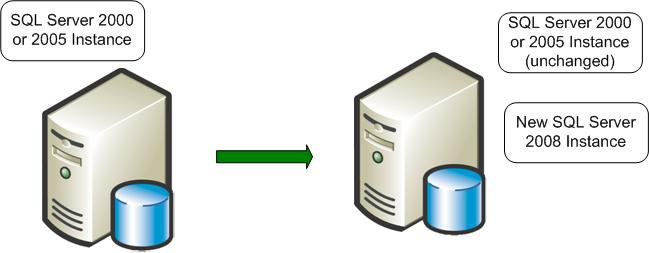
Both options let you run the new instance of SQL Server 2008 alongside the legacy instance of SQL Server 2000 or SQL Server 2005. Typically, after the upgraded instance is accepted and moved into production, you can remove the older instance.

Figure 1-2 shows the before and after states of a side-by-side upgrade on two servers.



**Figure 1-2:** A side-by-side upgrade to another server leaves the legacy instance of SQL Server 2000 or SQL Server 2005 unchanged.

You can also use the side-by-side method to upgrade to SQL Server 2008 on the same server as the legacy instance of SQL Server 2000 or SQL Server 2005. Figure 1-3 shows a side-by-side upgrade on the same server.



**Figure 1-3:** You can perform a side-by-side upgrade on the same server, leaving both instances running.

Whether a side-by-side upgrade is to a separate instance on the same server or to a new instance on another server, data must be transferred in what is mostly a manual process. The result is two instances, legacy and new, that can run side by side.

As just noted, the key point in a side-by-side upgrade is that you must manually transfer data files and other supporting objects from the older instance of SQL Server to the instance of SQL Server 2008. The SQL Server 2008 Setup program will not perform this task. The objects that you must transfer include the following:

* Data files
* Database objects
* SSAS cubes
* Configuration settings
* Security settings
* SQL Server Agent jobs
* SSIS packages

Here are the main steps that you must perform when doing a side-by-side upgrade of SQL Server 2000 or SQL Server 2005 to SQL Server 2008:

1. Install a separate instance of SQL Server 2008 on the legacy server or on a separate server. The legacy instance continues to be available.
2. Run the SQL Server 2008 Upgrade Advisor against the legacy instance, and remove any upgrade blocker issues it finds.
3. Stop all update activity to the legacy instance. This might involve disconnecting all users or forcing applications to read-only activity.
4. Transfer data, packages, and other objects from the legacy instance to the instance of SQL Server 2008.
5. Apply supporting objects such as SQL Server Agent jobs, security settings, and configuration settings to the new instance of SQL Server 2008.
6. Upgrade SSIS (and potentially Data Transformation Services—DTS) packages to SSIS (see Chapter 13, "Integration Services," for more information).
7. Verify that the new instance supports the required applications by using validation scripts and user-acceptance tests.
8. If the new instance passes validation and acceptance tests, redirect applications and users to the new instance. At this point, the new instance is available and databases are online.
9. Keep your legacy instance for data recovery until you are absolutely confident that no problems exist on your new production database instance.

A side-by-side upgrade to a new server offers the best of both worlds: You can take advantage of a new and potentially more powerful server and platform, but the legacy server remains as a fallback if you encounter a problem. This method could also potentially reduce an upgrade downtime by letting you have the new server and instances tested, up, and running without affecting a current server and its workloads. You can test and address hardware or software problems encountered in bringing the new server online, without downtime of the legacy system. Although you would have to find a way to export data out of the new system to go back to the old system, rolling back to the legacy system would still be less time-consuming than a full SQL Server reinstall and restoring the databases, which a failed in-place upgrade would require. The downside of a side-by-side upgrade is that increased manual interventions are required, so it might take more preparation time by an upgrade/operations team. However, the benefits of this degree of control can frequently be worth the additional effort.

#### Comparing In-Place and Side-by-Side Methods

Table 1-1 summarizes the difference between the two upgrade strategies.

Table 1-1: Characteristics of an In-Place Upgrade vs. a Side-by-Side Upgrade

|  |  |  |
| --- | --- | --- |
| **Process** | **In-Place Upgrade** | **Side-by-Side Upgrade** |
| **Number of resulting instances** | One only | Two |
| **Number of physical servers involved** | One | One or more |
| **Data file transfer** | Automatic | Manual |
| **SQL Server instance configuration** | Automatic | Manual |
| **Supporting tool** | SQL Server Setup | Several data transfer methods |

Be aware that the main difference between an in-place upgrade and a side-by-side upgrade depends on the resulting instances. An in-place upgrade replaces the old instance so that only one instance remains.

Another way to view the differences between an in-place upgrade and a side-by-side upgrade is to focus on how much of the legacy instance you want to upgrade. Table 1-2 shows how you can use the component level of the upgrade, combined with the resulting number of instances, to determine what upgrade strategies are available for your needs.

Table 1-2: Upgrade Strategies and Components

|  |  |  |
| --- | --- | --- |
| **Component Level** | **Single Resulting instance of SQL Server 2008** | **Two Resulting Instances** |
| All components | In-place | Side-by-side |
| Single component | In-place | Side-by-side |
| Single database | Not available | Side-by-side |

Overall advantages of an in-place upgrade include the following:

* An in-place upgrade can be easier and faster, especially for small systems, because data and configuration options do not have to be manually transferred to a new server.
* It is mostly an automated process.
* The resulting upgraded instance has the same name as the original.
* Applications continue to connect to the same instance name.
* No additional hardware is required because only the one instance is involved. However additional disk is required by Setup (see "Setup Requirements for an In-Place Upgrade" in the "SQL Server 2008 Setup" section later in this chapter).
* Because it is mostly automated, it also takes the least deployment team resources.

Some overall disadvantages of an in-place upgrade include the following:

* You must upgrade the whole instance or a major SQL Server component. For example, you cannot directly upgrade a single database.
* You must inspect the whole instance for backward-compatibility issues and address any blocking issues before SQL Server 2008 Setup can continue.
* Upgrading in place is not recommended for all SQL Server components, such as some DTS packages. See Chapter 13, "Integration Services," for more information about how to upgrade DTS packages. In addition, Notification Services cannot be upgraded in place (see Chapter 9, "Notification Services").
* Because the new instance of SQL Server 2008 replaces the legacy instance, you cannot run the two instances side by side to compare them. Instead, you should use a test environment for comparisons.
* Rollback of upgraded data and the upgraded instance in an in-place upgrade can be complex and time-consuming. See "Rolling Back an Upgrade" in the next section for more information.

Overall advantages of a side-by-side upgrade include the following:

* It gives more granular control over which database objects are upgraded.
* The legacy database server can run alongside the new server. You can perform test upgrades and research and resolve compatibility issues without disturbing the production system.
* The legacy database server remains available during the upgrade, although it cannot be updated for at least the time that is required to transfer data.
* Users can be moved from the legacy system in a staged manner instead of all at the same time. Even though your system might have passed all validation and acceptance tests, a problem could still occur. But if a problem does occur, you will be able to roll back to the legacy system.

Overall disadvantages of a side-by-side upgrade include the following:

* A side-by-side upgrade might require new or additional hardware resources.
* If the side-by-side upgrade occurs on the same server, there might be insufficient resources to run both instances alongside one another.
* Applications and users must be redirected to a new instance. This redirection might require some recoding in the application.
* You must manually transfer data—as well as security, configuration settings, and other supporting objects—to the new instance.
* Synchronization of data from the legacy server to a new server will be required to capture data modifications that occurred to the legacy system while setting up the new system and its original copy of the data.

#### Summary of Factors Affecting the Upgrade Strategy Decision

Sometimes it is expediency, disk space, new server hardware, or high availability considerations that will help you decide which upgrade strategy to use. Use your best judgment to decide which, because there are no simple rules to follow. Table 1-3 is intended to give you some guidelines for your consideration as you make your decisions. And be aware that you might decide to upgrade some of your instances in-place and other instances side-by-side, depending on your organization’s needs. Many of these factors are discussed in more detail later in this chapter.

Table 1-3: Summary of Factors Affecting the Upgrade Strategy Decision

| **Consideration** | **In-Place Upgrade Advantages** | **Side-by-Side Upgrade Advantages** |
| --- | --- | --- |
| Require the fewest hours for the upgrade deployment team to plan and prepare the upgrade effort | * Setup automatically upgrades data and settings in place, without the need for a manual transfer of data or settings. * The resulting upgraded instance has the same name as the original. * Applications continue to connect to the same instance name. | * It gives more granular control over which database objects are migrated. * The legacy database server can run alongside the new server. You can perform test migrations and research and resolve compatibility issues without disturbing the production system. * The legacy database server remains available during the migration, although it cannot be updated for at least the time that is required to transfer data. * Applications can be moved from legacy system in a staged manner instead of all at the same time. Even though your system might have passed all validation and acceptance test, a problem could still occur (see Murphy’s Law). If a problem does occur, then you will be able to roll back to the legacy system. |
| Require minimal DBA skill set expertise (or no DBA available to implement the upgrade) | A junior DBA or System Engineer or similar with basic knowledge and adherence to good IT practices can complete upgrades without special scripts or manual interventions; Setup automates the upgrade process. | System recovery: If a junior DBA or anybody who is not familiar with the upgrade process has any problems, the production environment remains untouched. Only when the new system is approved by Test / QA will the users be migrated to the new server. |
| Require minimal user downtime | For small data sets, the total end-to-end time might be smaller because this is the most automated upgrade strategy. | By controlling the steps directly, you can do much of the preparation including much of the data transfer can be completed without user downtime; some user downtime is still required to update the instance version. |
| Require fastest possible revert/rollback in case issue encountered |  | The legacy instance of SQL Server is still present at the end of the upgrade and may be used as a rollback option. |
| Schedule different downtime windows for different user databases within the same instance |  | Can separately control when each user database is upgraded; after the last user database is upgraded, the legacy instance can be removed. |
| Preserve the Server and Instance name | Preserves the same server and instance name. |  |
| Server consolidation project |  | The upgraded instance can be placed upon the consolidation server, after which the old server can be taken out of service. |
| Applications required to run in parallel with the original and new instances of SQL Server |  | Can maintain both systems with production transactions until ready to turn off the original system. |
| New server hardware or  operating system |  | The upgraded instance can be put upon the new server, after which the old server can be taken out of service. |
| Shortage of disk space in production | Because there is only one resulting instance, there is less additional dataspace required than is possible with two complete resulting instances running side by side. |  |
| Legacy Server cannot meet the SQL Server 2008 Setup requirements for installation | None - not supported . | Only possible with side-by-side upgrade to a new server which meets setup requirements. |
| Changing to a earlier edition of SQL Server 2008 | None - not supported. | Only possible with side-by-side upgrade. |
| Upgrade a 32-bit version of SQL Server 2000 or 2005 to a 64-bit version of SQL Server 2008 | None - not supported . | Only possible with side-by-side upgrade. |
| Upgrade a 64-bit version of SQL Server 2000 or 2005 to a 32-bit version of SQL Server 2008. | None - not supported. | Only possible with side-by-side upgrade. |
| Target instance is SQL Server 2000 SP3a and only one downtime window is available |  | Transferring data to a new instance may be faster than the two steps required to apply the required SQL Server 2000 SP4 and then upgrading to SQL Server 2008. |
| Upgrading from a nonclustered legacy instance of SQL Server to a clustered instance of SQL Server 2008 | None - not supported. | Only possible with side-by-side upgrade. |
| Upgrading from SQL Server 7.0 | None - not supported; would need first to upgrade to SQL Server 2000 or 2005. | Can be done in one direct upgrade by using manual data transfer. |
| Upgrading multiple instances | Generally faster because data transfer and configuration steps are handled by Setup. |  |
| Upgrading very large databases (VLDBs) | Setup converts existing data files automatically; no data transfer steps are required. | Can control the timing of several steps and also the rollback if it is necessary. |
| Upgrade Testing |  | Re-testing might be easier because the legacy instance of SQL Server does not have to be rebuilt.  Testing can be done while legacy system still supports production applications. SQL Server Profiler can be used to capture SQL commands against the legacy system and used to play back against the new instance of SQL Server to verify that everything is working well. |
| Data must be transformed during the upgrade window |  | Data transformation tools such as SSIS can be used to transform data as it is being transferred from the legacy instance of SQL Server to SQL Server 2008. |
| Localization: change of SQL Server language |  | Enables upgrade to SQL Server 2008 with a different language from the legacy instance of SQL Server. Note: this issue is not to be confused with collation settings. This applies to the localized language of the SQL Server product. |
| Upgrading Notification Services | None - not supported. | Only possible with side-by-side upgrade. Requires Notification Services backward compatibility add-in. |
| Application integration | Simpler because the same server name, instance name, and database security settings are preserved by Setup, without manual intervention. | Applications can be moved from legacy system in a staged manner instead of all at the same time. Even though your system might have passed all validation and acceptance test, a problem could still occur. If a problem does occur, then you will be able to roll back to the legacy system. |
| Server Integration | Might be simpler because linked server, replication, and log shipping settings can be preserved, depending on the SQL Server version being upgraded. |  |

#### Rolling Back an Upgrade

When you evaluate which upgrade strategy to use, consider the risk that an in-place upgrade or side-by-side upgrade might have to be rolled back. The complexity and effort required to roll back is an important factor in selecting which method to use.

Rolling back an in-place upgrade can be complex and time-consuming. The new data file structures for SQL Server 2008 are incompatible with legacy instances of SQL Server 2000 and SQL Server 2005. Because the new instance of SQL Server 2008 replaces the legacy instance of SQL Server in an in-place upgrade, to roll back an upgraded instance, you must uninstall the instance of SQL Server 2008, remove the data files and other components, reinstall the legacy instance of SQL Server 2000 or SQL Server 2005, and restore the original data. Having a backup or image of the initial system might enable you to shorten the time that is required to restore the original system on the server. One option is copying the legacy data files from a backup location to the appropriate disk volume, applying the ghost image to retrieve executables, and then applying any scripts or components to complete the rebuild of the original system.

In a side-by-side upgrade, the new instance of SQL Server 2008 resides alongside the legacy instance of SQL Server, either on the same server or on a different server. Therefore, the legacy instance continues to be available for a rollback scenario.

However, after the upgraded instance of SQL Server 2008 goes into production and starts capturing new data, there will come a point in time when enough new data has been captured that a rollback is no longer realistic. For an in-place upgrade, if you encounter problems after the system is in production, making adjustments or updates to the new application would be a better option than trying a rollback. For a side-by-side upgrade, you could use SSIS to transfer new data from the instance of SQL Server 2008 to the legacy instance of SQL Server 2000 or SQL Server 2005 to bring it up-to-date. However, this might be a difficult process, depending on the complexity of the data.

The complexity and expense of a rollback reinforces the importance of testing an upgrade process beforehand. See the "Upgrade Test Plan" section later in this chapter for information about how to test an upgrade.

#### Considerations for Choosing an Upgrade Strategy

The upgrade method that is available for your specific needs depends on many factors, including the components that you want to upgrade and the editions you want to use.

* **Components**. A certain upgrade strategy might not be possible because the component does not support it. For example, there is no in-place upgrade for SSIS from SQL Server 2000. For more information, see Chapter 13, "Integration Services. Also, we recommend that you transfer most SSAS components if the source is SQL Server 2000. For more information, see Chapter 11, "Analysis Services.”
* **Editions**. The in-place upgrade strategy does not support all paths between editions. For example, to upgrade a SQL Server 2000 or SQL Server 2005 Enterprise instance to SQL Server 2008 Standard, you must perform a side-by-side upgrade because Setup does not support an in-place upgrade path. See "Allowable Upgrade Paths" later in this chapter for more information.
* **Partial upgrading**. To transition only a few databases on a server to SQL Server 2008 and leave the rest on the legacy version, you must use a side-by-side upgrade.
* **Upgrading over time**. To transition databases gradually, several databases at a time, from a legacy instance to SQL Server 2008, you can only use a side-by-side upgrade.
* **Effect on applications.** If your organization requires minimal disturbance to the existing applications and users, choose an in-place upgrade if you can.
* **Availability**. Both an in-place upgrade and a side-by-side upgrade require that the databases be unavailable for some time. The downtime required depends primarily on the size of the data sets. At first, it might seem that an in-place upgrade would be faster than a side-by-side upgrade because the data is not transferred from one server to another. However, an in-place upgrade also requires time for the installation of SQL Server 2008. In a side-by-side upgrade, SQL Server 2008 is already installed on another instance. If the data transfer proceeds quickly and few changes are needed on the new instance, a side-by-side upgrade might be faster than an in-place upgrade.
* **Rollback**. For many database systems in production, it is impossible to justify a change without a rollback strategy, in case the results are unacceptable. The side-by-side upgrade strategy supports rollback at the time of acceptance testing because the legacy instance can still be made available. However, after users update the databases in the new instance, rollback might no longer be workable.

Some factors alone might be enough for you to decisively choose one strategy over another. Regardless of what strategy you select, do not forget testing and validation. Even if you select an in-place upgrade strategy, test the upgrade process and results on a separate server first. For more testing information, see "Upgrade Test Plan" later in this chapter.

### Backward Compatibility

When planning for an upgrade to SQL Server 2008, you have to understand what features are deprecated, discontinued, or changed in the new version. Being aware of these changes beforehand can help you prevent both performance problems and issues related to making the application available.

Generally, SQL Server 2008 is backward compatible with SQL Server 2000 and SQL Server 2005. However, you should examine some feature changes during the planning process. The most serious backward-compatibility issues that will affect planning are those that will block an in-place upgrade and prevent an installation of SQL Server 2008. If the SQL Server 2008 Setup program detects these issues during an in-place upgrade, it will exit the installation, leaving the legacy instance unchanged. The SQL Server 2008 Upgrade Advisor is the best tool for finding these kinds of blocking issues beforehand. Chances are good that you will encounter only a few issues, if any.

In the component- and feature-specific chapters in this document, you can review the relevant details for each of these categories. For more information, see [SQL Server Backward Compatibility](http://go.microsoft.com/fwlink/?LinkId=134114) in SQL Server 2008 Books Online.

**Note:** The most serious backward-compatibility issues that will affect your planning are those that will block an in-place upgrade and prevent the installation of SQL Server 2008. If the SQL Server 2008 Setup program detects these issues during an in-place upgrade, it will exit the installation, leaving the legacy instance unchanged. You must resolve the blocking issues to continue.

#### Deprecated Features

Features that are deprecated in SQL Server 2008 still operate the same as in the legacy versions. However, they will be removed in the next version of SQL Server. Access to these features does not necessarily have to be removed to complete an upgrade. However, you should eventually address them because they could cause problems with upgrades after SQL Server 2008. For more information, see [Deprecated SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133935) in SQL Server 2008 Books Online. Also see "System Monitor—SQL Server: Deprecated Features" in the Upgrade Tools section later in this chapter.

**Note:** An upgrade will not be blocked if you use deprecated features. However, it is advised that you decide how or when you want to deal with any of these to give yourself lots of time to resolve the issues before they are discontinued in some future SQL Server release.

#### Discontinued Features

In any component of SQL Server 2008, some features of earlier SQL Server versions may have been discontinued. These features functioned in earlier versions of SQL Server but were removed from SQL Server 2008. Although some references to these features might not block an in-place upgrade, you should remove those references anyway. If the reference is not removed, the application might not behave correctly. Use Upgrade Advisor to detect whether your application is using discontinued features. For more information about such features, see [Discontinued SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133937) in SQL Server 2008 Books Online.

#### Breaking Changes

Breaking changes to SQL Server 2008 are those that might require changes to the applications because the features in question now have a different behavior. If you do not use the feature, there is no effect on you. However, if you do use the feature, your application might be affected. The best tool for discovering this kind of issue is Upgrade Advisor, which analyzes a legacy system and reports on all potential breaking changes and how to address them. For more information about this kind of change, see [Breaking Changes to SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133938) in SQL Server 2008 Books Online.

#### Behavior Changes

Behavior changes might not visibly affect your database code or applications. However, you have to be aware of them because interpretation might be different. For example, the behavior of the SQL Server Native Client changes from SQL Server 2005 to SQL Server 2008. For more information, see [Behavior Changes to SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133944) in SQL Server 2008 Books Online.

### Upgrade Tools

We have talked about the value of the SQL Server 2008 Upgrade Advisor several times already in this chapter, and some other tools are also available to help automate the upgrade process to SQL Server 2008. Each tool has its own purpose and timing so that it is best to become familiar with all the tools and then use those most appropriate to each upgrade project.

#### SQL Server 2008 Upgrade Advisor

Perhaps the most important tool of the several tools typically used for upgrade planning is Upgrade Advisor. Upgrade Advisor smoothes the transition to SQL Server 2008 by predicting issues in your legacy instances of SQL Server 2000 and SQL Server 2005. It analyzes objects and code within legacy instances and produces reports detailing upgrade issues, if there are any, organized by SQL Server component. The resulting reports show detected issues and provide guidance about how to fix the issues or work around them. The reports are stored on disk, and you can review them by using Upgrade Advisor or export them to Microsoft Excel for further analysis.

In addition to analyzing data and database objects, Upgrade Advisor can analyze Transact-SQL scripts and SQL Server Profiler/SQL Trace traces. Upgrade Advisor examines SQL code for syntax that is no longer valid in SQL Server 2008. It generates a report listing the code in question, together with links to where you can find more information to help resolve the questionable code. For information about how to upgrade Transact-SQL queries, stored procedures, scripts, and application code, see Chapter 8, "Transact-SQL Queries."

Requirements for running Upgrade Advisor are as follows:

* Windows Server 2003 SP2, Windows Server 2008, Windows Vista SP1, or Windows XP SP3
* The Microsoft .NET Framework 2.0 (the same version of the .NET Framework included with SQL Server 2008 and Visual Studio 2005)
* Windows Installer 4.5
* SQL Server 2000 Decision Support Objects (DSO) if analyzing SSAS (you can use SQL Server 2000 Setup to install DSO)
* SQL Server 2000 client components if analyzing DTS (you can use SQL Server 2000 Setup to install the SQL Server 2000 client components)
* Pentium III-compatible processor or a later version, with a processor speed of at least 500 MHz
* 15 MB of available hard disk space

Whether you choose an in-place upgrade or a side-by-side upgrade, run Upgrade Advisor on your legacy systems. You can run Upgrade Advisor from a local or remote server, and you can execute it from the Command Prompt window by using a configuration file name as an input parameter.

**Note**: You can run the SQL Server 2008 Upgrade Advisor only against instances of SQL Server 2000 and SQL Server 2005. You cannot run it against instances of SQL Server 2008 or on SQL Server 7.0.

Upgrade Advisor is a separate download. The most recent downloadable version is available as part of the [Microsoft SQL Server 2008 Feature Pack](http://go.microsoft.com/fwlink/?LinkID=133088).

You can find more information about this valuable tool in the [Upgrade Advisor Guide in SQL Server 2008 Books Online](ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10sq_GetStart/html/d85b0833-ddeb-42e3-9397-97ea60d521b7.htm), and in [Using Upgrade Advisor to Prepare for Upgrades](http://go.microsoft.com/fwlink/?LinkID=132624).

#### SQL Server 2008 Upgrade Assistant

The SQL Server 2008 Upgrade Assistant is an external tool that lets you determine in a test environment how an application currently running on SQL Server 2000 or SQL Server 2005 will run on SQL Server 2008. This tool uses Upgrade Advisor, together with baseline and trace replay in a test environment, to help identify compatibility issues.

Requirements for using Upgrade Assistant are as follows:

* Windows Server 2003 R2, Windows Vista, or Windows XP SP2 or later versions
* SQL Server 2000 SP4 or later versions or SQL Server 2005 SP2 or later versions
* Microsoft .NET Framework 2.0 SP1 or later versions

Follow these steps for using Upgrade Assistant:

1. Prepare a test environment by setting up test servers that have the necessary software and appropriate configuration.
2. Upgrade Assistant then guides the backing up relevant application databases and capturing the application database workload.
3. It transfers the relevant files and databases to the test environment.
4. The tool then runs Upgrade Advisor on the test databases to analyze database schema, trace files, and script files for potential compatibility issues.
5. Upgrade Assistant next replays the baseline trace, by using the baseline workload to capture relevant execution data from the legacy application on the test server.
6. You can now reset the test databases and upgrade the test server by using SQL Server 2008 Setup.
7. Replay the test trace workload on the test server that runs SQL Server 2008 and capture the relevant SQL Server 2008 execution data.
8. Compare the results of the baseline test workload on SQL Server 2008 against the original SQL Server 2000 or SQL Server 2005 results (also on the test server). If there are any material differences between the two results, work to resolve these in advance of the production upgrade.

Upgrade Assistant automates most of these tasks. For more information and download instructions, see [SQL Server 2008 Upgrade Assistant](http://go.microsoft.com/fwlink/?LinkID=133150) on the Scalability Experts Web site.

#### Best Practices Analyzer for SQL Server 2000 and SQL Server 2005

Before you install SQL Server 2008, you should also run the SQL Server Best Practices Analyzer (BPA) against your current legacy instances of SQL Server. If bad or questionable practices exist, you could address them before the upgrade, moving the fixes through test and into production. Using best practices on the legacy SQL Server systems first will help ensure a smoother upgrade, but that is not always possible. You might have to change some practices during the upgrade process instead.

You can download the SQL Server 2000 version of BPA at the [Best Practices Analyzer Tool for Microsoft SQL Server 2000 1.0](http://go.microsoft.com/fwlink/?LinkID=133151) Web site.

You can download the SQL Server 2005 version of BPA at the [SQL Server 2005 Best Practices Analyzer (August 2008)](http://go.microsoft.com/fwlink/?LinkID=133152) Web site.

#### SQL Server 2008 Setup: System Configuration Checker

An in-place upgrade uses SQL Server 2008 Setup to directly upgrade a SQL Server 2000 or 2005 instance. SQL Server 2008's Setup program installs prerequisites such as the .NET Framework and PowerShell 1.0. It also scans the destination computer for minimum hardware and software requirements, in addition to a compatible SQL Server edition upgrade path for an in-place upgrade. To do this, the SQL Server 2008 Setup program contains a utility named the System Configuration Checker (SCC) that performs a scan of the computer in preparation for an installation. For more information, see [Check Parameters for the System Configuration Checker](http://go.microsoft.com/fwlink/?LinkID=133947) in SQL Server 2008 Books Online.

The Setup SCC looks for conditions that will prevent a successful SQL Server installation or upgrade. These checks occur before Setup starts the SQL Server 2008 Installation Wizard and report any issues that would block an installation together with advice about how to address the blocking issues. The Setup SCC uses rules from the following categories; for more information about any of these categories, see the related link from SQL Server 2008 Books Online:

* [Feature Installation Rules](http://go.microsoft.com/fwlink/?LinkId=134115)
* [Upgrade and Repair Rules Check](http://go.microsoft.com/fwlink/?LinkId=134116)
* [Edition Upgrade Rules](http://go.microsoft.com/fwlink/?LinkId=134117)
* [Uninstallation Rules](http://go.microsoft.com/fwlink/?LinkId=134118)

The common, relevant rules—across all four categories—for an in-place upgrade and a side-by-side upgrade, are here; failing any of these rules will result in a blocking issue that could prevent an in-place upgrade:

* The destination computer must be connected to the Internet while the .NET Framework security check validates a certificate.
* The SQL Server registry keys must be consistent.
* The CPU architecture of the installation program must match the CPU architecture of features intended for upgrading.
* If the computer is clustered, the cluster service must be online.
* Windows PowerShell must be installed. (Setup will do this automatically when it installs prerequisites.)
* SQL Server Setup must be supported on this operating system platform.
* SCC checks whether a pending computer restart is required.
* The existing performance counter registry hive must be consistent.
* SCC checks that neither SQL Server 7.0 nor SQL Server 7.0 OLAP Services is installed on the server. SQL Server 2008 is not supported on the same server that has SQL Server 7.0.

Here are some additional checks that SCC performs to determine whether the SQL Server editions in an in-place upgrade path are valid:

* Checks the system databases for features that are not supported in the SQL Server edition to which you are upgrading.
* Checks all user databases for features that are not supported by the SQL Server edition.
* Checks whether the SQL Server service can be restarted.
* Checks that the SQL Server service is not set to Disabled.
* Checks whether the selected instance of SQL Server meets the upgrade matrix requirements (see "Allowable Upgrade Paths" in this section).
* Checks whether SSAS is being upgraded to a valid edition.
* Checks whether the edition of the selected instance of SQL Server is supported in this scenario (see "Allowable Upgrade Paths" in this section in addition to Chapter 4, "High Availability," later in this guide).

For more information about SQL Server 2008 Setup, see "SQL Server 2008 Setup Requirements" later in this chapter.

#### SQL Server Profiler

SQL Server Profiler can record a running workload and then replay that same activity from a given SQL Server instance, making it a valuable tool for preparing an upgrade.

Profiler is useful for simulating an upgrade to determine performance and correct behavior. For example, you can use SQL Server 2008 Profiler to trace a SQL Server 2005 database under load and save the trace. You can then restore the SQL Server 2005 database to two instances on equivalent hardware: an instance of SQL Server 2005 and an instance of SQL Server 2008. Run the replay on each (but at different times if on the same server), and while you are running the replay, also run a Profiler trace on each run, capturing for errors and query durations. By comparing the results, you can determine whether the upgrade behaves correctly (without error) and performs well.

Using Profiler to test upgrade results is made much easier by using the SQL Server 2008 Upgrade Assistant. Upgrade Assistant helps automate the process and reports for comparing performance and behavior of an upgraded SQL Server. For more information, see "SQL Server 2008 Upgrade Assistant" earlier in this chapter.

**Note:** Whether using Profiler with the SQL Server 2008 Upgrade Assistant or by itself, make sure that the trace file contains a truly representative load against the server, one that contains **the full range of all queries** that the application will submit to the database. With a full range of queries and sufficient load, testing can add confidence to the upgrade plan.

For more information about how to use Profiler for replay, see [Replaying Traces](http://go.microsoft.com/fwlink/?LinkId=134119) in SQL Server 2008 Books Online.

#### System Monitor—SQL Server: Deprecated Features Object

SQL Server 2008 provides a new System Monitor (Perfmon) counter called SQLServer:Deprecated Features to monitor whether your application is submitting commands to the SQL Server 2008 Database Engine that are scheduled for removal from SQL Server in future releases. You should remove such deprecated commands from SQL Server 2008 applications after they are detected. You can use this counter to help plan modifications to your application code so that when you upgrade to the next version of SQL Server after SQL Server 2008, the upgrade process will go more smoothly. Select which kind of feature to monitor by using the Instance selection box for the counter. System Monitor records the total number of times the deprecated feature was encountered since SQL Server 2008 was last started. For more information about how to use this tool, see [SQL Server, Deprecated Features Object](http://go.microsoft.com/fwlink/?LinkId=134120) in SQL Server 2008 Books Online.

#### Analysis Services Migration Wizard

The Analysis Services Migration Wizard can help with a side-by-side upgrade of SSAS. For information about this tool, see Chapter 11, "Analysis Services."

#### DTS Package Migration Wizard

Installing SSIS 2008 also installs the DTS Package Migration Wizard, which helps the migration of DTS packages to SSIS.

Also, SQL Server 2008 provides support for running DTS packages. For more information, see [Support for Data Transformation Services (DTS) in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132618) in SQL Server 2008 Books Online.

For information about how to upgrade DTS to SSIS and support for DTS, see Chapter 13, "Integration Services."

#### Notification Services

You cannot perform an in-place upgrade of SQL Server Notification Services because it is not installed by SQL Server 2008 Setup. But you can use the Notification Services backward compatibility add-in; see Chapter 9, "Notification Services," for information about this add-in.

### SQL Server 2008 Setup

When planning an upgrade to SQL Server 2008, you first have to make sure that the target servers meet the necessary hardware and software requirements for SQL Server 2008 Setup to be completed.

**Note**: If your legacy instance of SQL Server 2000 or SQL Server 2005 is installed on Windows 2000 Server, you must upgrade to a new server by using a side-by-side upgrade; SQL Server 2008 is not supported on Windows 2000 Server.

#### Setup Requirements for an In-Place Upgrade

SQL Server 2008 Setup has important version-level requirements for upgrading instances of SQL Server 2000 and SQL Server 2005 in-place. The basic requirements are as follows:

* SQL Server 2000: SP4 is required.
* SQL Server 2005:
  + On Windows 2008 Server, SQL Server 2005 SP2 or later versions is required. (SQL Server 2005 SP1 is not a supported configuration for an in-place upgrade on Windows Server 2008.)
  + Otherwise, SQL Server 2005 RTM or a later versions is required.

When you upgrade a SQL Server 2000 installation that is still at SP3a or an instance of SQL Server 2005 on Windows Server 2008 that is at SP1, two courses of action can be taken:

* Upgrade the legacy SQL Server instance to the minimum service-pack level during a scheduled maintenance window before the upgrade, and then execute an in-place upgrade. This carries the cost of an additional outage to install the service pack before the upgrade, in addition to the risk of introducing a new configuration to one that might have been in place for years. Do not consider this path without extensive application testing to make sure that the service-pack upgrade does not introduce behavior that could have adverse effects.
* Perform a side-by-side upgrade on the same server or a new server instead of an in-place upgrade. If the configuration of the current server cannot be changed or the server is too old, you might have to requisition a new server.

For an in-place upgrade, the target server and the legacy instance of SQL Server 2000 or SQL Server 2005 must satisfy some additional requirements for SQL Server 2008:

* Cross-version instances of SQL Server 2008 are not supported. Version numbers of the Database Engine, Analysis Services, and Reporting Services components must be the same throughout an instance of SQL Server 2008. Therefore, you must upgrade all these components together during an in-place upgrade.
* Make sure that sufficient disk space is available for SQL Server 2008 Setup. Disk space requirements vary based on the components selected to upgrade. For disk space amounts, see the "Hard Disk Space Requirements (32-Bit and 64-Bit)" section in the [Hardware and Software Requirements for Installing SQL Server 2000](http://go.microsoft.com/fwlink/?LinkID=133946) topic in SQL Server 2008 Books Online.
* Before upgrading SQL Server, enable Windows Authentication for SQL Server Agent and verify the default configuration (that the SQL Server Agent service account is a member of the SQL Server sysadmin group).
* Before upgrading from one edition of SQL Server 2008 to another, verify that the functionality currently being used is supported in the edition to which you are upgrading. For more information, see the section for specific components [Planning a SQL Server Installation](http://go.microsoft.com/fwlink/?LinkId=134121) in SQL Server 2008 Books Online.
* Cross-platform upgrades (from x86 SQL Server 2000 or SQL Server 2005 to x64 SQL Server 2008 and vice versa) are not supported.
* Make sure that you are running a supported version of the Windows operating system.
* The in-place upgrade will be blocked if:
  + The server has a pending restart
  + The Windows Installer service is not running
  + Windows System Monitor Performance Counters are corrupted
* To upgrade an instance of SQL Server to a SQL Server failover cluster, the instance being upgraded must be a failover cluster (or it must be upgraded to a failover cluster first). In other words, to upgrade a stand-alone instance of SQL Server to a SQL Server failover cluster, install a new SQL Server failover cluster, and then move user databases from the stand-alone instance by using the Copy Database Wizard. (See Chapter 4, "High Availability," for more information.)

**Note:** When the in-place upgrade process is running, avoid making any changes to the legacy SQL Server 2000 or SQL Server 2005 system.

For more information, see the "Upgrade Notes" section in the [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) topic in SQL Server 2008 Books Online.

#### SQL Server Prerequisites Installed by Setup

* The SQL Server 2008 Installation Wizard installs the following prerequisites (if they are not already present on the computer):
* .NET Framework 3.5 SP1
* SQL Server Native Client
* SQL Server support files

**Note**: SQL Server 2008 requires Visual Studio 2008 SP1 components in order to install the Integration Services, Management Tools, and Business Intelligence Development Studio components. If you are running a version of Visual Studio 2008 earlier than SP1, apply Visual Studio 2008 SP1 first. For more information, see [Visual Studio 2008 SP1 may be required for SQL Server 2008 installations](http://go.microsoft.com/fwlink/?LinkId=134122) in the Microsoft Knowledge Base.

To reduce the time that is required for the upgrade process, install the .NET Framework 3.5 SP1 components (you must have SP1 or a later version) and SQL Server 2008 Native Client beforehand on the server that will be upgraded. Then, include the same components on the baseline image of the test server. If the production system cannot be disturbed in any way before the scheduled downtime for the upgrade process, the SQL Server 2008 Setup program will automatically install the prerequisites as part of the upgrade process. However, this increases the time that is required for the upgrade.

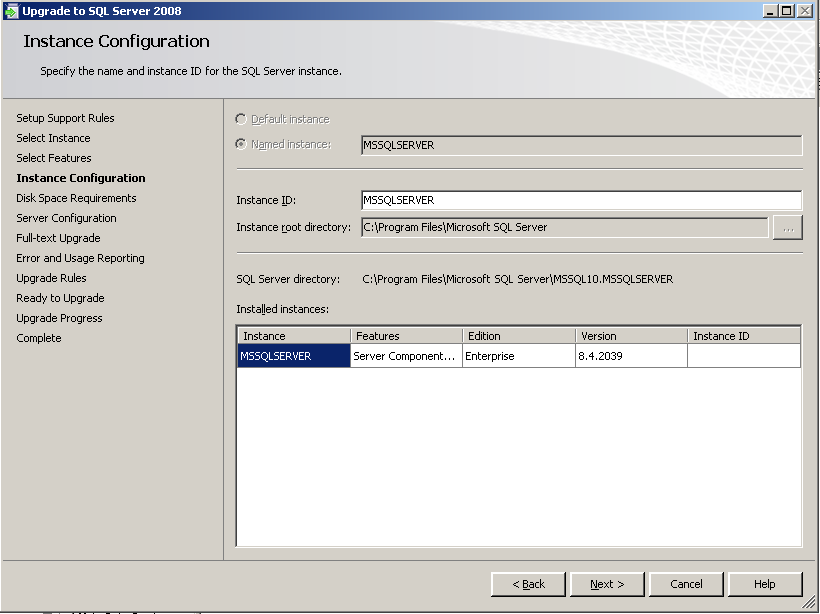
If either the Client Software or Database Services components are installed, Setup also installs:

* Windows PowerShell 1.0 (if it is not already present)
* SQL Server PowerShell snap-ins
* The sqlps.exe command prompt utility for running SQL Server 2008 PowerShell snap-ins

#### Instance ID and Paths

Whether you are performing an in-place upgrade or a side-by-side upgrade, SQL Server Setup will ask for an Instance ID. The Instance ID is a new concept in SQL Server 2008. It is a unique identifier specified during the upgrade (or install) to identify that specific SQL Server 2008 installation. The Instance ID behaves similarly to an instance of SQL Server 2000 or SQL Server 2005 name, but it has some additional features. Default instances of SQL Server always have a default value of MSSQLSERVER. In addition, the Instance ID is recorded in SQL Server 2008's program files, which are located by default at X:\Program Files\Microsoft SQL Server\MSSQL10.*InstanceID*, where X is your system drive, such as drive C.

Figure 1-4 shows an example of the Setup program requesting an Instance ID.



**Figure 1-4: Instance Configuration screen of an in-place upgrade, which shows the Instance ID**

Choose an Instance ID that makes sense; do not necessarily accept default values. This is especially true for failover clustering implementations, where instances are not "local" and will have a presence on each node. (For more information about clustering, see Chapter 4, "High Availability.")

#### Minimum Hardware and Software Requirements for SQL Server 2008

In this section, we describe the minimum hardware and software requirements for running SQL Server 2008. For detailed information about the minimum hardware and software requirements for all editions of SQL Server 2008, see [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132172) in SQL Server 2008 Books Online. The following subsections are pasted for the reader’s convenience from information found in this link. The link might contain more recent information than what is in this topic.

The following minimum hardware and software requirements apply to all SQL Server 2008 editions:

* All editions of SQL Server 2008 require .NET Framework 3.5. SQL Server 2008 Setup will install the .NET Framework 3.5, the SQL Server Native Client, and the required SQL Server 2008 Setup support files.
* There is one exception: For SQL Server 2008 running on Windows Server 2003 (64-bit) on an IA-64 system, .NET Framework 2.0 SP1 is required.
* SQL Server does not install the .NET Framework 3.5 SDK. However, the SDK contains tools that are useful when you use the .NET Framework for SQL Server development. You can download the .NET Framework SDK from the [.NET Framework Downloads Web page](http://go.microsoft.com/fwlink/?LinkId=134123).
* For SQL Server Express and SQL Server Express with Advanced Services, you must manually install the following components before you run SQL Server 2008 Setup:
  + SQL Server Express: .NET Framework 2.0 SP2 and Windows Installer 4.5 (on Windows Vista, use .NET Framework 3.5 SP1)
  + SQL Server Express with Advanced Services: .NET Framework 3.5 SP1, Windows Installer 4.5, and Windows PowerShell 1.0

#### Processor, Memory, and Operating System Memory Requirements

The processor, memory, and operating system minimum requirements vary based on the SQL Server 2008 edition you select. SQL Server 2008 Books Online on MSDN contains an updated cross-reference for the following combinations. The following links provide details for the server-based editions of SQL Server 2008:

* [SQL Server 2008 Enterprise (64-bit) IA-64](http://go.microsoft.com/fwlink/?LinkId=134126)
* [SQL Server 2008 Enterprise (64-bit) x64](http://go.microsoft.com/fwlink/?LinkId=134127)
* [SQL Server 2008 Standard (64-bit) x64](http://go.microsoft.com/fwlink/?LinkId=134128)
* [SQL Server 2008 Enterprise (32-bit)](http://go.microsoft.com/fwlink/?LinkId=134129)
* [SQL Server 2008 Standard (32-bit)](http://go.microsoft.com/fwlink/?LinkId=134130)
* [SQL Server 2008 Developer (64-bit) IA-64](http://go.microsoft.com/fwlink/?LinkId=134131)
* [SQL Server 2008 Developer (64-bit) x64](http://go.microsoft.com/fwlink/?LinkId=134132)
* [SQL Server 2008 Workgroup (64-bit) x64](http://go.microsoft.com/fwlink/?LinkId=134133)
* [SQL Server 2008 Web (64-bit) x64](http://go.microsoft.com/fwlink/?LinkId=134134)
* [SQL Server 2008 Express (64-bit) x64](http://go.microsoft.com/fwlink/?LinkId=134135)
* [SQL Server 2008 Express Advanced (64-bit) x64](http://go.microsoft.com/fwlink/?LinkId=134136)
* [SQL Server 2008 Developer (32-bit)](http://go.microsoft.com/fwlink/?LinkId=134137)
* [SQL Server 2008 Workgroup (32-bit)](http://go.microsoft.com/fwlink/?LinkId=134138)
* [SQL Server 2008 Web (32-bit)](http://go.microsoft.com/fwlink/?LinkId=134139)
* [SQL Server 2008 Express and Express with Advanced Services (32-bit)](http://go.microsoft.com/fwlink/?LinkId=134140)

Note the following restrictions or constraints:

* SQL Server 2008 is not supported on Windows Server 2008 Server Core installations.
* The Setup SCC will block Setup if the requirement for processor type and minimum operating system, in addition to other conditions, are not met. For more information, see [Check Parameters for the System Configuration Checker](http://go.microsoft.com/fwlink/?LinkID=133947) in SQL Server 2008 Books Online.
* An in-place direct upgrade of SQL Server 2000 (64-bit) will not install SQL Server 2008 Management Tools. To install the SQL Server 2008 Management Tools, you must run Setup again after the upgrade is completed. For more information, see Footnote 5 in [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) in SQL Server 2008 Books Online.

#### Hard Disk Space Requirements (32-bit and 64-bit)

Hard disk is required to store the user data within your user databases. Consider the user data in your disk space calculations, and if you intend to keep the target and new instances online in parallel for any length of time, remember that you will need double the disk space to store the same data two times.

Additionally, there are disk space needs for the SQL Server system files and objects, which must be considered in your calculations. Installing SQL Server 2008 requires that the Windows Installer create temporary files on the system drive. Before you run Setup to install or upgrade SQL Server, verify that you have at least 2 GB of available disk space on the system drive for these files. (This requirement applies even if you install SQL Server components to a non-system drive.) Table 1-4, from SQL Server 2008 Books Online, shows the minimum disk space requirements for the major SQL Server 2008 components. For complete details, see [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132172).

Table 1-4: SQL Server 2008 Disk Space Minimum Requirements

|  |  |
| --- | --- |
| **Feature** | **Disk Space Minimum Requirement** |
| Database Engine and data files, Replication, and Full-Text Search | 280 MB |
| Analysis Services and data files | 90 MB |
| Reporting Services and Report Manager | 120 MB |
| Integration Services | 120 MB |
| Client Components | 850 MB |
| SQL Server Books Online and SQL Server Compact Books Online | 240 MB |

#### Extended System Support (WOW64)

SQL Server 2008 on Windows x64 can use extended systems, also known as Windows on Windows 64 (WOW64). WOW64 is a 64-bit Windows edition feature that enables 32-bit applications such as SQL Server 2008 management tools to execute natively in 32-bit mode. Such applications function in 32-bit mode even though the underlying operating system is running on the 64-bit operating system. As a result, you can upgrade to a 32-bit SQL Server 2008 edition on Windows x86, a 64-bit SQL Server 2008 edition on Windows x64, or a 32-bit SQL Server 2008 edition on Windows x64 using WOW64. For more information, see the "Extended System Support" section in the [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132172) topic in SQL Server 2008 Books Online.

**Note:** A cross-platform, in-place upgrade from a 32-bit instance of SQL Server 2000 or SQL Server 2005 (x86) to a SQL Server 2008 64-bit instance (x64), or vice versa, is not supported. Similarly, an upgrade in the reverse direction (an upgrade of a SQL Server 2005 64-bit instance to a SQL Server 2008 32-bit instance) is not supported. To change from a 32-bit to 64-bit instance, or vice versa, you must use a side-by-side upgrade. To upgrade from SQL Server 2000 or SQL Server 2005 on a 32-bit system to SQL Server 2008 on a 64-bit system, install SQL Server 2008 separately on the 64-bit version of Windows and then transfer the database objects and settings of the legacy instance to the new instance. For more information, see the "Upgrade Notes" section in [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) in SQL Server 2008 Books Online.

#### Installing SQL Server on a Domain Controller

* Although you can install SQL Server 2008 on a domain controller, we do not recommend it for the following reasons:
* On Windows Server 2003, SQL Server services can run under a domain account or a local system account. However, on a domain controller, SQL Server services cannot run under a local service account or a network service account.
* After SQL Server 2008 is installed on a domain member server, the server's network role cannot be changed from a domain member to a domain controller. SQL Server must be uninstalled before the host computer is changed to a domain controller.
* Similarly, after SQL Server 2008 is installed on a domain controller, it cannot be changed to a domain member unless SQL Server is first uninstalled.
* SQL Server 2008 failover cluster instances are not supported where the cluster nodes are domain controllers.
* SQL Server 2008 is not supported on a read-only domain controller.

For more information, see [Installing SQL Server on a Domain Controller](http://go.microsoft.com/fwlink/?LinkId=134141) in SQL Server 2008 Books Online.

### Allowable Upgrade Paths

We have mentioned that certain versions and components of SQL Server can be upgraded to SQL Server 2008. Now let's be more specific about what versions, components, and upgrade paths are available.

#### Minimum Versions of SQL Server 2000 and SQL Server 2005

For an in-place upgrade, SQL Server 2008's Setup program requires you to have certain versions of either SQL Server 2000 or SQL Server 2005. (For more information, see "Setup: System Configuration Checker" earlier in this chapter.) Specifically, the in-place method that is provided by SQL Server 2008 Setup can be used to directly upgrade the following versions:

* SQL Server 2000 SP4 or later versions
* SQL Server 2005 RTM or later versions (Windows Server 2003)
* SQL Server 2005 SP2 or later versions (Windows Server 2008)

If you have to upgrade earlier versions of SQL Server 2000 or SQL Server 2005, use the side-by-side method.

#### Upgrading from SQL Server 7.0 to SQL Server 2008

SQL Server 2008 cannot directly upgrade a SQL Server 7.0 instance in-place. For upgrading from SQL Server 7.0 to SQL Server 2008, available options include the following:

* Upgrade the instance of SQL Server 7.0 in-place to SQL Server 2000 or SQL Server 2005, and then upgrade the resulting instance in-place to SQL Server 2008. This in-place upgrade path requires two steps.
* Upgrade the instance of SQL Server 7.0 to a new instance of SQL Server 2008 by using a side-by-side upgrade method.

Whether you use a two-step in-place upgrade or a side-by-side upgrade, you should use the SQL Server 2005 Upgrade Advisor to inspect the instance of SQL Server 7.0. If any blocking issues (discontinued features or breaking changes) are found, you should remove them. For more information, see the [SQL Server 2005 Upgrade Technical Reference Guide](http://go.microsoft.com/fwlink/?LinkID=71736) white paper.

#### Upgrading SQL Server Components

SQL Server is a complex product, featuring many components that are fairly independent. Table 1-5 shows the SQL Server 2000 components that you can upgrade and what paths are available; Table 1-6 shows the same information for SQL Server 2005.

Table 1-5: Components and Upgrade Strategies: SQL Server 2000 to SQL Server 2008

|  |  |  |
| --- | --- | --- |
| **SQL Server 2008 Component** | **In-Place Upgrade**  **(SQL Server 2000)** | **Side-by-Side Upgrade**  **(SQL Server 2000)** |
| Database Engine | SQL Server Setup  (upgrades all databases and preserves server configurations when it is possible) | One or two servers  (Use backup/restore, detach/attach, or Copy Database Wizard) |
| Analysis Services | SQL Server Setup | Use the Analysis Services Migration Wizard  (see Chapter 11, "Analysis Services") |
| Integration Services | None | Use DTS Migration Wizard  (see Chapter 13, "Integration Services") |
| Reporting Services | Use SQL Server Setup (for a default configuration only) | Use manual data transfer steps  (see Chapter 14, "Reporting Services") |
| Notification Services | Not Available | Use the provided Notification Services runtime add-in  (see Chapter 9, "Notification Services") |

Table 1-6: Components and Upgrade Strategies: SQL Server 2005 to 2008

| **SQL Server 2008 Component** | **In-Place Upgrade**  **(SQL Server 2005)** | **Side-by-Side Upgrade**  **(SQL Server 2005)** |
| --- | --- | --- |
| Database Engine | SQL Server Setup  (upgrades all databases and preserves server configurations when it is possible) | One or two servers  (see Chapter 3, "Relational Databases") |
| SQL Server High Availability Solutions | SQL Server Setup, with special considerations  (see Chapter 4, "High Availability") | See Chapter 4, "High Availability" |
| Analysis Services | SQL Server Setup | Manual data transfer  (see Chapter 11, "Analysis Services") |
| Integration Services | SQL Server Setup | Manual data transfer  (see Chapter 13, "Integration Services") |
| Reporting Services | SQL Server Setup | Manual transfer of reports  (see Chapter 14, "Reporting Services") |
| Notification Services | Not Available | Use the provided Notification Services runtime add-in  (see Chapter 9, "Notification Services") |

#### Allowable In-Place Upgrade Paths by Edition

In a side-by-side upgrade, if data transfer is handled manually, there are no restrictions on the paths taken from one edition to another: In principle, any edition can be upgraded to another edition, if the database data is compatible. However, there are some restrictions based on Enterprise features. For example, table partitioning is supported only by the Enterprise edition of SQL Server 2008 and SQL Server 2005. So, a SQL Server 2005 Enterprise database that contains partitioned tables cannot be restored to SQL Server 2008 Standard.

However, there are restrictions on the upgrade path when you upgrade directly. When you use SQL Server 2008's Setup program for an in-place upgrade, the program will verify that the upgrade path for SQL Server editions is enabled. For more information, see [Edition Upgrade Rules](http://go.microsoft.com/fwlink/?LinkID=134117) in SQL Server 2008 Books Online.

**Note**: The different components of SQL Server 2008 must be kept at the same edition level when they are on the same instance. For example, if you are running the SQL Server 2008 Enterprise Database Engine on a particular instance, you must also install the Enterprise edition of Analysis Services as part of that instance.

For a detailed view of what in-place upgrade paths are allowed, see the table "Version and Edition Upgrade Paths" in the Appendix to this guide, and [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) in SQL Server 2008 Books Online.

#### Upgrading Database Collation

As databases continue to expand and support a growing global market, users must be able to work with character data in meaningful ways. Collations are a powerful way for users to sort and compare strings according to their own cultural conventions. Therefore, collations are an important factor when you create a database and operating on the data. In addition, when you use the character data types such as char and varchar, the collation dictates the code page and therefore determines which characters can be represented for that data type.

**New Collations in SQL Server 2008.** SQL Server 2008 has introduced new collations that are in full alignment with the collations provided by Windows Server 2008. These new collations are denoted by \*\_100 version and give users the most up-to-date and linguistically accurate cultural sorting conventions.

The new collations include support for new East Asian government standards, linguistically correct for surrogates, support for the Chinese minority scripts, Unicode 5.0 case table, and weights to previously non-weighted characters that would have compared equal. Although new versions are being added to the existing Windows collations from SQL Server 2000 and SQL Server 2005 to reflect these changes, all current collations will be maintained in SQL Server 2005 for backward compatibility. Be aware that no changes were made to the SQL\_\* collations.

**Database collation upgrade considerations.** Here are some considerations that might affect an upgrade of a database collation:

* Compatibility with existing instances of SQL Server or application code that depends on consistent SQL Server collations sorting behavior. Changing collations might require rebuilding indexes, so consider the decision to change collations carefully based on business and technical requirements.
  + The current need or future requirement to store character data from different languages. If you have both Unicode and non-Unicode data in your database, consider Using the Windows-based collations because they apply Unicode-based sorting rules to both Unicode and non-Unicode data. This provides the following benefits:
  + Enable consistency across data types in SQL Server because non-Unicode data is converted to Unicode to perform comparison operations
  + Enable developing applications that use sort semantics consistent with SQL Server, Windows, and other Microsoft applications
* No one else is using the database during the change in collation.
* You have no schema-bound objects that depend on the database collation, including the following:
  + User-defined functions and views created by using SCHEMABINDING
  + Computed columns
  + CHECK constraints
  + Table-valued functions that return tables with character columns with collations inherited from the default database collation
  + A collation change that can potentially cause duplicate system names, such as switching from case-sensitive to case-insensitive or changing from a collation with unique character weights, which will raise an error and fail. Objects that can potentially cause duplications are as follows:
    - Object names
      * Procedure, table, trigger, or view
    - Schema names
      * Group, role, or user
    - Scalar-type names
      * System and user-defined types
    - Full-text catalog names
    - Column or parameter names in an object
    - Index names in a table
* Changing the database collation does not alter the collations in pre-existing tables. New tables that are created after you change to the new collation will use the new collation.
* Changing between collations with different code pages is not recommended. This might result in data loss when you change the collation of a column to a collation that is not recognized in the target collation’s code page. For example, if you change from a Japanese to an English collation, be aware that most Japanese characters are not recognized by the Latin1\_General\_CI\_AS 1252 code page. If you have to change collations to different languages, try the following:
  + Store only data that belong to the code page for the character column.
  + Change from a non-Unicode data type (char, varchar) to a Unicode data type (nchar, nvarchar).

For more information about how to change collations, see the [Changing Collation Best Practices](http://go.microsoft.com/fwlink/?LinkID=113891) and [Best Practices for Migrating Non-Unicode Data Types to Unicode](http://go.microsoft.com/fwlink/?LinkID=113890) white papers.

#### In-Place Upgrade and Localization

The English version of SQL Server is supported on all localized versions of supported operating systems. In addition, localized versions of SQL Server are supported on localized operating systems that are the same language as the localized SQL Server version.

However, there are some important in-place upgrade restrictions when you change from one language to another:

* You can upgrade localized versions of SQL Server 2000 and SQL Server 2005 to localized versions of SQL Server 2008 of the same language.
* You cannot upgrade non-English localized versions of SQL Server to the English-language version of SQL Server 2008.
* You cannot upgrade the English version of SQL Server to any non-English-language localized version of SQL Server 2008. Note: This is a change from SQL Server 2005.
* You cannot upgrade localized versions of SQL Server to localized SQL Server 2008 versions if to a different localized language. The in-place upgrade must be to the same localized language.

Localized versions of SQL Server are also supported on English-language versions of supported operating systems by using the Windows Multilingual User Interface Pack (MUI) settings. However, you must verify certain operating system settings before you install a localized version of SQL Server on a server that is running an English-language operating system with a non-English MUI setting. Verify that the following operating system settings match the language of SQL Server to be installed:

* The operating system user interface setting
* The operating system user locale setting
* The system locale setting

If these operating system settings do not match the language of the localized SQL Server, set them correctly before you install SQL Server 2008. For more information, see [Cross-Language Support](http://go.microsoft.com/fwlink/?LinkId=134142) in SQL Server 2008 Books Online.

For more information about localization issues, see [Collation and Unicode Support](http://go.microsoft.com/fwlink/?LinkID=131366) in SQL Server 2008 Books Online.

### Application and Connection Requirements

All applications, whether purchased or developed in-house, have to be tested and verified to work against the new SQL Server 2008 back end. Do not assume all applications will "just work" after an upgrade. You might also have to perform specific application-related tasks before, during, or after the SQL Server upgrade.

**Note:** If you have a packaged application, before even planning an upgrade, check with the software vendor to make sure that the software version that you are using is certified to work with SQL Server 2008 and supported on that platform. It is a good idea that you not upgrade until you receive assurance of continued vendor support for the SQL Server 2008 version.

SQL Server 2008 is closely coordinated with the .NET Framework. To take full advantage of many new features in SQL Server 2008, both the updated SQL Server Native Client (Version 10.0) on the server and the .NET Framework 3.5 SP1 on the client are required. SQL Server 2008 will automatically install the SQL Server Native Client 10.0 on all SQL Server 2008 servers, but software developers must upgrade their applications to use .NET Framework 3.5 to take full advantage of the new driver and many of SQL Server 2008’s new features. However, it is not required that all applications use .NET Framework 3.5.

#### SQL Server Native Client 10.0

SQL Server 2008 includes an updated version of the SQL Server Native Client, version 10.0. The behavior changes that software developers should be aware of are documented in [Updating an Application to SQL Server 2008 Native Client from SQL Server 2005 Native Client](http://go.microsoft.com/fwlink/?LinkId=134143) in SQL Server 2008 Books Online.

Applications developed by using OLE DB or ODBC will still work when they connect to SQL Server 2008. Although the overall recommended method for connecting to SQL Server 2008 for non-.NET Framework applications is through the new SQL Server Native Client, you can use the current OLE DB and ODBC drivers until you upgrade the clients. However, new features in SQL Server 2005 and SQL Server 2008, such as the new spatial and XML data types, are not fully supported in the OLE DB or ODBC drivers. For more information, see [When to use SQL Server 2008 Native Client](http://go.microsoft.com/fwlink/?LinkId=134144) in SQL Server 2008 Books Online.

Although DB-Library client tools (such as isql.exe) are no longer supported in SQL Server 2008, you can still make connections to SQL Server 2008 by using the DB-Library API. However, clients will be unable to take full advantage of all the new SQL Server 2008 features unless they use the SQL Server Native Client API. Only the SQL Server Native Client API supports all the new SQL Server 2008 features (through the built-in new OLE DB and ODBC interfaces.) Embedded SQL (ESQL) applications are still supported. For more information, see the note about DB-Library in [Client Network Configuration](http://go.microsoft.com/fwlink/?LinkId=134145) in SQL Server 2008 Books Online.

#### Upgrading Applications That Use the .NET Framework

Clients that use .NET Framework 1.x can still work unchanged when they connect to an instance of SQL Server 2008. However, these applications cannot take advantage of new SQL Server 2008 (or SQL Server 2005) capabilities unless they are upgraded to ADO.NET 3.5 and the new SQL Server .NET Managed Data Provider. For example, some new features such as Database Mirroring Automatic Failover on LINQ or Multiple Active Result Sets (MARS) require the new drivers.

There might also be issues related to upgrading client applications. If client applications have to take full advantage of new SQL Server 2008 data types or features such as the XML data type, the client applications have to be upgraded to .NET Framework 3.5 SP1 or a later version.

There are very few issues when you move .NET 1.1 applications from SQL Server 2000 or SQL Server 2005 to SQL Server 2008. However, the clients cannot take full advantage of new SQL Server 2008 features. Some features such as snapshot isolation are available only through Transact-SQL commands. Also, SQL Server 2005 Notification Services is not available for .NET 1.1 clients and is not part of SQL Server 2008.

If you upgrade clients from .NET 1.1, 2.0, or 3.0 applications to .NET 3.5 SP1 or later versions, by using SQL Server 2005 or SQL Server 2008, the client applications will be able to take full advantage of new SQL Server 2005 and SQL Server 2008 capabilities.

For information about SQL Server 2008 Native Client, see the following links in SQL Server 2008 Books Online:

* [SQL Server 2008 Native Client Programming](http://go.microsoft.com/fwlink/?LinkId=134146)
* [Updating an Application to SQL Server 2008 Native Client from SQL Server 2005 Native Client](http://go.microsoft.com/fwlink/?LinkID=134143)
* [When to Use SQL Server Native Client](http://go.microsoft.com/fwlink/?LinkID=134144)
* [SQL Server Native Client Features](http://go.microsoft.com/fwlink/?LinkId=134147)
* [Building Applications with SQL Server Native Client](http://go.microsoft.com/fwlink/?LinkId=134148)
* [System Requirements for SQL Server Native Client](http://go.microsoft.com/fwlink/?LinkId=134149)
* [SQL Server Native Client (OLE DB)](http://go.microsoft.com/fwlink/?LinkId=134150)
* [SQL Server Native Client (ODBC)](http://go.microsoft.com/fwlink/?LinkId=134151)
* [Finding More SQL Server Native Client Information](http://go.microsoft.com/fwlink/?LinkId=134152)

For developer information about SQL Server 2008 data access, see the [Microsoft Data Platform Development Web site](http://go.microsoft.com/fwlink/?LinkID=62396). This is the primary data access site for Microsoft technologies.

### Plan for Backups

A backup of each database in an instance, including system databases, is the keystone of an upgrade plan. Even when you upgrade to a new server by using a side-by-side method, you should still take backups. Perform backups at the following points in the upgrade process:

* Make a backup of the user databases and data after all users are out of the system and before the upgrade process has begun. Do nothing until this is completed. Back up all system databases at this point. These backups form the databases of record, marking the final versions of your old environment. If you can do this, copy these backups to a different server and make sure that they can be easily accessed even in a complete server-down situation. Make sure that the media is intact so that you can restore the backups if it is necessary.
* When the upgrade is complete, but before you do any configurations or changes, make backups under SQL Server 2008. This lets you easily roll back to a point where the SQL Server 2000 or SQL Server 2005 upgrade completed successfully but where an error was introduced after that point.
* After you make any changes to the SQL Server 2008 databases and configuration as well opening SQL Server for acceptance testing, make backups again. If the testers consider the upgrade a success, these backups will be the initial backups for your new environment. If testing finds errors but they are not serious enough to cause a rollback, you can restore from these initial backups to revert the database to its original state, ready for a second round of acceptance testing. Then apply required changes and repeat the backup process. When testing is complete, still backup all databases before rolling out to production. These backups then capture the final state of the database before they are put into production.

**Important: Make sure that either the Windows Server installation media with the appropriate keys or good backups of the Windows Server installation image are available for a potential reinstall.** You can rebuild SQL Server only if Windows is stable and in a state to have applications installed on it.

It might seem that these backups are too many and will consume lots of disk space, but you can delete most of them after the upgrade to SQL Server 2008 is completed. It is a good practice to perform all of this, just in case an unexpected issue requires a restore at any point in time.

### Upgrading Both Windows and SQL Server

Because SQL Server 2008 was released on the heels of Windows Server 2008, you might be tempted to combine a SQL Server 2008 upgrade with a Windows Server 2008 upgrade to reduce the downtime of two major upgrades into one downtime event, instead of having two separate upgrade outages. However, upgrading both at the same time can introduce a significant new variable into the SQL Server 2008 upgrade plan, increase the risk of failure, and increase the cost of rolling back. Therefore, a decision to combine these two should be made carefully.

**Note:** If you decide to change the operating system at the same time as the SQL Server upgrade, we recommend that you design and execute a test of the upgrade and operating system changes in a test or staging environment before you try it in production.

Some considerations when you upgrade both Windows and SQL Server:

* If the legacy instances of SQL Server 2000 or SQL Server 2005 are on Windows Server 2000, an upgrade of Windows is required because SQL Server 2008 is not supported on Windows Server 2000. In that case, consider a side-by-side upgrade to a separate server, especially if the legacy server is older and does not meet the minimum requirements for SQL Server 2008.
* If the legacy instances of SQL Server 2000 or SQL Server 2005 are on Windows Server 2003, be aware that SQL Server 2000 is *not* supported under Windows Server 2008. Therefore, if you are currently using SQL Server 2000 and plan to upgrade the same server to Windows Server 2008, upgrade your SQL Server 2000 instance to SQL Server 2008 before upgrading to Windows Server 2008. This process will translate into two outages, but if you upgrade to Windows Server 2008 before upgrading to SQL Server 2008, the resulting SQL Server 2000 instances will not be supported.
* If a legacy instance of SQL Server 2000 is running on Windows Server 2000, use a multistep process. For example, upgrade Windows Server 2000 to Windows Server 2003, and then SQL Server 2000 to SQL Server 2008, and then Windows Server 2003 to Windows Server 2008.
* If you are upgrading to Windows Server 2008 and the server is currently running SQL Server 2005, make sure that you apply SQL Server 2005 SP2 or later versions before the Windows operating system upgrade; otherwise, you might encounter problems.
* If you will reuse the existing server and upgrade Windows Server, depending on which version of Windows that you start with and which is the final destination, you could perform an in-place upgrade. This approach might also require installing a fresh version of Windows Server. If you perform a fresh install of Windows Server, make sure that all SQL Server databases are backed up, that all settings are known, and that all users are scripted out.
* Windows Server 2008 has a new installation option called Core. Core is basically a locked-down, minimal version of Windows Server that does not have an interface other than a command line. SQL Server does not support Core because Core does not support the SQL Server-required installation of the .NET Framework.
* There are two kinds of virtualization with a Windows Server 2008 installation: with Hyper-V and without Hyper-V. SQL Server is supported on both types. However, as a rule, a server that is used for SQL Server production data should be dedicated only to SQL Server to reduce the risk of another process bringing SQL Server down. Unless a server must run virtual machines that will host SQL Server, install Windows Server 2008 without Hyper-V. (Going without Hyper-V also means that Windows administrators have one less feature to worry about when patching for security.) If Windows Server 2008 is installed with Hyper-V, apply the final Hyper-V Release to Manufacturing (RTM) patch. For more information, see [Hyper-V Update for Windows Server 2008 x64 Edition (KB950050)](http://go.microsoft.com/fwlink/?LinkId=134154) or [Hyper-V Update for Windows Server 2008 (KB950050)](http://go.microsoft.com/fwlink/?LinkId=134153) in the Microsoft Knowledge Base.
* You cannot use an old Windows NT-style domain for a Windows Server 2008 server. You must be using Active Directory. If your organization is using older-style domains, you cannot deploy Windows Server 2008 until your Active Directory infrastructure is upgraded.
* By default, Windows Server 2008 is more secure than Windows Server 2000 and Windows Server 2005. Many of its features are not configured so that there will be additional work involved in the upgrade.
* Windows Server 2008 supports only SAS, fiber channel, and iSCSI. If you are using older parallel SCSI, you have to upgrade your disk solutions to support Windows Server 2008.

For more Windows-specific information about how to deploy and upgrading to Windows Server 2008, see [Upgrading to Windows Server 2008](http://go.microsoft.com/fwlink/?LinkId=134155). For information about how to upgrade to Windows Server 2008 failover clustering with SQL Server 2008, see the "Upgrading Failover Clusters" section in Chapter 4, "High Availability."

### Upgrading Multiple Instances

You might have multiple instances of SQL Server 2000 or SQL Server 2005 on a stand-alone or clustered server. Multiple instances can make the upgrade process more difficult, so you must consider them before the upgrade. The options are simple: Should all instances be upgraded in one maintenance window, or should you upgrade them individually?

Upgrading them all at the same time will minimize overall deployment resources because you have to schedule only a single outage. But that also means that if anything goes wrong for any reason, you might have more to fix and deal with, which could increase downtime. It is a risk versus reward decision.

### Upgrading Very Large Databases

Many SQL Server databases are in the hundreds of gigabytes or terabytes range. This presents special challenges in any upgrade process because you have to account for constraints in time and hard disk space when you deal with large amounts of data in a short window of maintenance time. Databases in the terabyte range can potentially take days to copy over a network—even the fastest networks. These VLDBs might be mission-critical databases powering the largest systems in your business and might tolerate very little downtime. When an upgrade has to occur over a weekend, you might have to use several techniques and put in many preparation hours to meet the required time frames for success. You might have to revise the upgrade window if the upgrade will not fit within it.

When you work with VLDBs, more than any other scenario except perhaps high availability, careful planning is required to ensure a successful upgrade experience. The cost of a failure if there is an unexpected issue is much larger because of the time involved to resolve the issue. For example, if a copy of a database backup file stops half way through, or if a stored procedure uses discontinued Transact-SQL syntax, you will lose time. With an in-place upgrade, you gain more time because you do not have to physically move the database. However, this upgrade method makes a restore costlier if there is a serious failure causing you to roll back to the original database. Advance testing in a full-size test or staging environment is very important in these cases.

For more information about VLDB upgrades involving failover clustering or database mirroring, see Chapter 4 "High Availability."

### Upgrading High Availability Servers

For issues related to high availability features such as clustering, database mirroring, log shipping, and replication, see Chapter 4 "High Availability." See Chapter 4 even if you do not use these features for high availability. For example, if you have to upgrade systems that use replication, even if the upgrade is not for high availability purposes, you have to review Chapter 4 information.

### Minimizing Upgrade Downtime

For small, simple instances where you can afford maintenance downtime during the business day, it might not be important to take additional measures to speed up the upgrade and minimize end-user downtime. But many times, you will want to take the quickest route to minimize the inconvenience to your business when a business application is offline waiting for the upgrade to be complete. For high availability applications, you might have to take additional measures to absolutely minimize the downtime. For high availability and VLDB upgrades, see the detailed information within Chapter 4, "High Availability," which goes beyond the basic information in this section.

The following steps can help minimize the downtime involved in an in-place upgrade or side-by-side upgrade:

* **Check the legacy SQL Server versions.** Make sure that the legacy instances of SQL Server 2000 or SQL Server 2005 are at the correct service-pack level for an in-place upgrade. If they are not at the correct level, plan to update them before an in-place upgrade (see "Setup Requirements for an In-Place Upgrade" earlier in this chapter).
* **Make sure that installation requirements are met.** SQL Server 2008 Setup also has Windows and other component requirements (see "Setup Requirements for an In-Place Upgrade" earlier in this chapter).
* **Preinstall .NET and Windows components.** If you can do this, install the .NET Framework 3.5 SP1 or a later version and the Windows Installer (MSI) 4.5 beforehand on the target server. Both require a restart. For an in-place upgrade, if a restart in production cannot be enabled except for the upgrade, install them during the upgrade and not earlier. For a side-by-side upgrade to a separate server, installing these components before the upgrade will help reduce downtime. When you run the SQL Server 2008 Upgrade Advisor on a legacy server, a restart will be required if MSI 4.5 must be installed.
* **Preinstall Visual Studio 2008 SP1 or a later version.** If Visual Studio 2008 is installed on the server, make sure that it is at the SP1 level (see "SQL Server Prerequisites Installed by Setup" earlier in this chapter).
* **Preinstall SQL Server 2008 common components.** Install the SQL Server 2008 Common Components (such as the SQL Server 2008 SQL Native Client) before the upgrade. Also consider pre-installing the Management Tools if SQL Server 2008 Management Studio (SSMS) is not required to manage SQL Server 2000 instances. (For more information, see Chapter 2, "Management and Development Tools.")
* **Select the optimal side-by-side upgrade strategy.** In a side-by-side upgrade, SQL Server databases must be transferred during the upgrade, and several strategies are available, such as backup and restore, detach and attach, and log shipping. (For more information, see Chapter 3, "Relational Databases," and Chapter 4, "High Availability.") For Analysis Services, you can use backup and restore (see Chapter 11, "Analysis Services").
* **Use new service accounts.** Create and use new service accounts and groups, if it is necessary, with your SQL Server 2008 implementations. This guarantees account separation from the legacy instances of SQL Server 2000 and SQL Server 2005. An in-place upgrade will not automatically change the legacy instance’s service accounts. Create new domain-based service accounts that have the correct user rights on each server, and after the upgrade, use SQL Server Configuration Manager to update the SQL Server 2008 services to use these new accounts.
* **Check data consistency.** When you upgrade relational databases, run a Full DBCC CHECKDB on databases before upgrading, while the database is online so that you do not affect downtime. (See Chapter 3, "Relational Databases," for more information.)
* **Back up data before and after the upgrade.** Check that your backup media has no errors and is available for restores if it is necessary (see "Plan for Backups" earlier in this chapter).

## Developing an Upgrade Plan

Every upgrade of a production database system that contains valuable data should occur in the context of a good plan. In addition to performing the upgrade-plan tasks we have already covered, you should also follow some other general best practices when planning for an upgrade.

### Treat the Upgrade as an IT Project

If you do not have the luxury of extensive planning or testing and you decide to just upgrade, we recommend that you take some basic precautions so that you can recover from any issues that might arise:

* Schedule a longer maintenance downtime window than you think that you must have. If everything finishes without issue, you will be able to bring users back online faster than they expected. But if unexpected issues arise, you will have more time to deal with them without inconveniencing users.
* Do a complete backup of all the files and installed software on the server in case you have to restore the server to its previous state. Confirm that your backup media is complete and available.
* Do a complete backup of all the user databases on the server in case you have to selectively restore user databases. Confirm that your backup media is complete and available.

After the upgrades, check before you leave to make sure that everything looks to be working. Make sure that applications return to running correctly and that transaction workloads are processed correctly.

It is best not to treat an upgrade informally, as a one-off procedure. Instead, treat it as a major database upgrade. This approach implies the following steps:

* Applying database change control procedures to the upgrade project
* Guaranteeing sufficient integration and quality assurance (QA) testing
* Developing verification and acceptance tests
* Scripting and automating the deployment process and tests as much as you can
* Developing a rollback plan in addition to a priority patch strategy
* Guaranteeing sufficient downtime for the upgrade

The actual steps that you use should comply with the rules and procedures already existing in your organization, but most likely, they will include all the above. The important point is that an upgrade to SQL Server 2008 is not a minor change to your production database system; instead, it should fit into existing procedures for major database changes.

#### Updating Skills

Before you try to upgrade to SQL Server 2008, make sure that those administering or deploying SQL Server 2008 are ready. Just as you would with any other application, never assume that your staff can deploy and then manage the upgraded system without being correctly prepared.

Before deployment, set up a SQL Server 2008 environment so that everyone who has to update his or her skill set can become familiar with the new version. If the database administrators (DBAs) are comfortable with SQL Server 2008 by the time of production deployment, the transition from SQL Server 2000 or SQL Server 2005 will go much more smoothly.

There are many resources that are available to update skills. For example, there is free access to more than 100 hours of training material on the [SQL Server 2008 Jumpstart](http://sqlserver2008jumpstart.microsofttraining.com) technical training site. Presentations, recordings, hands-on labs and demonstrations cover Overview Sessions, Database Infrastructure & Scalability, Business Intelligence, Developer & Software + Services, and Application Compatibility & Upgrade.

#### Documenting the Upgrade Plan

Work as part of a team, document the planning process, and communicate the upgrade effectively. Team members and stakeholders usually include not only DBA staff, but Operations staff, QA staff, the Security officer, and application/business owners. Explicitly document requirements and establish agreement from relevant stakeholders, just as for any major database server change. Use the documentation as a basis to execute the upgrade during the deployment phase. The plan should be as detailed as possible, and you should store the resulting document or documents by using some form of change control such as a source control system. In the rest of this section, we will detail these steps.

When documenting the plan, include the upgrade requirements in addition to the rationale for choosing an upgrade strategy for each instance or class of instances. Use the rest of the plan to detail remaining issues. For example, the plan might include the following steps:

* **Identify pre-upgrade tasks.** This step includes installing Microsoft Windows Installer (MSI) 4.5, .NET Framework 3.5 SP1, and the SQL Server Native Client on the target instances. Because these steps do not affect the application, they can occur before the actual upgrade deployment begins. Be aware that installing MSI 4.5 requires a restart of the server.
* **Establish performance baselines.** If performance is an important feature of the current legacy system, collect data that indicates typical performance measurements for important and common queries. Refer to these baselines after the upgrade if reports show that performance has changed. Users might be mistaken, and you might find through the baselines that the new system performs equally well or better. See the "SQL Server 2008 Upgrade Assistant" coverage in the "Upgrade Tools" section earlier in this chapter for a tool that can help establish performance and compatibility baselines.
* **Estimate required downtime.** The deployment of an upgrade will involve some downtime for the targeted database servers. When you perform the actual upgrade, allow for enough downtime so that the processes can be completed successfully. Try to give yourself time to roll back in case an unexpected issue arises that cannot be resolved in the downtime window. This might mean that you reach a go/no-go deadline within your downtime window where you must decide whether to finish the upgrade or roll back.
* **Develop upgrade checklists.** The server environment for targeted database servers might have their own infrastructure complexities. Detail the steps that you must follow for taking the systems offline for a while and bringing them back online. Also detail the steps to take during the upgrade processes. The upgrade steps, in particular, might be more complex. Regardless of which method that you use, you might have to apply scripts at certain important points to resolve issues that are identified by Upgrade Advisor. (For more information about how to develop checklists, see the "Upgrade Checklists" section later in this chapter.)
* **Identify backup and restore operations.** Although a part of the deployment process, this task is worth calling out separately. One of the first steps in the deployment plan should be to back up the targeted databases. Also verify the backups, and have a strategy for restoring them if it is necessary.
* **Determine upgrade validation criteria.** Clearly state what criteria your organization will use to validate that the upgrade was successful—in other words, that it produced the result expected. This might consist of scripts run to inspect the instances of SQL Server 2008 and verify that issues are resolved, that configuration settings are as expected, and so on. It might include bringing applications online selectively, and processing some transactions to confirm successful operation.
* **Design final acceptance criteria.** The upgrade might succeed at the instance of SQL Server 2008 level, but some other unaccounted-for variable in the server infrastructure might still prevent applications from running correctly. Whatever the case, determine how the organization will accept the upgrade, and how it will make the "go/no-go" decision. This goes beyond validating the upgrade result: It focuses on whether the applications that use the targeted database servers run as expected and required. It might be appropriate to enlist the support of the QA team to develop appropriate acceptance tests.
* **Design a rollback plan.** If the upgrade is unsuccessful or if the acceptance tests are unsuccessful, be prepared to back out of the process and restore the original conditions. This is much easier to do with a side-by-side strategy than with an in-place upgrade. However, the importance is clear. For example, with an in-place upgrade, a rollback plan might require restoring a disk image (also known as a "ghost" image) of the computer that is running SQL Server 2000 or SQL Server 2005, and then restoring the SQL Server 2000 or SQL Server 2005 databases from the deployment backup.
* **Identify post-deployment steps.** Even after the upgrade is validated and accepted, you might have some remaining tasks to perform, such as updating statistics in the relational database or rebuilding cubes in SSAS. You might also have to reconfigure log shipping, reconfigure database mirroring, reestablish replication, test a failover cluster, or verify that certain SQL Server Agent jobs run correctly.

### Minimize Variables Involved in the Upgrade

As an experienced IT professional, you probably realize that the more changes started in a project, the more complex the testing becomes and the higher the risk of problems. The same is true of upgrading. The simpler the upgrade process, the better its chance of success without much intervention. On the other hand, the more complex, the more likely you will want to make the additional effort to ensure a successful upgrade outcome.

However, because an upgrade to SQL Server 2008 involves some downtime, you might be tempted to "slipstream" in other changes to the databases or database server. If you decide to do this, your required testing will be accordingly more complex and time-consuming. You will find the upgrade process easier if you can minimize changes to only those required only for the upgrade process, if your downtime windows let you do this.

Here are some key variables to consciously decide to add to an upgrade process and the new system. Some changes might occur at the SQL Server-instance level:

* **Change of version.** First, you must deal with the consequences of changing SQL Server versions. Upgrade Advisor details these issues and is the best source for this kind of information. These changes cannot be avoided. For more information, see the previous coverage of Upgrade Advisor and backward-compatibility issues.
* **Change of edition.** You also have to consider the effects of upgrading and at the same time changing the SQL Server edition you are running. This combination of changes can have significant consequences. If you are using an in-place upgrade, the result will be at the same edition level or higher. Even this might present issues. For example, if you are upgrading from SQL Server 2000 MSDE to SQL Server 2008 Express, you can no longer rely on SQL Server Agent jobs. In this case, you might be better served by upgrading to SQL Server Workgroup instead of SQL Server Express. If you choose a side-by-side upgrade, you can change edition level in any direction. For example, you can transition by using a side-by-side upgrade from a SQL Server 2000 or SQL Server 2005 Enterprise two-node failover cluster to a SQL Server 2008 Standard two-node failover cluster. However, in that case, you would lose access to Enterprise features that you previously had available. Generally, reduce the number of changes to the system by staying at the same edition level unless your organization needs a different edition’s features.
* **Change of kind of instance: default or named.** If you perform an in-place upgrade, the kind of instance will remain the same. If you plan an upgrade of a default instance of SQL Server 2000 or SQL Server 2005, the result will be a default instance of SQL Server 2008. However, if you choose a side-by-side upgrade onto two servers or even on a single server, you could upgrade a default instance to a named instance, or vice versa. Changing from a default to a named instance is a variable that could add complexity to the upgrade and require changes in applications and users who are connecting to the resulting instance. Do not try this change unless you are prepared for its effects.
* **Change of configuration.** Making changes to the SQL Server configuration at the server level might make the upgrade more difficult. For example, suppose that on the current system, there is no value set for the maximum degree of parallelism. But in the new system, you decide you want to set it to 4. Such a change could adversely affect the behavior of some queries. To make sure that the upgrade is under control, do not change configuration values unless you are reasonably sure of the consequences.
* **Change of authentication**. You might want to change the kind of authentication that is used for a given database server, such as changing the server to use only Windows authentication. However, that kind of change would be better reserved for another time unless there is no possibility that the change will harm the system. If the authentication change causes problems, users might mistakenly blame the upgrade process.

In a complex upgrade scenario, you can reduce the risk of problems by keeping all server-level configurations the same between instances and avoiding new introductions to the system until after the upgrade is considered successful.

At the database level, possible changes during a direct in-place upgrade or side-by-side upgrade include the following:

* **Consolidation or distribution of databases.** Minimizing changes at the database level will make the upgrade process smoother. If you can do this, make consolidation or distribution of databases a separate project.
* **Change of database compatibility level.** Keep the new databases at the same compatibility level, and move them to the new compatibility level after you have validated the upgrade’s success.
* **Change of database objects.** Avoid making upgrades to the database schema, structure, or code objects part of the upgrade project.

If you decide to make these kinds of changes at the database level, we recommend that you design and execute a test of the upgrade and database changes in a test or staging environment before you try it in production. When you introduce other variables into the upgrade process, it might be difficult to separate the success of the upgrade from any problems associated with the other changes.

You might also be tempted to change the database server infrastructure while you are upgrading. These changes might include the following:

* **Change of physical server.** For an in-place upgrade, you must use the same server. However, in a side-by-side upgrade, you can put the new instance on the same server or a different server. If on a different server, such a change implies a change of IP address and server name in the network, unless you perform a rename and readdress of the servers in some manner. Users and applications must now adapt to the new server name and IP address, which might be a complexity that you cannot avoid.
* **Swapping servers.** For a side-by-side upgrade, you might decide to keep the same server and instance name by pulling the legacy server out of the domain, adding in the new server, giving it the same IP address as the legacy server, and renaming the new server to the legacy server name. However, with an in-place upgrade, in which the new instance of SQL Server 2008 becomes a named instance, be aware that you cannot rename an instance name or change a named instance to a default instance.
* **Change of CPU type: 32-bit to 64-bit.** You might also view a side-by-side upgrade as an opportunity to put the new instance on a 64-bit server instead of your current 32-bit server. In this case, be prepared to install the correct editions and be aware of any restrictions, as described earlier in this chapter.
* **Changing hardware on the server.** You might see an in-place upgrade as an opportunity to change server hardware, such as memory, number of CPUs, or disk configuration. Such changes could affect the resulting user acceptance of the upgrade if problems arise. Only make these changes if you can fully test their effects beforehand.
* **Change of Windows version.** Changing the Windows version in a side-by-side upgrade might have unintended consequences if the new operating system is configured incorrectly and in the same manner as the legacy computer. Again, you must test this kind of change beforehand. For an in-place upgrade, upgrade the operating system first, and make sure that the legacy SQL Server is stable on the newer version of Windows before upgrading to SQL Server 2008. (For more information, see "Upgrading Both Windows and SQL Server" earlier in this chapter.)

**Note**: If your legacy instance of SQL Server 2000 or SQL Server 2005 is installed on Windows 2000 Server, you must upgrade to a new server by using a side-by-side upgrade because SQL Server 2008 is not supported on Windows 2000 Server.

* **Moving to a failover cluster.** For information about failover clustering, see Chapter 4, "High Availability."
* **Installing database mirroring, replication, or log shipping.** For information, see Chapter 4, "High Availability."

Just remember: The fewer changes that are introduced to the infrastructure during the upgrade process, the less complex an upgrade will be. The more changes that you introduce concurrently, the more testing you will want to do in your test or staging environment. If you absolutely must combine multiple changes within the upgrade process, plan for additional testing in your test or staging environment to reduce the risk of encountering unexpected issues during the production upgrade.

### Create Upgrade Checklists

A key component of a complex application upgrade in IT shops is a checklist of activities. Because many of the steps that are required for an upgrade to SQL Server 2008 are the same as would be required for a major application upgrade, existing application upgrade checklists can become the basis for SQL Server upgrade checklists. For a sample checklist for several different upgrade tasks, see Appendix 2 Upgrade Planning Deployment and Checklist.

* When you develop an upgrade checklist, consider the following points:
* Classify instances of SQL Server into classes, based on how important the data is. Consider a possible scenario in which a given SQL Server instance could be classified into one of three levels. Perhaps only the top-level systems use high availability technologies, the mid-level systems might use log shipping for creating warm standby servers, and the lowest-level databases might require only nightly backups. In this scenario, the checklist for upgrading the top-level instances of SQL Server would be much more extensive and require full testing and a complex rollback plan. In contrast, upgrading the lowest-level instances of SQL Server would have a much simpler checklist.
* Incorporate application steps and SQL Server integration into upgrade checklists. For example, if a given instance of SQL Server is involved in replication as a subscriber, make sure that you add steps into the upgrade checklist for removing the legacy instance SQL Server as a subscriber, putting the new instance in its place, and testing the result to ensure that replication is still running.

The other chapters in this guide and SQL Server 2008 Books Online contain many topics that contain valuable information that can help in developing upgrade checklists:

* Database Engine: [Considerations for Upgrading the Database Engine](http://go.microsoft.com/fwlink/?LinkId=134156) and Chapter 3, "Relational Databases"
* Management and Development Tools: See Chapter 2, "Management and Development Tools"
* Clustered environments: [Upgrading a SQL Server 2008 Failover Cluster](http://go.microsoft.com/fwlink/?LinkID=132180) and the section on failover clustering in Chapter 4, "High Availability"
* Replication: [Considerations for Upgrading Replicated Databases](http://go.microsoft.com/fwlink/?LinkID=132650) and the section on replication in Chapter 4, "High Availability"
* Analysis Services: [Considerations for Upgrading Analysis Services](http://go.microsoft.com/fwlink/?LinkID=134170) and Chapter 11, "Analysis Services"
* Integration Services and DTS: [Considerations for Upgrading Data Transformation Services](http://go.microsoft.com/fwlink/?LinkID=132615) and Chapter 13, "Integration Services"
* Reporting Services: [Considerations for Upgrading Reporting Services](http://go.microsoft.com/fwlink/?LinkID=134175) and Chapter 14, "Reporting Services"

### Test the Upgrade Plan

Testing is easy to propose but frequently difficult to perform in practice, usually because of its complexity and other pressing work. If your data is valued, test the upgrade plan fully in a test or staging environment before you try it in production.

Make sure that you test the plan with the people who will actually perform it. Testers should be familiar with the upgrade plan. Trying to troubleshoot mistakes because of unfamiliarity during the deployment process is stressful and costly.

Work with your QA team to make sure that the testing occurs. QA departments frequently already have acceptance criteria, smoke tests, and so on that they apply during application upgrades. Check with the QA team for help and support for a SQL Server upgrade also.

Here are some general considerations for testing either an in-place upgrade or side-by-side upgrade:

* Begin by building a test or staging environment. Consider putting the test server or servers in a test environment that has its own domain so that you can use the same server and SQL Server instance names. Make sure that the test servers match production in disk volume assignments and free disk space. This is especially important in an in-place upgrade or a side-by-side upgrade on the same server. In a side-by-side upgrade to a different server, you might use the additional server for testing as well.
* Test the upgrade multiple times. To make repeated testing easier, you could reset the test environment back to its original state quickly. Making a disk image of the database server, in addition to copies of all data files, could be helpful. Then, restore the ghost image and data file copies to the original servers. Or, you could use a Virtual PC (VPC) image or images to build the test environment, if the VPC image really can support all the components and behavior of the production server. Reverting the VPC image to its original state when closing it down makes a second test run much easier.
* Install the .NET Framework 3.5 SP1 and SQL Server 2008 drivers on the production server by using SQL Server 2008 Setup before the upgrade process, if that is possible and matches the upgrade plan for production. This saves time during the actual upgrade.
* Use Upgrade Assistant to establish baseline performance data on the test server that uses the legacy SQL Server 2000 or SQL Server 2005 database and to then compare the results of an upgraded version of the application that uses SQL Server 2008. See the "SQL Server 2008 Upgrade Assistant" section earlier in this chapter for more information.
* Execute Upgrade Advisor remotely against the test server that runs the legacy instance of SQL Server 2000 or SQL Server 2005 and validate that its output matches what is received when it is executed against the production system. Then, apply scripts to fix blocking issues that Upgrade Advisor reveals. In some cases, fixing blocking issues might break the legacy application. If that is the case, apply additional scripts or code as workarounds to update the database code or the application so that it continues to operate. Again, it is important to verify that the application operates correctly. You can run Upgrade Advisor by itself, but Upgrade Assistant will also start it. For more information, see "SQL Server 2008 Upgrade Advisor" earlier in this chapter.
* When the upgrade is complete, apply any later scripts that might be required to resolve post-upgrade issues uncovered by Upgrade Advisor.
* Upgraded relational databases will have a compatibility level of 80 if you upgraded from SQL Server 2000, or 90 if you upgraded from SQL Server 2005. At this point, make sure that databases have the compatibility level that your company requires for continued production after the upgrade.
* As soon as the test server reaches its final state of the upgrade process, test all relevant applications that are running against the upgraded instance of SQL Server 2008.
* Test the process of rolling back the in-place upgrade to the original SQL Server version so that you are confident of the rollback process in case you have to revert the upgrade during the maintenance window.

#### Testing an In-Place Upgrade

Some additional considerations for testing an in-place upgrade include the following:

* To test an in-place upgrade, put a copy of the production instance of SQL Server 2000 or SQL Server 2005 on a test server so that your test includes "real" data and objects. To the best of your ability, make the initial state of the test server duplicate all necessary components of the production server.
* After the test environment is built, verify that it behaves correctly with the same, or copies of, application components that connect to the production system.

#### Testing a Side-by-Side Upgrade

There are some special considerations for testing a side-by-side upgrade. A side-by-side upgrade can be more complex because more SQL Server instances will be involved, either on one server or on two distinct servers.

Whether you perform a side-by-side upgrade on the same server or separate servers, consider the following points:

* A side-by-side upgrade requires manually transferring data and components to the instance of SQL Server 2008. After you install the parallel instance of SQL Server 2008, test the process of manually transferring components to the new instance.
* To repeat the side-by-side upgrade tests, you might not have to restore the test server from a ghost image. You could uninstall SQL Server 2008 and remove the data files to set the server back to its original state and repeat the upgrade tests. Make sure that you test uninstalling SQL Server 2008, and verify that the legacy version of SQL Server 2000 or SQL Server 2005 is working correctly.
* In a side-by-side upgrade, the rollback will most likely be to the original instance of SQL Server 2000 or SQL Server 2005. Make sure that you test the rollback scenario.

In a side-by-side upgrade on a single server, consider the following additional items:

* You could run the legacy SQL Server instance in parallel with the new instance of SQL Server 2008. If that is part of the upgrade plan, make sure in the test environment that running in parallel will not require too much of the server's resources.
* When the cutover to the new instance of SQL Server 2008 occurs and the instance is verified as ready for production, stop the legacy instance of SQL Server, leaving it dormant for a while as a potential rollback instance.
* After the upgrade has passed acceptance tests, uninstall the legacy instance without disturbing the new production instance.
* Decide whether the production system will be online during the installation of SQL Server 2008. If this is the case, test the effect that SQL Server 2008 Setup has on application performance.

In a side-by-side upgrade to a different server, consider the following additional item:

* If the upgrade plan includes removing the old server from the domain and renaming the new server with the legacy name and legacy IP address, test this step as well.

Now, perform final tests on the new SQL Server 2008 server. To rerun the tests and gain confidence in the plan and deployment, repeat the process by restoring the baseline target server.

### Develop Acceptance Criteria and Rollback Steps

An upgrade plan must consider all possible possibilities. Some might be ignored as improbable, but for all likely scenarios, the highest priorities must be to protect any production data and to be able to restore the system to its original state if it is required. The following tasks help provide that protection:

* **Back up production data**. In the upgrade checklist, include steps for backing up all the databases and other data that would be required to rebuild the system.
* **Develop acceptance criteria and a go/no-go decision point**. As part of the upgrade checklist, at some point decide whether the upgrade is successful and the system can be put back into production. The final go/no-go decision might involve a team of people, including the testers who determine whether the application operates as expected.
* **Have a rollback plan in place**. Specify in sufficient detail how to restore the system if it is necessary. The more detailed the plan, the better, because rollbacks usually occur in high-stress situations. Clearly defined steps are easier to follow in those contexts.
* **Test the rollback**. Test the rollback plan to make sure that it will actually work. The degree of testing might be a function of how important the data is and how time-critical a rollback would be. There can be no confidence in an untested rollback plan.

## Post-Upgrade Tasks

As soon as you have completed the upgrade tasks, two more steps are required:

* Integrating the new SQL Server instance into the application and database server environment
* Determining whether the upgrade was successful

These two steps are not necessarily sequential: For example, you might apply some acceptance criteria immediately to obtain a go/no-go decision. This could then be followed by integrating the new instance and applying the remaining set of acceptance tests.

### Integrate the New Instance into Its New Environment

First, make sure that the new instance of SQL Server 2008 will operate with the applications that need the data. You might have to perform upgrades to the application to get everything working correctly. But in most cases, the transition should be seamless and not require any application changes.

There might be additional requirements affecting the upgrade that depend on the environment of the resulting instance of SQL Server 2008. Some examples include the following:

* **Linked servers.** The current system might depend on linked server relationships and definitions that must be applied for an upgrade. Failures in the application might result if those linked servers are not defined and tested correctly.
* **Imports and exports.** The legacy database system might receive data imports and be the source of data exports. These imports and exports might use DTS, be converted to SSIS, or use other tools. You have to isolate these requirements and make sure the resulting upgraded instance’s correct participation.
* **Components referring to older SQL Server versions.** If selectively transitioning legacy SQL Server instances, make sure that the resulting instance of SQL Server 2008 has components that can still connect successfully to the older SQL Server versions.
* **Drivers required for changing to a 64-bit version of SQL Server.** These required drivers might include drivers for accessing other database systems and mainframes from a 64-bit server.
* **Patches, hotfixes, and cumulative updates.** After you upgrade to SQL Server 2008 from another edition of SQL Server, you must reapply any hotfix or service pack updates to the upgraded SQL Server instance.

### Determine Application Acceptance

Now you are ready to apply the criteria for final acceptance of the upgrade and decide whether the new system can go live. For a complex application, this decision will likely require the input of a QA team to make sure that the applications are running correctly. For simpler applications, you could use "smoke tests" that give brief but reliable results as to the viability of the application. Consider involving your stakeholders (such as application owners) in this kind of final acceptance testing.

### Troubleshooting an Upgrade

The best time to discover problems with an upgrade is when validating the upgrade plan in a test environment. Most upgrade issues related to the static code and objects in the instance, and the techniques recommended to resolve them, are fully documented in Upgrade Advisor. Dynamic code requires a workload to verify and troubleshoot.

General troubleshooting techniques include the following:

* Troubleshooting an in-place upgrade is basically the same as troubleshooting a SQL Server 2008 installation. The SQL Server 2008 Setup program logs a summary of its actions in the Summary.txt file in the Program Files\Microsoft SQL Server\90\Setup Bootstrap\LOG\ folder. The Summary.txt file contains a section for each SQL Server component's installation summary.
* Detailed log files are located in the Files folder underneath the path just mentioned, providing one file for each component and for many subcomponents. For information about interpreting the log files, see [How to: Read a SQL Server Setup Log File](http://go.microsoft.com/fwlink/?LinkId=134490) in SQL Server 2008 Books Online.
* You use the same log files for troubleshooting a side-by-side upgrade as an in-place upgrade, except that the actual data transfer will not be logged because it is a manual process that is not under the control of SQL Server 2008 Setup.
* The steps for troubleshooting a side-by-side upgrade are basically based on the technique that you use for transferring data. If you use the backup/restore or detach/attach side-by-side upgrade method, capturing the output of those processes to text files is possible as long as you use the SQLCMD command-line tool and redirect the output to a file. If you use the Copy Database Wizard to transfer data, the process is interactive, and you have to watch it live for potential errors.

### Decommission and Uninstall After a Side-by-Side or New Hardware Upgrade

After you perform a side-by-side upgrade on a single server or separate server, the old servers and SQL Server instances will still exist. Before uninstalling or decommissioning the server, stop and disable the SQL Server services before going live with the new SQL Server 2008 environment. This makes sure that no application or connection will mistakenly connect to the old instance or server.

**Warning:** If you are not upgrading all databases or components from an instance at the same time, do not disable and shut down any SQL Server instance still running active databases for other applications.

After your new instance has passed acceptance tests and the newly upgraded server is successfully in production, schedule a time to either uninstall the instance or fully decommission the server. If you are uninstalling in a side-by-side upgrade on one server, a restart might be required to fully remove the legacy instance of SQL Server 2000 or SQL Server 2005, and that outage must be scheduled. When uninstalling a legacy SQL Server instance, be very careful to select the correct components. Upgrading to a separate server is generally the better option, because it is easier to decommission a whole server than it is to uninstall on an actively used one.

## Considerations for Upgrading without a DBA

Some groups are tasked with deciding whether to upgrade to SQL Server 2008 without a SQL Server DBA available for support. In other cases, no DBA may be available. In either case, make sure that you determine at what point you should seek additional help.

The key issue is the nature and value of the data that is stored in your current SQL Server instance. If the data is business critical or irreplaceable, address how to back it up before an upgrade and restore it if a rollback is needed in the upgrade process for any reason. As soon as either of these steps is unclear or uncertain, seek the help of a professional DBA with SQL Server 2008 skills.

Assuming that your organization can handle a rollback, the following issues must be dealt with related to testing and acceptance:

* Do you have a clear sense of what criteria would qualify the upgrade as successful?
* Do you have a tester or team of testers who can verify that the applications that depend on SQL Server work correctly after the upgrade?
* Can you test the upgrade in a test environment first so that you can be confident that the upgrade will succeed?
* Above all, have you run Upgrade Advisor and detected and resolved any blocking issues it found?

If you are confident that your organization can back up and restore data, successfully test the results, and rebuild the SQL Server database if it is necessary, next consider what upgrade strategy would best meet your needs.

By far the easiest upgrade strategy without a DBA available is the in-place upgrade, in which the new instance of SQL Server 2008 replaces your legacy SQL Server instance. With this strategy, SQL Server automatically transfers your data from the old to the new instance. In most cases, upgrading an instance without a DBA is best performed by using in-place upgrade. However, you will want to consider all the relevant factors before you make a final decision.

**Upgrade description using a flowchart**

The following flow chart (Figure 1-5) describes the different stages in upgrading your instance of SQL Server 2000 or SQL Server 2005 to SQL Server 2008. The chart also describes basic tools available to help you upgrade to SQL Server 2008.

|  |
| --- |
|  |

**Figure 1-5: Different Stages in the Upgrade Process**

## Conclusion

The key steps in planning for a successful SQL Server 2008 upgrade are as follows:

* Researching the upgrade requirements
* Establishing pre-upgrade tasks
* Establishing upgrade implementation tasks
* Establishing post-upgrade tasks

The result should include the following:

* Choice of an appropriate upgrade strategy
* Checklists for implementing each upgrade
* Tested backup, restore, and rebuild steps
* Clear acceptance criteria, go/no-go decision processes, and conditional rollback steps

As with any significant database application upgrade, an upgrade to SQL Server 2008 will benefit from limiting the number of variables during the upgrade process and applying standard IT production release procedures to the upgrade process.

## Additional References

For an up-to-date collection of additional SQL Server 2008 upgrade planning and deployment references, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkId=132510).

For more information about how to upgrade each SQL Server component—such as the Database Engine, Analysis Services, Integration Services, and Reporting Services—see the remaining chapters in this document.

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkId=132512)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132514)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkId=132515)

# Management and Development Tools

## Introduction

SQL Server 2008 is the culmination of many years of hard work by several Microsoft-focused technical teams. These teams were charged with developing front-office applications, back-office online transaction processing (OLTP) systems, and business information systems that let you effectively analyze and report on these systems. In addition, the architecture teams have made major in-roads in monitoring and controlling all the aspects of the technical environment, such as Microsoft Operations Management (MOM) through Windows Management Instrumentation (WMI).

This chapter covers the new and improved features of the SQL Server Studios—SQL Server Management Studio (SSMS) and Business Intelligence Development Studio (BIDS)—which are the user interface tools used to manage and develop in SQL Server 2008. Introduced in SQL Server 2005, these tools are additions to the longstanding Visual Studio integrated development environment (IDE) for client-side development and are designed to hide excessive detail while maximizing the information available to administrators for the task at hand.

This chapter details the management tools issues you need to consider as you prepare to upgrade your SQL Server 2000 or SQL Server 2005 installations to SQL Server 2008. This chapter also contains references to valuable management tools upgrade resources.

## Feature Changes in SQL Server 2008 Management and Development Tools

When upgrading from SQL Server 2005 to SQL Server 2008, the impact of the upgrade on management and development tools will be minimal. There are some menu changes, but the interface is still similar to Visual Studio. However, if you are upgrading from SQL Server 2000 to SQL Server 2008, note that SQL Server Enterprise Manager and Query Analyzer have been replaced by two new Visual Studio-based tools called the SQL Server Studios.

A *studio* provides an environment for management or development. The SQL Server Studios are:

* SQL Server Management Studio (SSMS) is for managing all SQL Server 2008 components, including relational database and business intelligence (BI).
* Business Intelligence Development Studio (BIDS) is for development experience for BI-related components.

If you are new to Management Studio, the user interface might seem disorienting at first sight. However, you will soon realize that it has familiar elements buried within the layout. You just need to start with what you know and then extend from that as your confidence in the environment grows. Developers will find the layout of the BI Development Studio environment familiar because it is an extension of Visual Studio. But SQL Server 2000 database administrators who have used only Enterprise Manager and Query Analyzer have a slightly steeper learning curve. For more information about the SQL Server Studios, see [SQL Server Studios Overview](http://go.microsoft.com/fwlink/?LinkId=134160)in SQL Server 2008 Books Online.

In addition to looking at Management Studio and BI Development Studio, this chapter will also cover the following management tools you need to consider as you upgrade SQL Server:

* SQL Server Configuration Manager
* SQL Server Profiler
* Database Engine Tuning Advisor
* Command- prompt tools

There are many other tools in SQL Server 2008. To describe them in detail is beyond the scope of this document. For more information about all the tools, see [Feature and Tools Overview](http://go.microsoft.com/fwlink/?LinkId=134161).

### SQL Server Management Studio 2008 Changes

SQL Server Management Studio is an integrated environment for managing all SQL Server components: the SQL Server Database Engine, SQL Server Reporting Services (SSRS), SQL Server Integration Services (SSIS), and SQL Server Compact, as well as developing Transact-SQL database queries and components. Management Studio uses the Visual Studio IDE to give administrators a single, easy-to-use graphical tool for managing the most sophisticated systems and to give developers a consistent experience for database and application development.

#### Consolidating SQL Server 2000 Enterprise Manager and Query Analyzer

For those still using the SQL Server 2000 tools, one of the biggest changes in SQL Server 2005 and SQL Server 2008 is the consolidation of SQL Server 2000’s Enterprise Manager, Query Analyzer, and the MDX Sample Application—as well as some Analysis Services Manager features—into Management Studio. The functionality of both Enterprise Manager and Query Analyzer has been incorporated into subsets of Management Studio. To see how to work with Management Studio, see [Tutorial: SQL Server Management Studio](http://go.microsoft.com/fwlink/?LinkId=134162) in SQL Server 2008 Books Online.

**Note:** SQL Server Management Studio can be customized to make it appear more similar to the familiar SQL Server 2000 tools. You can hide all the new Management Studio components except for the object browser and the query editor, making the initial learning curve easier. For instructions about how to customize the Management Studio environment, see [Changing the Environment Layout](http://go.microsoft.com/fwlink/?LinkId=134163) in SQL Server 2008 Books Online. We cover the key Management Studio elements later in this chapter.

#### Registered Servers

In SQL Server 2008, you register servers by using the Registered Servers component of Management Studio. To add SQL Server 2000 Enterprise Manager registered servers into Management Studio after an in-place upgrade, right-click the Database Engine node in the Registered Servers window, and select Previously Registered Servers.

#### Object Explorer

In Management Studio’s Object Explorer, you will find the following enhancements:

* **Customizable columns.** You can display in the browser the information that is relevant to your needs rather than having to accept what is provided as a default.
* **Object property information.** Additional details of the selected object are displayed in a property window.
* **Object sorting.** By simply clicking an object column heading, you can sort objects into the desired order based on the context of the click.
* **Enhanced manipulation of objects and object groups in a detail window.** You can
  + Move backward and forward through related objects
  + Move upward to parent objects
  + Synchronize the object selected in the detail window with the ones in the object window
  + Filter to show subsets of objects
  + Select the scope of an object group based on the level of object selected in Object Explorer
  + Search for objects using wildcards

For more information about Object Explorer enhancements, see the "Object Explorer" section in [Manageability Enhancements (Database Engine)](http://go.microsoft.com/fwlink/?LinkId=134164) in SQL Server 2008 Books Online.

#### Query Editor

For the Query Editor, you will find the following enhancements:

* **Transact-SQL Query Editor IntelliSense.** The Transact-SQL Query Editor provides IntelliSense functionality such as word completion and error underlining. IntelliSense is provided for frequently used Transact-SQL elements and will be extended to other Transact-SQL elements in future releases.
* **Database Engine Error List window.** Management Studio includes an Error List window that displays the syntax and semantic errors generated from the IntelliSense code in the Transact-SQL Query Editor.
* **Color-coded status bar.** The status bar changes depending on activity. For example, when performing a distributed query, the color changes to indicate this.
* **Customizable editor title bar.** The title bar can include the names you want instead of being predefined.

For details about these Query Editor enhancements, see the "Query Editor" section in [Manageability Enhancements (Database Engine)](http://go.microsoft.com/fwlink/?LinkId=134164) in SQL Server 2008 Books Online.

#### Keyboard Shortcuts

Database administrators managing SQL Server 2000 environments often use keyboard shortcuts in Enterprise Manager and Query Analyzer to increase their productivity. Database administrators upgrading to SQL Server 2005 or SQL Server 2008 will find that some of these shortcuts have changed.

If you want to keep your SQL Server 2008 environment similar to your SQL Server 2000 or SQL Server 2005 environment, you can change the Management Studio default standard keyboard scheme to the SQL Server 2000 or SQL Server 2005 keyboard scheme. To change the keyboard scheme in Management Studio, follow these steps:

1. From the Tools menu in SQL Server Management Studio, click Options.
2. Expand the Environment node, and highlight the Keyboard page.
3. Change the keyboard scheme by using the Keyboard Scheme drop-down box.

If you change the keyboard scheme in Management Studio, note that the following SQL Server 2000 and SQL Server 2005 shortcuts are not available in SQL Server 2008:

* CTRL+O (Open a new query editor window)
* CTRL+SHIFT+P (Insert the body of the specified file into the window)
* CTRL+K (Include actual execution plan in the query output)
* CTRL+SHIFT+O (Open the Query Option dialog box)

For a list of keyboard shortcut changes, see [SQL Server Management Studio Keyboard Shortcuts](http://go.microsoft.com/fwlink/?LinkId=134165) in SQL Server 2008 Books Online.

#### Database Diagrams

During an in-place upgrade to SQL Server 2008, database diagrams created on earlier releases of SQL Server will automatically be upgraded but not activated. The following steps describe how to set up database diagramming on SQL Server 2008 to complete the upgrade. Database administrators wanting to set up database diagramming on a SQL Server 2008 database must be a member of the sysadmin fixed server role or the db\_owner role for each database they want to configure:

1. From Object Explorer in SQL Server Management Studio, expand the database you want to configure.
2. Expand the Database Diagram node under the database connection.
3. Select Yes when prompted to create the support objects.
4. When you then open the database diagrams, SQL Server 2008 automatically upgrades them.

For details about upgrading database diagrams, see [How to: Upgrade Database Diagrams from Previous Editions (Visual Database Tools)](http://go.microsoft.com/fwlink/?LinkId=134166) in SQL Server 2008 Books Online.

#### Index Tuning Wizard

In SQL Server 2005 and SQL Server 2008, the Index Tuning Wizard has been replaced by the Database Engine Tuning Advisor. If you are using the tool from the command line, this means that itwiz.exe has been replaced by the dta.exe command-line utility. Database administrators should review utility scripts and replace itwize.exe calls with dta.exe calls.

Database administrators should also note the inability of dta.exe to tune workload tables on remote servers. You should use one of the following two options to modify utility scripts that call the itwiz.exe command-line utility with the –t argument:

* Use a trace file instead of a remote trace table, and modify the script to execute against a file instead of a trace table.
* Copy the remote trace table to an instance on the local server, and modify the server connection.

For a detailed comparison of the Index Tuning Wizard and the Database Engine Tuning Advisor, see [Differences Between Database Engine Tuning Advisor and Index Tuning Wizard](http://go.microsoft.com/fwlink/?LinkId=134167) in SQL Server 2008 Books Online.

#### Customer-Requested Enhancements in SQL Server 2008

Note that you can now perform the following actions in Management Studio 2008 as a result of customer requests:

* Query against multiple servers simultaneously.
* Access SQL Server Profiler from the Query Editor window.
* Return Top *n* rows in a result set.
* Configure the number of rows that are returned by the Open Table dialog box.
* Specify the action that results by double-clicking a table in Object Explorer.
* Disable the table designer from recreating tables when you are implementing design changes.
* Additional Connection Parameters option in Connect to Dialog box of Management Studio 2008. For more information, see [Connect to Server (Additional Connection Parameters Page)](http://go.microsoft.com/fwlink/?LinkId=134168) in SQL Server 2008 Books Online.

For the latest information about Management Studio features that you will find after an upgrade, see [Manageability Enhancements (Database Engine)](http://go.microsoft.com/fwlink/?LinkId=134169) in SQL Server 2008 Books Online.

### Business Intelligence Development Studio2008 Changes

BI Development Studio is the SQL Server 2005 and SQL Server 2008 environment for developing BI solutions that use SQL Server Analysis Services (SSAS), SQL Server Integration Services (SSIS), and SQL Server Reporting Services (SSRS). BI Development Studio is essentially Visual Studio 2008 with additional project types specific to SQL Server 2008 BI, giving you templates for creating objects for BI solutions as well as the designers, tools, and wizards you need to work with those objects. Let’s do a quick survey of the tools in BI Development Studio to help as you plan your upgrade strategy.

**Business Intelligence Development Studio Replacement**

SQL Server 2008 Setup will *not* replace BI Development Studio 2005 tools with BI Development Studio 2008 tools.

* SQL Server 2008 setup will install BI Development Studio 2008 side-by-side with BI Development Studio 2005
* BI Development Studio 2008 enables you to create, edit and deploy Business Intelligence projects to SQL Server 2008 instances of Analysis Services, Integration Services, and Reporting Services.
* BI Development Studio 2005 lets you create, edit and deploy Business Intelligence projects to your instances of SQL Server 2005 Analysis Services, Integration Services, and Reporting Services.
* BI Development Studio 2008 requires Visual Studio 2008 Service Pack 1. If Visual Studio 2008 RTM is present on the computer, you need to upgrade Visual Studio 2008 RTM to Visual Studio 2008 SP1 in order to install BI Development Studio 2008. If Visual Studio 2008 is not present on the computer, SQL Server 2008 Setup will install the Visual Studio 2008 SP1 shell. For more information on the dependent components, see the SQL Server 2008 release notes.
* BI Development Studio 2008 installs Visual Studio Tools for Applications 2.0, which is used for SSIS projects. Visual Studio Tools for Applications 2.0 is a replacement for Visual Studio for Applications.
* BI Development Studio 2005 on the upgraded instance must be removed separately using Add/Remove Programs. This applies whether there are one or many instances of SQL Server 2005 on the server.

**Business Intelligence Development Studio Project Files**

BI Development Studio 2008 can open projects created with BI Development Studio 2005.  BI Development Studio 2008 upgrades the BI Development Studio 2005 projects to the BI Development Studio 2008 components. After the projects have been upgraded for 2008 components, you can edit and update the projects in BI Development Studio 2008. After the projects have been upgraded, they can work only with SQL Server 2008 components. BI Development Studio 2008 is not designed to work with SQL Server 2005 instances. You need to use BI Development Studio 2005 to work with SQL Server 2005 components. As such, it is highly recommended for customers to back up their BI Development Studio 2005 projects before opening and saving them in BI Development Studio 2008.

#### Business Intelligence Development Studio Replaces SQL Server 2000 Analysis Services Manager and Other BI Tools

SQL Server 2000 provides Analysis Services Manager for managing BI solutions, but for performing extraction, transformation, and loading (ETL) activities or reporting, you need to use Enterprise Manager or external tools. For SQL Server 2005 and SQL Server 2008, the tools essential to BI are collected into a common interface: BI Development Studio.

#### SQL Server Analysis Services - Multidimensional Data

The SSAS elements of the BI Development Studio toolset are designed to optimize the analysis of and reporting on data using online analytical processing (OLAP) techniques. In OLAP, you copy the OLTP data and restructure it into a different model that is easier to create PivotTables on and produce ad hoc reports from. SQL Server 2008 features various improvements to the tools in this area, including optimization of data aggregation through enhanced design techniques, more support for backing up and restoring, and the ability to provide customized extensions to the tools.

For more information about upgrading to SSAS 2008, see Chapter 11, "Analysis Services," and [Considerations for Upgrading Analysis Services](http://go.microsoft.com/fwlink/?LinkId=134170) in SQL Server 2008 Books Online.

#### SQL Server Analysis Services - Data Mining

BI Development Studio also includes data mining tools to help BI developers, administrators, and business analysts set up environments of raw data that can be investigated by software algorithms to identify relationships between elements that might not be obvious through the use of common business sense and pivot tables. Data mining can take the business user beyond typical analysis to find information that would be difficult to find without a huge amount of effort.

For detailed coverage of the upgrade considerations for SQL Server 2008 data mining, see Chapter 12, “Data Mining,” and [SQL Server Analysis Services—Data Mining](http://go.microsoft.com/fwlink/?LinkID=132525) in SQL Server 2008 Books Online.

#### SQL Server Integration Services

There are many new features available in SSIS, but if you are coming from the SQL Server 2000 environment, make sure you are familiar with the new interface in BI Development Studio. For information about upgrading to SSIS 2008, see Chapter 13, “Integration Services.” You can find more details about these and other new features in [What's New (Integration Services)](http://go.microsoft.com/fwlink/?LinkId=134171) in SQL Server 2008 Books Online.

#### SQL Server Reporting Services

SSRS 2008 provides the new Reporting Services Configuration tool for configuring an SSRS installation. Start the tool from the Configuration Tools group within the Microsoft SQL Server 2008 Start menu group. When you upgrade SSRS from a previous version of SQL Server, you can use this tool to update the Report Server database to the new SQL Server 2008 format. For moreinformation about how to use this tool, see Chapter 14, “Reporting Services,” and [Reporting Services Configuration Tool](http://go.microsoft.com/fwlink/?LinkId=134172) in SQL Server 2008 Books Online.

SSRS 2008 also comes with an ad hoc reporting tool called Report Builder, which lets users create, edit, and deploy their own reports. A browser-based client application, Report Builder gives business users an intuitive model of their data, predefined templates, and a friendly interface for building ad hoc reports. A new version, Report Builder 2.0 for SQL Server 2008, is also scheduled for release. For more information about this tool, see [Report Builder 2.0](http://go.microsoft.com/fwlink/?LinkId=134173) in SQL Server 2008 Books Online.

### SQL Server Configuration Manager

The SQL Server 2000 Service Manager and Network Protocol tools are now grouped under the SQL Server Configuration Manager tool, which is a Microsoft Management Console (MMC) snap-in. It manages connectivity components and replaces the following tools from pre-SQL Server 2005 releases of SQL Server:

* The Server Network Utility
* The Client Network Utility
* The Service Manager

You can use SQL Server Configuration Manager to start, stop, pause, and resume SQL Server services, change the service properties, and configure SQL Server network protocols.

SQL Server Configuration Manager manages all SQL Server services, including those for the Database Engine, SSAS, SSRS, SSIS, SQL Server Agent, and SQL Server Browser. For more information about the latest Configuration Manager features, see [SQL Server Configuration Manager](http://go.microsoft.com/fwlink/?LinkId=134174) in SQL Server 2008 Books Online.

### Reporting Services Tools

In preparation for upgrading to SQL Server 2008, database administrators working with SSRS should note that there might be certain expected elements that are no longer available. For example, the Rsactivate.exe command-line utility no longer initializes a single report server instance or multiple report server instances that are part of the same Web farm because this functionality is obsolete in SQL Server 2008. In addition, SQL Server 2008 does not support the Report Server Windows Management Instrumentation (WMI) Provider from SQL Server 2000; SSRS 2008 includes a new version of the WMI provider.

For more information about these changes, see Chapter 14, “Reporting Services,” and the following topics in SQL Server 2008 Books Online:

* [Considerations for Upgrading Reporting Services](http://go.microsoft.com/fwlink/?LinkId=134175)
* [Upgrade (Reporting Services)](http://go.microsoft.com/fwlink/?LinkId=134176)

### DTS Tools

Although you can migrate DTS packages to SQL Server 2008 SSIS packages, DTS and SSIS packages can also co-exist on SQL Server 2008. However, to get the richer features for ETL processing, you need to migrate your packages to SSIS.

Note that support for developing, managing, and executing existing DTS packages will remain untouched if an existing instance of the SQL Server 2000 or SQL Server 2005 relational engine remains on the server after the upgrade. For example, if you install a new instance of the SQL Server 2008 relational engine or if you install other components of SQL Server 2008 without installing the new relational engine, then SQL Server Enterprise Manager will remain available as well as the DTS runtime and command-line tools. With these components still in place, you can still execute DTS packages manually or via SQL Server Agent jobs. For more information about this capability, see Chapter 13, "Integration Services."

However, if all instances of the SQL Server 2000 relational engine are removed or upgraded to SQL Server 2008, the ability to develop DTS packages will no longer exist. The SQL Server 2008 Setup application will install an updated DTS runtime and command-line tools, but no DTS design environment will be available. In this case, you can install the DTS Designer Components via a Web download as part of the [Microsoft SQL Server 2008 Feature Pack, August 2008](http://go.microsoft.com/fwlink/?LinkID=133088).

For more DTS-related upgrade information, see [Considerations for Upgrading Data Transformation Services](http://go.microsoft.com/fwlink/?LinkID=132615) in SQL Server 2008 Books Online.

### SQL Server Agent

SQL Server Agent has undergone several changes and enhancements since SQL Server 2000, providing stronger security, improved performance, and support for job steps that use SSAS and SSIS. Because SQL Server Agent is central to multiple SQL Server components, including maintenance plans, you need to review and understand the issues you might face when upgrading your use of SQL Server Agent.

#### Proxy Accounts

In SQL Server 2000, SQL Server Agent Transact-SQL jobs execute under the security context of the account that created the job; however, the sysadmin fixed server role can specify a different account for a Transact-SQL job’s execution context. Job steps in non-Transact-SQL jobs owned by sysadmin execute under the context of the SQL Server Agent service account. Job steps in non-Transact-SQL jobs owned by a non-sysadmin run under the context of a proxy account if it is enabled.

One limitation that SQL Server 2000 database administrators have is the inability to create more than one proxy account to use for SQL Server Agent jobs. SQL Server 2005 and SQL Server 2008 address this issue through multiple proxy accounts.

A proxy account defines the security context for a job step and gives SQL Server Agent access to security credentials for a Windows user. SQL Server Agent impersonates the credentials defined in the proxy account before it executes a job step. This lets SQL Server Agent execute the job step by using the security context of the proxy account.

After upgrading from SQL Server 2000 to SQL Server 2005, however, the user proxy account you defined on SQL Server 2000 before the upgrade will be upgraded to a proxy account called UpgradedProxyAccount. After the upgrade, this UpgradedProxyAccount proxy account is granted access only to those subsystems that were explicitly used in SQL Server 2000 and will not have access to all subsystems; the same restriction applies to non-sysadmin users who in SQL Server 2000 explicitly used the proxy account to execute their job step. You will need to manually grant access to any other subsystem that this proxy account needs. Table 2-1 lists the valid subsystems to which you can grant proxy access.

Table 2-1: Valid Subsystems to Which You Can Grant Proxy Access

| **Subsystem Name** | **Description** |
| --- | --- |
| Microsoft ActiveX Script | Run an ActiveX scripting job step. |
| Operating System (CmdExec) | Run an executable program. |
| Replication Distributor | Run a job step that activates the replication Distribution Agent. |
| Replication Merge | Run a job step that activates the replication Merge Agent. |
| Replication Queue Reader | Run a job step that activates the replication Queue Reader Agent. |
| Replication Snapshot | Run a job step that activates the replication Snapshot Agent. |
| Replication Transaction Log Reader | Run a job step that activates the replication Log Reader Agent. |
| Analysis Services Command | Run an SSAS command. |
| Analysis Services Query | Run an SSAS query. |
| SSIS package execution  PowerShell  Unassigned | Run an SSIS package.  Run a PowerShell script |

You can modify the proxy account by using Management Studio as follows:

1. In Object Explorer, expand a server, expand SQL Server Agent, and then expand Proxies.
2. Expand the subsystem node for the proxy, right-click the proxy you want to modify, and click Properties.

Alternatively, you can use the sp\_grant\_proxy\_to\_subsystem system stored procedure to grant access to disk subsystems to a proxy account, as this example shows:

USE msdb

GO

EXEC dbo.sp\_grant\_proxy\_to\_subsystem

@proxy\_name = 'UpgradedProxyAccount',

@subsystem\_name = N'SSIS'

GO

For more information about the SQL Server Agent proxy account, see [Creating SQL Server Agent Proxies](http://go.microsoft.com/fwlink/?LinkId=134177) in SQL Server 2008 Books Online.

#### SQL Server Agent Token Calls

In SQL Server 2008, the syntax for calling tokens in SQL Server Agent job steps has changed. The change replaces the use of square brackets with $() when calling out SQL Server Agent job step tokens—for example, [DATE] changes to $(DATE). This change does not require you to modify SQL Server Agent jobs after the upgrade; the change is performed behind the scenes by the upgrade process.

In addition, after the upgrade, the replacement for alert tokens is turned off by default, so you need to turn on this feature after ensuring that only trusted users have write permissions to the event log. You can turn on this feature by using the SQL Server Agent Properties dialog box and selecting the *Replace tokens for all job responses to alerts* check box under the Alert Subsystem tab.

For details about upgrading token references for SQL Server Agent jobs, see the [Using Tokens in Job Steps](http://go.microsoft.com/fwlink/?LinkId=134178) in SQL Server 2008 Books Online.

#### Upgrading Target Servers

Database administrators upgrading multi-server environments need to upgrade all target servers (TSX) before upgrading master servers (MSX).

You can re-enlist target servers by using Management Studio as follows:

1. In Object Explorer, connect to an instance of the Microsoft SQL Server Database Engine and expand that instance.
2. Right-click SQL Server Agent, point to Multi Server Administration, and then click Make this a Target. The Target Server Wizard guides you through the process of making a target server.

Keep in mind that SQL Server 2008 introduces two new features that make working in a distributed environment more cost-effective than using master and target servers as they were used previously:

* A method of managing servers that is called Policy-Based Management.
* Multi-server queries that use configuration servers and server groups.

For information about these features, see the following SQL Server 2008 Books Online topics:

* [Administering Servers by Using Policy-Based Management](http://go.microsoft.com/fwlink/?LinkID=133089)
* [Administering Multiple Servers Using Central Management Servers](http://go.microsoft.com/fwlink/?LinkId=134179)

#### Additional SQL Server Agent Issues When Upgrading from SQL Server 2000

If you are upgrading from SQL Server 2000, the following SQL Server Agent upgrade issues might affect the functionality of SQL Server Agent, so you need to address them after you have completed your upgrade:

* SQL Server Agent is available only for members of the sysadmin, SQLAgentUserRole, or MaintenanceUserRole roles.
* The SQL Server Agent service account no longer allows SQL Server authentication.
* After upgrading to SQL Server 2008, you must modify user scripts that use the xp\_sqlagent\_proxy\_account extended stored procedure to remove references to this extended stored procedure.
* SQL Server Agent jobs that perform log shipping will not be enabled after upgrading to SQL Server 2008. You must recreate the log shipping configuration by using SQL Server 2008's log shipping after the upgrade is finished. For details, see the log shipping section in Chapter 4, "High Availability."

For related information about upgrading SQL Server Agent from SQL Server 2000 to SQL Server 2005, see the [SQL Server 2005 Upgrade Technical Reference Guide](http://go.microsoft.com/fwlink/?LinkId=134180) white paper.

### Maintenance Plans

After upgrading from SQL Server 2000 to SQL Server 2008, you will find changes to the way you create maintenance plans. In pre-SQL Server 2005 releases, you use only the Maintenance Plan Wizard to create database maintenance plans that, in turn, result in multiple SQL Server Agent jobs that perform the defined maintenance tasks. The wizard in earlier versions of SQL Server, however, did not let you define all SQL Server-related maintenance tasks, including the important task of performing differential backups.

SQL Server 2005 and SQL Server 2008 have a more functional Maintenance Plan Wizard for creating basic maintenance plans. The wizard lets you create plans for almost any SQL Server-related maintenance task while taking advantage of the SSIS designer to extend the functionality of the maintenance plans. To gain the advantages of the newmaintenance plan functionality, you need to transfer your maintenance plans to SQL Server 2008. This transfer can be done automatically or manually.

For details about the Maintenance Plan Wizard, see [Maintenance Plan Wizard](http://go.microsoft.com/fwlink/?LinkId=134181) in SQL Server 2008 Books Online.

## Preparing to Upgrade

To make sure your upgrade process goes smoothly, consider the following changes in the way certain management tools work in SQL Server 2008 so that you can make appropriate modifications after your upgrade and maintain effective administration of your SQL Server environment.

### Deprecated Features

The following management tools features are deprecated in SQL Server 2008, meaning that they are still supported for backward compatibility only and will be removed from a future release of SQL Server. For a complete discussion of deprecated features, see [Deprecated SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133935) in SQL Server 2008 Books Online.

#### SQL-DMO

SQL Server 2008 replaces the SQL Distributed Management Objects (SQL-DMO) object library from earlier releases of SQL Server with SQL Management Objects (SMO). SQL-DMO is supported in SQL Server 2008 but only for backward compatibility. It has been upgraded to work with the latest release of SQL Server but will not support features beyond those available in SQL Server 2000. For more information, see [Developing SQL-DMO Applications](http://go.microsoft.com/fwlink/?LinkId=134182) in SQL Server 2008 Books Online.

#### osql Utility

SQL Server 2005 deprecated the ODBC-based osql command-line utility but without deprecating the functionality in any great way. For more information, see [osql Utility](http://go.microsoft.com/fwlink/?LinkId=134183) in SQL Server 2008 Books Online. If you intend to keep versions of SQL Server 2000 in your solution space, see [Backward Compatibility with SQL Server 2000 Tools](http://go.microsoft.com/fwlink/?LinkId=134184) in SQL Server 2008 Books Online.

Like the osql utility, the sqlcmdutility lets you enter Transact-SQL statements, system procedures, and script files at the command prompt, in a Windows script file, or in an operating system (Cmd.exe) job step of a SQL Server Agent job as well as in QueryEditor in SQLCMD mode. This utility uses OLE DB or SqlClient rather than ODBC as the data delivery mechanism. If you have not used this utility before, see [Tutorial: sqlcmd Utility](http://go.microsoft.com/fwlink/?LinkId=134185) in SQL Server 2008 Books Online.

Management Studio uses the Microsoft .NET Framework SqlClient for regular execution and SQLCMD mode in Query Editor. When sqlcmd is run from the command line, sqlcmd uses the OLE DB provider. Because different default options might apply, you could see different behavior when you execute the same query in Management Studio in SQLCMD Mode versus in the sqlcmd utility. For more information, see [sqlcmd Utility](http://go.microsoft.com/fwlink/?LinkId=134186) in SQL Server 2008 Books Online.

In addition to providing backward compatibility with SQL Server 2000 via osql and with SQL Server 2005 via continued support for SQLCMD, SQL Server 2008 can now host scripting solutions using Windows PowerShell to facilitate common administration and data access requirements. The main benefits of using PowerShell scripting are that you can use the same scripting language despite the server type you are communicating with and the language is object based. For more information, see [Scripting (Database Engine)](http://go.microsoft.com/fwlink/?LinkId=134187) in SQL Server 2008 Books Online.

#### bcp

The SQL Server 2000 version of the bulk copy program (bcp) lets a user with INSERT and SELECT permissions on a given table load data into that table by using the following command:

bcp <target\_table> IN <datafile> -c –T

This default format of the bcp command disables CHECK constraints and triggers on that table implicitly, which is not very secure because it raises a data manipulation language (DML) operation into a data definition language (DDL) operation invisibly. In addition to being a poor security practice, this could also lead to unwanted DDL triggers firing.

Therefore, in both SQL Server 2005 and in SQL Server 2008, the user must have the additional permission of ALTER for the target table to bulk-load data into it using the default bcp option of no checking; otherwise, the operation will fail.

To ensure that an attempt at the implicit escalation of trust does not occur, you can change the script to use a non-default transfer mode, which explicitly checks the constraints and fires the triggers, as follows:

bcp <target\_table> IN <datafile> -c –T –h “CHECK CONSTRAINTS, FIRE TRIGGERS”

This command checks the constraints and fires the triggers, but compared with SQL Server 2000, there will be no loss in performance of the data load due to the constraint checking and the consequent requirement to check the data using alternative methods.

For more information about this bcp issue, see [Controlling Constraint Checking by Bulk Import Operations](http://go.microsoft.com/fwlink/?LinkId=134188) in SQL Server 2008 Books Online.

#### isql Utility

The isql command-prompt utility is not available in SQL Server 2008. In its place, the new SQL Server release provides the SQLCMD utility. This means you need to remove all calls and references to isql or modify those calls to use the SQLCMD utility. If you do use isql to connect to SQL Server 2008, at most you can access features only available in SQL Server 7.0. For more information, see the "Legacy Command Prompt Tools" section in [Backward Compatibility with SQL Server 2000 Tools](http://go.microsoft.com/fwlink/?LinkId=134189) in SQL Server 2008 Books Online.

#### Rebuild.exe

SQL Server 2008 does not support Rebuild.exe. Database administrators should review administration scripts for use of the Rebuild.exe utility and modify those scripts to use the REBUILDDATABASES option of the Setup.exe utility. For more information, see [How to: Install SQL Server 2008 from the Command Prompt](http://go.microsoft.com/fwlink/?LinkID=132474) in SQL Server 2008 Books Online.

#### Setup.exe

The Setup.exe command-line interface has undergone a complete overhaul in SQL Server 2008. For instance, in SQL Server 2008, the Setup.exe utility does not support the TARGETCOMPUTER parameter for remote setup. Database administrators must remove this functionality from any administrative scripts they have. To set up SQL Server 2008 on a remote server using Setup.exe in command-line mode, you need to use a remote connection to run Setup.exe or set up SQL Server in user-interface mode. For more information, see [How to: Install SQL Server 2008 from the Command Prompt](http://go.microsoft.com/fwlink/?LinkID=132474) in SQL Server 2008 Books Online.

#### SQL Mail

SQL Server 2005 deprecated SQL Mail, but it is still shipped with SQL Server 2008. However, the preferred mail feature, Database Mail, is independent of an external API such as MAPI and instead uses the .NET Framework. Database Mail lets you send email messages from database applications, but it does not let you read e-mail messages; for this, you need a client. Although the SQL Mail Client will upgrade from SQL Server 2000 to SQL Server 2008, it has been tested only against Outlook 2000 and 2002.

Before beginning your upgrade, be sure to review scripts using SQL Mail to determine whether the scripts are sending attachments. The SQL Server 2008 version of SQL Mail will not send mail attachments if the mail client is connected using SQL Server Authentication; the authentication mode must be set to Windows Authentication. For information about how to convert a stored procedure to Database Mail, see [How to: Convert Stored Procedures from SQL Mail to Database Mail (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=134190) in SQL Server 2008 Books Online.

### Discontinued Functionality

SQL Server 2008 marks the end of the road for the following tools. If you use these tools, you need to make necessary revisions to your system before upgrading. You can find the latest information about removed features in [Discontinued SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133937) in SQL Server 2008 Books Online.

#### Web Assistant Stored Procedures

The Web Assistant stored procedures have been removed in SQL Server 2008. Use Reporting Services instead for building reports. For more information, see [Discontinued Database Engine Functionality in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133135).

#### English Query

As of SQL Server 2005, English Query—a set of tools for developing a natural-language interface to the database—has been discontinued and cannot be installed in or upgraded to SQL Server 2005 or SQL Server 2008. For in-place upgrades of SQL Server 2000 installations to SQL Server 2008, English Query will not be affected.

#### Meta Data Services

SQL Server 2000 Meta Data Services 3.0 is not available in SQL Server 2008, and database administrators upgrading to SQL Server 2008 will not be able to use the service to access repository tables. The repository tables will upgrade during the in-place upgrade process, but you should manually delete them from the msdb database after the upgrade process has completed.

You can execute the following script to obtain the drop statements for the repository tables and review the script’s output to remove user tables not associated with Meta Data Services:

USE msdb

GO

SELECT 'DROP TABLE dbo.' + TABLE\_NAME

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_SCHEMA = 'dbo'

AND TABLE\_NAME LIKE 'RTbl%'

GO

#### The Surface Area Configuration Tool

The Surface Area Configuration tool is discontinued for SQL Server 2008. Table 2-2 shows what you can use instead to configure settings, options, and component features.

Table 2-2: Tools for Configuring SQL Server 2008 Settings

| **Settings and Component Features** | **How to Configure** |
| --- | --- |
| Protocols, connection, and startup options | Use SQL Server Configuration Manager. |
| Database Engine features | Use Policy-Based Management, the property settings in Management Studio, or sp\_configure. |
| Analysis Services features | Use the property settings in Management Studio. |
| Reporting Services - EnableIntegrated Security property | Use the property settings in Management Studio. |
| Reporting Services - "Schedule events and report delivery" and "Web service and HTTP access" | Edit the RSReportServer.config configuration file. |
| Command-line options | No support in this release. |
| SOAP and Service Broker endpoints | Use [CREATE ENDPOINT](http://go.microsoft.com/fwlink/?LinkId=134191) and [ALTER ENDPOINT](http://go.microsoft.com/fwlink/?LinkId=134192). |

In addition, make sure that you review Chapter 5, "Database Security," because the SAC tool was designed to create a secure environment by default. For more information, see [Discontinued SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133937) in SQL Server 2008 Books Online.

**Note:** Only modify the various SQL Server service accounts by using the SQL Server Configuration Manager and not through Services in Control Panel, Administrative Tools. The SQL Server Configuration Manager will modify User Groups, permissions, and registry settings specific to SQL Server when accounts are added; Services will not.

### Breaking Changes

Breaking changes are those that might block an upgrade. The Microsoft SQL Server 2008 Upgrade Advisor, which we discuss in the “Upgrade Tools” section in a moment, helps detect major breaking changes by looking through the environment to be upgraded and checking to make sure that the relevant elements are present. For information about these types of changes, see [Breaking Changes to SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133938) in SQL Server 2008 Books Online.

### Behavior Changes

Behavior changes are non-breaking changes that might not cause your upgrade to fail but that might affect your applications after the upgrade. You can use Upgrade Advisor after the upgrade to help detect non-breaking changes. Upgrade Advisor checks the environment, looking at the objects that can be accessed, such as scripts, stored procedures, triggers, and trace files. Upgrade Advisor cannot analyze desktop applications or encrypted stored procedures, so you cannot assume that these are non-breaking changes.

The Upgrade Advisor output is in the form of an XML report that you view by using the Upgrade Advisor report viewer. The reports might contain an "other upgrade issues" item, which links to a list of issues that are not detected by Upgrade Advisor but that might exist on your server or in your applications. You should review the list of undetectable issues and determine whether you must change your server or applications because of the undetectable issues.

For more information about these types of changes, see [Behavior Changes to SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133944) in SQL Server 2008 Books Online.

### Upgrade Tools

Microsoft has released a Feature Pack for SQL Server 2008 that includes, among other useful extras, the SQL Server 2008 Upgrade Advisor Analysis Wizard. You can download the [August 2008 Microsoft SQL Server 2008 Feature Pack](http://go.microsoft.com/fwlink/?LinkID=131220) before you install SQL Server 2008. However, if you already have the SQL Server 2008 installation DVD, Upgrade Advisor is available as one of the options when you start to install the product. For details about using this useful tool, see Chapter 1, "Upgrade Planning and Deployment."

The Upgrade Advisor wizard checks more than 100 rules for possible upgrade issues, divided into the following categories:

* SQL Server
* Analysis Services
* Reporting Services
* Notification Services (only confirms lack of support)
* Data Transformation Services
* Integrations Services

After the checks are completed, a reporting interface lists the issues found and advises you about how to resolve them. For more information, see [Using Upgrade Advisor to Prepare for Upgrades](http://go.microsoft.com/fwlink/?LinkID=132624) in SQL Server 2008 Books Online.

### 64-bit Considerations

The client tools are developed for the 32-bit machine environment, and when running on a 64-bit machine, the software functions within Windows On Windows 64 (WOW64), an emulation environment that hosts 32-bit software within the 64-bit machine. Therefore, there should be no tools issues regarding upgrading to the 64-bit environment. For more information, see [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133946) in SQL Server 2008 Books Online.

### Known Issues and Workarounds

When you perform a side-by-side upgrade, you can import some settings from legacy management tools. As issues arise, Microsoft will document them and supply fixes or workarounds. The best place to find such information is in Microsoft Knowledge Base articles. For updated information regarding Knowledge Base articles, blogs, and other resources related to SQL Server management tools, see the link at the end of this chapter under "Additional References."

#### **Summary of Major Changes**

Table 2-3 summarizes the major changes to management tools from SQL Server 2000 to SQL Server 2008.

Table 2-3: Major Changes in Management and Development Tools from SQL Server 2000 to SQL Server 2008

| **SQL Server 2000 Feature** | **SQL Server 2008 Feature** | **Side-by-Side Upgrade** | **In-Place Upgrade** | **Notes about Old Tools** |
| --- | --- | --- | --- | --- |
| Enterprise Manager | SQL Server Management Studio | Both available | SSIS only | Deprecated |
| Analysis Manager | Business Intelligence Development Studio | Both available | Business Intelligence Development Studio only | Deprecated |
| ISQL/ OSQL | SQLCMD/ PowerShell | All available | Only osql and SQLCMD available | Osql deprecated  Isql discontinued |
| SQLMAIL | Database Mail | Both available | Both available | Deprecated and only 32-bit support |
| Surface Area Configuration | Declarative Management Framework (DMF) | Both available | DMF only | Discontinued (see separate table) |
| English Query | Not available | Available on old version | Not available | Discontinued |
| SQL Server Reporting Services (SSRS) 2000 | New version plus external Report Builder | Both available |  | Removed reliance on IIS; must use Report Builder |
| Data Transformation Services (DTS) | SQL Server Integration Services (SSIS) | Both available; scripting option | Both available; migration option | Support and migration available |
| Notification Services | Not available | Available on old server |  | Discontinued |
| SQL Server Agent | Still available |  |  | Support for multiple proxy accounts |
| Database Maintenance Plans (Agent jobs) | Database Maintenance Plans (SSIS packages) | Both available | Both available; plans to legacy components | Migrate to SSIS for more control over plans |
| Index Tuning Wizard (ITW) | Database Tuning Advisor (DTA) | Both available | Only DTA available | Use DTA; it has useful features such as time-limiting optimization |
| SQL Server Profiler | SQL Server Profiler |  |  | New features such as running Profiler from the Query Editor |
| SQL-DMO | SQL-SMO | Both available | Both available | SQL-DMO supported only for backward compatibility |
| BCP | BCP | Available | Available | Fully supported, but check for behavior changes |

## Upgrading from SQL Server 2000

When upgrading in-place from SQL Server 2000 to SQL Server 2008, SQL Server 2008 Setup will require that you upgrade all components: both the Database Engine and Analysis Services.

### Tools Replacement

As a general rule, with an in-place upgrade from SQL Server 2000 to SQL Server 2008, if SQL Server 2008 Setup is installed including the Common Components/Management Tools, it will replace the SQL Server 2000 Client Tools with SQL Server 2008 Management Tools.

* If there are multiple instances of SQL Server 2000 on the server and any of the remaining instances were also installed with the SQL Server Client Tools option, the SQL Server 2000 tools will remain.
* When the last SQL Server 2000 instance that had the Client Tools option is upgraded, the SQL Server 2000 tools will be removed.
* If a SQL Server 2000 instance was installed with the Minimal option, it will not prevent the SQL Server 2000 tools from being removed.
* SQL Server 2008 Setup will detect the Registered Servers of SQL Server 2005 Enterprise Manager and attempt to preserve them when upgrading to SQL Server 2008.

When upgrading to a new instance of SQL Server 2008 on the same server or a separate server, SQL Server 2008 will not remove the SQL Server 2005 Client Tools.

### Tools Connectivity

Generally speaking, upward connectivity is not supported. SQL Server 2000 Enterprise Manager, for example, cannot connect to SQL Server 2008 or SQL Server 2005. The one exception is SQL Server 2000 Query Analyzer, which uses ODBC to connect to SQL Server 2008; however, Query Analyzer cannot use new SQL Server 2008 data types or properly display query plans.

Therefore, when upgrading a SQL Server 2000 instance to SQL Server 2005, always include the Common Components (and therefore Management Tools), because the SQL Server 2000 tools cannot manage SQL Server 2008.

### SQL Server Agent Jobs

When upgrading in-place from SQL Server 2000 to SQL Server 2008, SQL Server Agent jobs will be upgraded along with the agent, but only to the best of the Setup program’s ability to do so.

**Note:** Because SQL Server 2008 uses a different mechanism for SSIS packages as opposed to DTS packages, any job that references a DTS package might run differently. For more information about converting DTS packages or running them in SQL Server 2008, see Chapter 13 "Integration Services."

Consider these best practices for SQL Server Agent jobs:

* In an in-place upgrade, recreate and test each SQL Server Agent job on the new SQL Server instance to eliminate issues that might arise due to changes in functionality.
* In a side-by-side upgrade, script out the SQL Server Agent job, transfer it to the SQL Server 2008 instance, and test it. Keep in mind that changes might need to be made to preserve the same functionality.
* External script files might also need to be moved to a new location. For more information about upgrading Transact-SQL scripts, see Chapter 8, "Transact-SQL Queries."

SQL Server 2000 maintenance plans will not be directly upgraded to SQL Server 2008 maintenance plans. Instead, the upgraded SQL Server 2000 plans can be viewed in the Legacy node of the Management folder in Management Studio 2008’s Object Explorer. These maintenance plans can be migrated to run under SQL Server 2008 by right-clicking over the legacy plan and selecting Migrate. For more information, see [How to: Migrate SQL Server 2000 Database Maintenance Plans](http://go.microsoft.com/fwlink/?LinkId=134193) in SQL Server 2008 Books Online.

Here are some considerations for migrating legacy maintenance plans:

* Maintenance plans will no longer support SQL Server 2000 log shipping. You must configure log shipping outside of the maintenance plan using SQL Server 2008 log shipping.
* Maintenance plans in a master server/target server distributed environment are not supported. Thus, you will need to create an individual maintenance plan for each target server as you would for a stand-alone server, instead of creating one plan and downloading it to the target servers.
* For maintenance plans to execute successfully, you must have SSIS installed on the same machine on which the SQL Server engine is installed.
* Unlike with SQL Server 2000, in SQL Server 2008, clean-up tasks do not clean up files in sub-directories of a chosen folder. The work-around is to have multiple clean-up tasks for each sub-folder in a folder or to have the backup task back up databases to only one folder.
* Maintenance plans will no longer attempt to repair minor problems currently configured under the Database Integrity Check task of the Maintenance Plan Wizard.
* To view maintenance plan tasks in SQL Server 2008, administrators must log in to SQL Server using Windows authentication with sysadmin privileges or as sa using SQL Server authentication.
* During the upgrade process, maintenance plan metadata will be migrated to SQL Server 2008 catalog views. You should modify any code that references the old maintenance plan system tables to reference the new catalog views.

## Upgrading from SQL Server 2005

When upgrading in-place from SQL Server 2005 to SQL Server 2008, SQL Server 2008 Setup will require that you upgrade all components: Database Engine, Analysis Services, Integration Services, and so on.

### Tools Replacement

SQL Server 2008 Setup will *not* replace the SQL Server 2005 tools with SQL Server 2008 tools.

* The SQL Server 2005 tools on the upgraded instance must be removed separately by using Add/Remove Programs. This applies whether there are one or many instances of SQL Server 2005 on the server.
* SQL Server 2008 Setup will detect the Registered Servers and other settings of the SQL Server 2005 tools and attempt to preserve them when upgrading to SQL Server 2008.
* Exporting and importing Management Studio registered servers between SQL Server 2005 and SQL Server 2008 is not supported because the underlying .regsrvr files do not have the same structure.

### Tools Connectivity

After an in-place upgrade, you can use only SQL Server 2008 tools and utilities to manage SQL Server 2008 instances.

SQL Server 2005 tools cannot be used to manage SQL Server 2008 instances. For example, if you try to use Management Studio 2005 against SQL Server 2008, you will see the message similar to that shown in Figure 2-1.



**Figure 2-1:** SQL Server Management Studio 2005 cannot connect to a SQL Server 2008 instance

However, the SQL Server 2008 tools can manage SQL Server 2005 instances. To manage SQL Server remotely, install the SQL Server 2008 tools on a workstation to ensure that you can manage SQL Server 2008 after it is installed.

On the server where you might be upgrading SQL Server in-place to SQL Server 2008, the SQL Server 2008 install process will not remove the existing SQL Server 2005 management tools. After the upgrade, you should check and, if necessary, uninstall the tools by using Add or Remove Programs.

For this reason, when moving to a new instance of SQL Server 2008 on the same server or a separate server, SQL Server 2008 will not remove the SQL Server 2005 Client Tools.

### Project Files

SQL Server Management Studio 2008 can open and work with project files built with Management Studio 2005. In addition, Management Studio 2005 can open and work with project files built in Management Studio 2008. However, because Management Studio 2005 cannot connect to SQL Server 2008, if Management Studio 2005 is used to open a Management Studio 2008 project and attempts to connect to a SQL Server 2008 instance, there will be a connection failure when attempting to execute a script.

BI Development Studio 2008 is built on a different version of Visual Studio than BI Development Studio 2005. When a BI Development Studio 2005 project is opened using BI Development Studio 2008, the project file will be converted to the new Visual Studio format. At that point, BI Development Studio 2005 will not be able to open the project. Opening BI Development Studio 2008 projects using BI Development Studio 2005 is not supported.

## Post-Upgrade Tasks

After the upgrade, you might need to perform some tasks to have your new management tools behave the same as before the upgrade.

After an in-place upgrade, SQL Server 2008 will preserve your settings in Management Studio. However, in a side-by-side upgrade, you will need to customize the tools to have the same functionality as before. These include the following features discussed earlier in this chapter:

* Exporting and importing registered servers from Management Studio.
* Transferring Management Studio 2005 and BI Development Studio 2005 projects to Management Studio 2008 and BI Development Studio 2008.
* Customizing the layout of Management Studio 2008.
* Making appropriate SQL Server settings using SQL Server 2008 Policy-Based Management.
* Recreating linked servers on the new instance.
* Using Database Mail instead of SQL Server 2000 SQL Mail.
* Recreating SQL Server Agent jobs and SQL Server 2000 or SQL Server 2005 maintenance plans.
* Transferring external Transact-SQL scripts to the correct location for the new instance.

There are other resources available to help get you familiarized with the new environment post upgrade. These will be especially useful if you are upgrading from a pre-SQL Server 2005 installation because there have been such dramatic changes in SQL Server’s management interface. An example of a resource that is vital no matter which environment you are coming to SQL Server 2008 from is the virtual lab [TechNet Virtual Lab: SQL Server 2008 Policy-Based Management](http://go.microsoft.com/fwlink/?LinkId=134194). You can also learn more about new Policy-Based Management features in Chapter 5, "Database Security."

The [Microsoft Events Web site](http://go.microsoft.com/fwlink/?LinkId=134195) lists many more resources to help you get started with the SQL Server 2008 management environment after the upgrade; just search for SQL Server 2008.

## Conclusion

SQL Server 2008 provides many valuable new features and enhanced functionality in the management tools area. You can minimize the risk of an upgrade to SQL Server 2008 and gain confidence by following these practices and considerations and understanding the management tools changes you need to consider to prepare for an effective upgrade.

## Additional References

For an up-to-date collection of additional references for upgrading SQL Server management and Development tools, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkID=132510).

For the latest updates to SQL Server 2008 Books Online, see the [Microsoft SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkID=132511).

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkID=132515)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=133003)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkID=132515)

# Relational Databases

## Introduction

Customers currently using SQL Server 2000 or SQL Server 2005 have several options for upgrading their relational databases to SQL Server 2008. The best approach depends on how the SQL Server 2000 or SQL Server 2005 instances and databases are deployed, the level of database availability required during the upgrade, and how much upgrade testing you have to do.

## Relational Database Configurations

SQL Server 2000 and SQL Server 2005 can be deployed in different configurations. For example, a SQL Server might have a single default instance or multiple named instances. In high availability (HA) environments, a given configuration might participate in a cluster, a log shipping scenario, or perhaps a database mirroring relationship. Other configuration options could include replication or full-text indexing. This chapter discusses upgrading relational databases in any of these configurations. However, there are special considerations when you upgrade relational databases that are part of a cluster, log shipped, mirrored, replicated, or enabled for full-text indexes. For specific information about how to upgrade these configurations, see the following resources:

* [Considerations for Upgrading the Database Engine](http://go.microsoft.com/fwlink/?LinkID=134156) in SQL Server 2008 Books Online provides a great overview of the upgrade process and covers many high-level details to consider before you start the upgrade process.
* For HA features such as clustering, log shipping, mirroring, and replication, see Chapter 4, "High Availability." You can find additional specific information about each of these technologies in the following resources.
  + For cluster upgrade information, see [Upgrading a SQL Server 2008 Failover Cluster](http://go.microsoft.com/fwlink/?LinkID=132180) in SQL Server 2008 Books Online.
  + For log shipping upgrade information, see the following SQL Server 2008 Books Online topics:
    - [Migrating a SQL Server 2000 Log Shipping Configuration to SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132489)
    - [Upgrading SQL Server 2005 Log Shipping to SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132491)
  + For mirroring, see [How to: Minimize Downtime for Mirrored Databases When Upgrading Server Instances](http://go.microsoft.com/fwlink/?LinkID=132643) in SQL Server 2008 Books Online.
  + For replication upgrade information, see [Considerations for Upgrading Replicated Databases](http://go.microsoft.com/fwlink/?LinkID=132650) SQL Server 2008 Books Online.
* For full-text upgrade information, see Chapter 6, "Full-Text Search," and [Full-Text Search Upgrade](http://go.microsoft.com/fwlink/?LinkId=134495) SQL Server 2008 Books Online.
* For MSDE and SQL Server Express upgrades, see Chapter 10, "SQL Server Express."

## Upgrade Considerations

Before upgrading from SQL Server 2000 or SQL Server 2005 to SQL Server 2008, you have to address the following considerations.

Make sure that you have a valid backup of all the databases participating in the upgrade process. For backup details, see Chapter 1, "Upgrade Planning and Deployment."

Only instances of SQL Server 2000 Service Pack 4 (SP4) or later versions, instances of SQL Server 2005 RTM or later versions on Windows Server 2003, and SQL Server 2005 SP2 or later versions on Windows 2008 can be upgraded to SQL Server 2008. For comprehensive information about which versions and editions can be upgraded, see [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) in SQL Server 2008 Books Online.

To minimize potential problems when a database is upgraded to SQL Server 2008, the database keeps its existing compatibility level (except for system databases, which have a 100 compatibility level). If you upgrade a database from SQL Server 2000, it will have an initial compatibility level of 80, and if you upgrade from SQL Server 2005, the database will have an initial compatibility level of 90. Before you change the compatibility level of an upgraded database to 100, you must assess how the change might affect your applications. For specific compatibility-level guidance, see [ALTER DATABASE Compatibility Level](http://go.microsoft.com/fwlink/?LinkID=133122) in SQL Server 2008 Books Online.

On a server that contains multiple instances of the SQL Server Database Engine (SQL Server 2000 or SQL Server 2005), you must upgrade each instance individually; upgrading one instance has no affect on other instances on the same server. From the SQL Server 2008 point of view, you can upgrade instances in any order and at any time.

You cannot perform a direct, in-place upgrade from a 32-bit edition of SQL Server 2000 or SQL Server 2005 to any 64-bit edition of SQL Server 2008. However, databases from a 32-bit edition can be restored on or attached to a 64-bit edition of SQL Server 2008, and they will be automatically upgraded. For more information about this process, see Chapter 1, "Upgrade Planning and Deployment."

**Note**: Be aware of significant tool upgrade issues in upgrading to SQL Server 2008. For comprehensive information about how to upgrade SQL Server tools, see Chapter 2, "Management and Development Tools."

SQL Server 2008 has multiple hardware and software requirements that you must consider when you perform an in-place upgrade because you might have to update your operating system (OS) service pack or install or upgrade other operating system components. For a complete list of requirements, see [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132172) in SQL Server 2008 Books Online.

### Full-Text Search

In SQL Server 2008, full-text search functionality is integrated into the Database Engine and is improved in several ways. Improvements include the following:

* Stop lists
* Thesaurus improvements
* New troubleshooting tools
* New word breaker family
* Performance improvements in indexing and in querying and query parallelism

When you perform an upgrade from SQL Server 2000 or SQL Server 2005 to SQL Server 2008, you have three options for how to upgrade full-text indexes:

**Import.** This option imports full-text catalogs and brings them online and ready to search fairly quickly. The downside to this option is that it does not take advantage of the new and improved word breakers. If you later decide to rebuild your full-text indexes, SQL Server will use the new word breakers.

**Rebuild.** This option enables all new full-text features of SQL Server 2008, including the new and improved word breakers. However, this option will likely take longer and consume more CPU and memory resources.

**Reset.** If you select the Reset option, only the metadata of the full-text indexes remains. If you want to use the full-text indexes after the upgrade and you select this option, you have to manually issue a full population.

For more information about SQL Server 2008’s full-text search improvements, see Chapter 6, "Full-Text Search."

### What Can Be Upgraded?

#### Versions

You can upgrade SQL Server 2000 SP4; SQL Server 2005 RTM, SP1, and SP2; and SQL Server 2008 Community Technology Preview (CTP) 5 and later to SQL Server 2008 RTM. For more information about versions that you can upgrade, see the "Upgrade Considerations" section later in this chapter.

#### Components

You can upgrade the Database Engine component, including SQL Server Agent, to SQL Server 2008. You can also upgrade the Management Tools, Full-Text Search, Analysis Services, and Reporting Services components. For more information about each of these topics, see the following chapters in this guide:

* Chapter 2, "Management and Development Tools"
* Chapter 6, "Full-Text Search"
* Chapter 11, "Analysis Services"
* Chapter 14, "Reporting Services"

#### Editions

There are multiple in-place upgrade paths available to you, depending on the edition of SQL Server that you are upgrading from. Generally, you can upgrade to an edition equal or later to your current edition. For example, you can upgrade from SQL Server 2005 Standard to SQL Server 2008 Standard or Enterprise.

However, you might have to perform multiple in-place upgrades to reach the edition you require. For example, to upgrade from SQL Server 2005 Express, to SQL Server 2008 Standard, you would first have to upgrade to SQL Server 2008 Express and then from SQL Server 2008 Express to SQL Server 2008 Standard. In this case, it might be a better idea to perform a side-by-side upgrade, which would let you upgrade compatible features directly from one version of SQL Server to the next.

When you upgrade editions in-place, you must maintain the same processor architecture type. There is no support for upgrading from a 32-bit edition to a 64-bit edition or from X64 to IA-64 or vice versa. If you have to upgrade to a different processor architecture, you must perform a side-by-side upgrade. For a complete upgrade path list, see [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) in SQL Server 2008 Books Online.

Note also that SQL Server 2008 is not supported on Windows 2000 and earlier operating systems. For a complete list of supported operating systems, see [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132172) in SQL Server 2008 Books Online.

#### Cross-Language Support

There is broad support for localization within SQL Server 2008. The English version of SQL Server 2008 is supported on all localized versions of supported operating systems. If you are running a localized version of SQL Server 2000 or SQL Server 2005, you can perform an in-place upgrade to SQL Server 2008 as long as the target instance and operating system are of the same language as your original SQL Server 2000 or SQL Server 2005 instance. You can run a localized instance of SQL Server 2008 on English-language versions of supported operating systems by using the Multilingual User Interface Pack (MUI) and verifying that the following operating system settings match the language of SQL Server to be installed:

* The operating system user interface setting
* The operating system user locale setting
* The system locale setting

For more information about cross-language support, see [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) in SQL Server 2008 Books Online.

### What Cannot Be Upgraded?

#### Versions

Although SQL Server 7.0 and earlier databases are not directly upgradeable to SQL Server 2008, there are options available to upgrade these databases indirectly. For more information, see [Copying Databases from SQL Server 7.0 or Earlier](http://go.microsoft.com/fwlink/?LinkId=134503) in SQL Server 2008 Books Online.

Instances of SQL Server 2000 with service pack levels less than SP4 cannot be upgraded. However, individual databases can be upgraded by using the side-by-side upgrade model even if SQL Server 2000 is only at the RTM level.

#### Components

Some components have either been deprecated or removed in SQL Server 2008. For example, Notification Services was removed from SQL Server 2008. For information about how to run Notification Services with SQL Server 2008, see Chapter 9, "Notification Services." In addition, Data Transformation Services (DTS) was replaced by SQL Server Integration Services (SSIS). For more information about DTS and SSIS, see Chapter 13, "Integration Services."

#### Editions

Although Microsoft has made a great effort to support as many in-place upgrade paths as possible, there remain some editions of SQL Server 2000 and SQL Server 2005 that do not support in-place upgrades. These include the following:

* SQL Server 2000 (32-bit) Developer SP4
* SQL Server 2000 (32-bit) Personal SP4
* SQL Server 2000 Enterprise Evaluation (32-bit, IA-64, X64)
* SQL Server 2005 (32-bit) Workgroup
* SQL Server 2005 (32-bit) Developer
* SQL Server 2005 Enterprise Evaluation (32-bit, IA-64, X64)
* SQL Server 2005 IA-64 (64-bit) Developer

For more information, see [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) in SQL Server 2008 Books Online.

#### Platforms

When you perform an in-place upgrade, you must upgrade to the same processor architecture. For example, you cannot upgrade from a 32-bit edition of SQL Server 2005 to a 64-bit edition by using the in-place upgrade method. However, you can upgrade to a different processor architecture by using the side-by-side upgrade method.

#### Cross-Language Support

Upgrading across localized versions of SQL Server is not supported. For more information, see [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) in SQL Server 2008 Books Online.

#### Additional Limitations on Upgrades

Be aware that when you perform an in-place upgrade, you cannot add features or make configuration changes. If you have to do either, you must do so after the upgrade. In addition, when you perform an in-place upgrade, you must upgrade the whole instance. You cannot upgrade one database in an instance and leave the rest of the instance at the previous level. To upgrade a single database, for example, you would have to perform a side-by-side upgrade. This would let you copy or restore a database from your old instance to your new instance while leaving the old instance in place.

## In-Place Upgrade vs. Side-by-Side Upgrade

When you upgrade a SQL Server 2000 or SQL Server 2005 database to SQL Server 2008, you have two main options: You can perform an in-place upgrade or a side-by-side upgrade.

For information related to planning and deploying an in-place or side-by-side upgrade, see Chapter 1, "Upgrade Planning and Deployment."

### In-Place Upgrade

An in-place upgrade is the fastest and easiest upgrade method because it upgrades all system and user databases and settings for you. In addition, you do not have to update client applications to connect them to a new instance of the relational Database Engine. However, an in-place upgrade is an all-or-nothing approach. In the unlikely event that an in-place upgrade of the relational Database Engine fails, you cannot quickly roll back to SQL Server 2000 or SQL Server 2005.

If the upgrade fails, you have to take the following steps:

1. From the installation media, run the repair option in an attempt to fix the instance; if this does not work, go to step 2.
2. Uninstall the corrupted SQL Server 2008 instance that was created during the failed upgrade attempt.
3. Restart.
4. Reinstall the earlier version of SQL Server (SQL Server 2000 or SQL Server 2005).
5. Reinstall any required SQL Server service packs to your SQL Server 2000 or SQL Server 2005 instance.
6. Restore the system and user databases from database backups.
7. Review issues that prevented a successful upgrade in the previous attempt, resolve them, and restart the upgrade process.

Be aware that downtime if there are upgrade problems can be significant.

### Side-by-Side Upgrade

With a side-by-side upgrade, the SQL Server 2008 relational Database Engine is installed as a second instance, and the original SQL Server 2000 or SQL Server 2005 relational Database Engine remains installed. Then you move or copy one or more SQL Server 2000 or SQL Server 2005 user databases to the SQL Server 2008 instance (each moved database is automatically upgraded).

During the side-by-side upgrade process, users can continue to access the SQL Server 2000 or SQL Server 2005 relational Database Engine and its databases (which are unaffected by the upgrade process) while the new SQL Server 2008 server is being built. When you are ready to switch to the new instance, users must stop activity on the older instance while you transfer databases to the new SQL Server 2008 server. After the side-by-side upgrade is complete, the SQL Server 2008 relational Database Engine and the SQL Server 2000 or SQL Server 2005 relational Database Engines co-exist. After you verify the SQL Server 2008 relational Database Engine, you can let applications access the new server. And after the SQL Server 2008 relational Database Engine is in production, you can uninstall SQL Server 2000 or SQL Server 2005 from the old server.

A side-by-side upgrade can require much more effort than an in-place upgrade. In an in-place upgrade, SQL Server 2008 Setup makes sure that the new SQL Server 2008 instance has the same name as the old instance and automatically preserves server configuration and server objects, such as logins, SQL Server Agent jobs, and so on. In a side-by-side upgrade, you must perform all those tasks yourself, either manually or through scripting techniques.

With a side-by-side upgrade, you either install the SQL Server 2008 relational Database Engine as a new named instance on the same server or on a new server as either the default instance or a named instance.

**Licensing Note:** If you install SQL Server 2008 on a separate computer or as a separate instance on the same computer as part of the upgrade to SQL Server 2008, you have 60 days to complete the upgrade from the earlier version (by removing the earlier version) before you are out of licensing compliance.

**Warning**: A relational database that was upgraded to SQL Server 2008 cannot be moved back to an earlier version of SQL Server. You could extract the data out of the SQL Server 2008 instance. However, if the upgrade fails because of a persistent issue such as disk corruption, you must have a verified backup of your original databases to retrieve data from.

A side-by-side upgrade lets you continue to use the existing relational database environment until you are ready to switch to the new one. This approach can help maximize your ability to quickly roll back to the prior instance should any difficulties arise. A side-by-side upgrade might also result in simpler testing scenarios because both versions are available at the same time.

However, a side-by-side upgrade is not as fast or simple as an in-place upgrade because of the additional effort required to transfer all server objects and redirect clients to the new instance. The side-by-side upgrade method does not upgrade any system databases. Although you can use the Copy Database Wizard to help move some system objects, most database objects in the system databases—such as server logins, jobs, alerts, maintenance plans, user-defined error messages, and DTS packages—must generally be moved separately or recreated manually.

**Important**: If you continue to use the instance of SQL Server 2000 or SQL Server 2005 while you are testing an upgraded database in an instance of SQL Server 2008, the databases will not remain synchronized. Data changes that are made to the existing database will not be made to the upgraded database. To bring the instance of SQL Server 2008 forward in time, you must bring the data from SQL Server 2000 or SQL Server 2005 forward by using a kind of data transfer, such as data file detach and attach, transaction log backup and restore, or transactional replication. Some of these topics are covered later in this chapter.

**Note**: When you install a new instance of the SQL Server 2008 relational Database Engine and then move one or more user databases to this instance, be aware that the following SQL Server 2000 and SQL Server 2005 relational database features are disabled by default in new installations:

* Ad hoc distributed queries
* Automation
* SQL Mail
* Web Assistant stored procedures
* Named pipes
* xp\_cmdshell

To enable some or all of these features, use the SQL Server Configuration Manager utility.

If you opt to use the side-by-side method of upgrading to the SQL Server 2008 relational Database Engine, you have several methods to decide from, including the following:

* Backup and restore
* Detach and attach
* Manual schema rebuild and data export/import
* Log shipping
* Copy Database Wizard

Look at each of these methods in turn.

#### Backup and Restore

You can upgrade a SQL Server 2000 or SQL Server 2005 relational database by performing a database backup of that database and then restoring it to a SQL Server 2008 instance. You can create a database backup by using SQL Server Enterprise Manager in SQL Server 2000 or SQL Server Management Studio (SSMS) in SQL Server 2005. You can also create a backup in either version by executing a Transact-SQL script. You can then restore this database backup on the SQL Server 2008 instance by using SSMS or a Transact-SQL script. The database will automatically be upgraded as it is restored.

This side-by-side upgrade method lets you leave the SQL Server 2000 or SQL Server 2005 relational database online and available to users until you are ready to switch to the upgraded database in the new instance of SQL Server. Before switching over, make sure that you perform post-upgrade compatibility and functionality testing (described later in this chapter) and any necessary application modifications such as connection string changes, modifications required by the Database Engine upgrade, and Transact-SQL changes.

The advantage of using the backup/restore method is that database backups are usually smaller than the original database files because the database backup process captures only actual data, not reserved but unused database space. In addition, only the tail of the transaction log is backed up instead of the whole log file. This decrease in file size usually makes any file transfer over the network faster than trying to transfer the original data and log files. You can also use third-party utilities to compress the backup before transferring it over the network.

One drawback to the backup/restore method is that if the backup is made to disk instead of to tape, the destination SQL Server will need additional disk space to accommodate the backup file in addition to the original database files.

**Important**: As noted in the main text, if you continue to use the SQL Server 2000 or SQL Server 2005 instance while you are testing an upgraded database in a SQL Server 2008 instance, the databases will not remain synchronized. Data changes that you make to the existing database will not be made to the upgraded database. You must perform another backup/restore of the database when you are ready to switch to SQL Server 2008.

To minimize downtime, you might consider putting the database into Full recovery mode before you make the second backup and restore. You can then restore the database on your SQL Server 2008 instance, specifying the NORECOVERY option. Because you are in the Full recovery mode, users can continue to change the original database while all the backing up and restoring is occurring. When you are ready for the final cutover, back up the transaction log on the original SQL Server 2000 or SQL Server 2005 instance and restore the transaction log on the SQL Server 2008 instance, specifying the RECOVERY option.

#### Detach and Attach

You can also upgrade a SQL Server 2000 or SQL Server 2005 database by detaching the database from its current instance, moving or copying the underlying data and log files, and then reattaching those data and log files to a SQL Server 2008 relational database instance. The database will automatically be upgraded as it is attached. If you copy the data and log files, you can reattach the original data and log files to the existing instance of SQL Server with only minimal disruption to the availability of the databases to be moved. The detach/attach upgrade method has the safety advantage in that the current databases remain available until you are ready to switch over after you perform post-upgrade compatibility and functionality testing and any necessary application modifications (such as connection string changes, modifications required by the Database Engine upgrade, and Transact-SQL changes).

**Note:** In many SQL Server databases, significant space is reserved in the data and transaction log files that has no data. This is by design to minimize the frequency with which a data file must expand as data is added. However, if you plan to detach a SQL Server database of any version and copy or move it to another location to be reattached, you will be moving this empty space together with your data. This increases the time that is required to complete the upgrade process.

**Important**: As with the backup/restore method, if you continue to use the SQL Server 2000 or SQL Server 2005 instance while you are testing a detach/attach upgraded database in a SQL Server 2008 instance, the databases will not remain synchronized. Data changes that you make to the existing database will not be made to the upgraded database. You must perform another detach/attach upgrade of the database when you are ready to switch to SQL Server 2008. As with all upgrade methods, make sure that you have reliable backup copies of the databases that you are upgrading. In the unlikely event that the data files become corrupted during this process, you will be able to restore from these backups.

When you move a very large database (VLDB) from one instance to another on the same server, detach/attach can have the advantage of requiring less disk space than some other upgrade methods if you reuse underlying data and log files instead of copying them. On systems that use a SAN disk configuration, you can detach the SAN volume from the older instance of SQL Server and then present it to the instance of SQL Server 2008. These options will save disk space and might save database administrators from having to move the database files over the network. But they will also eliminate the ability to roll back if the relational database upgrade should fail for any reason.

With a SAN disk configuration, you can also clone the disk volume while the original SQL Server relational database is online and then re-create that clone on another disk array, which you can then attach to the SQL Server 2008 relational database instance for upgrade. Database administrators with a SAN disk configuration should meet with their disk engineers to discuss possible methods for moving the database files without having to perform a copy over the network and without attaching the original files if it is possible.

**Caution:** For rollback purposes, you should create a copy of the relational database file (or perform a backup) before attaching it to a new relational database instance. After you have attached a relational database file to SQL Server 2008, you cannot reattach it to an earlier version of the SQL Server relational Database Engine.

#### Manual Schema Rebuild and Data Export/Import

Another way that you can perform a side-by-side upgrade of a SQL Server 2000 or SQL Server 2005 database is by using SSMS to generate a database creation script for the database and executing the script in the desired SQL Server 2008 instance. You can then manually copy the data from the original relational database to the new relational database by using Transact-SQL scripts, DTS packages, SSIS packages, BCP commands, or any number of other methods that are available to SQL Server database administrators for copying data from one database to another.

Most database administrators do not choose this method for upgrading their relational databases because it is primarily a manual process and provides few advantages over the side-by-side upgrade methods previously discussed. However, it does leave the current relational database online and lets you schedule the upgrade at a convenient time (such as overnight or over a weekend). This upgrade method also enables you to modify database schema, clean up database data, or filter data being moved to the upgraded databases during the upgrade process.

#### Log Shipping

You can use log shipping to make a side-by-side relational database upgrade easier with minimal downtime. You can log ship from a legacy database on one instance to a target database on SQL Server 2008 and then fail over the log shipping as the final upgrade step. For more information about this alternative, see the "Methods for New Hardware and Side-by-Side Upgrades" section in Chapter 4, "High Availability."

#### Copy Database Wizard

You can also upgrade a SQL Server 2000 or SQL Server 2005 database by using the Copy Database Wizard. This wizard supports two upgrade methods: detach and attach and SQL Server Management Objects (SMO).

* **Detach and attach.** This method is identical to the detach/attach upgrade method previously discussed. It lets you move or copy the underlying relational database files after they are detached and automatically reattaching the detached files after a copy (and optionally after a failed move).
* **SMO.** This method is similar in concept to the manual schema rebuild and data export/import upgrade method we looked at earlier. This method uses SMO to read the definition of each database object in the relational database being upgraded, without taking it offline, and then recreates each object in the destination database. Then it creates and executes an SSIS package to transfer the data from the source table to the newly created destination table, recreating indexes and metadata.

Each of these methods lets you automate and schedule the upgrade process at a convenient time. Each method within the Copy Database Wizard also lets you select one or more of the following additional object types to upgrade:

* Logins
* User stored procedures in the master database
* SQL Server Agent jobs
* User-defined error messages

With the Copy Database Wizard, you cannot copy extended stored procedures, alerts, DTS packages, or linked server configurations. You must move these manually.

## Evaluating Potential Upgrade Issues

Regardless of whether you choose an in-place upgrade or a side-by-side upgrade of a relational database, there are a range potential issues that you might face during the upgrade. You can obtain a report that identifies many of these potential issues before you start an upgrade by running the [Microsoft SQL Server 2008 Upgrade Advisor](http://go.microsoft.com/fwlink/?LinkID=133088). You should run Upgrade Advisor to analyze all the SQL Server 2000 or SQL Server 2005 databases that you want to upgrade. For information about how to install and running this tool, see Chapter 1, "Upgrade Planning and Deployment." However, there is a category of issues that Upgrade Advisor cannot detect or whose detection would result in too many false-positive results.

Look at the most important SQL Server 2008 upgrade issues—including deprecated and discontinued functionality, changes that might prevent an upgrade, and changes in feature behavior that might require modifications—whether detected by Upgrade Advisor or not. For a complete list of backward-compatibility issues, see [Backward Compatibility](http://go.microsoft.com/fwlink/?LinkID=132171) in SQL Server 2008 Books Online. For a complete list of the Database Engine upgrade issues that Upgrade Advisor detects, see the "Database Engine Upgrade Issues" topic in the SQL Server 2008 Upgrade Advisor Help file.

### Deprecated Features

There are several features in SQL Server 2008 that are marked for removal in the next version of SQL Server. After you upgrade, you should remove the usage of these features from existing applications and avoid them in new development work. For a complete discussion of deprecated features in SQL Server 2008, see Chapter 8, "Transact-SQL Queries," and [Deprecated Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133092) in SQL Server 2008 Books Online.

### Discontinued Functionality

Several features from earlier versions of the SQL Server Database Engine are not supported in SQL Server 2008, so you must use replacement features for these. Table 3-1 lists the discontinued features that you will most likely encounter as well as the recommended replacement features.

**Important**: Backward compatibility with earlier versions of SQL Server was a high priority in SQL Server 2008, so in most cases, applications will behave as in the past. For more information, see [Discontinued Database Engine Functionality in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133135) in SQL Server 2008 Books Online.

Table 3-1: Discontinued Features and Replacements

| **Discontinued Feature/Functionality** | **Replacement Feature/Corrective Action** |
| --- | --- |
| **sp\_configure with the ‘allow updates’ option.** Used to directly update system tables, but direct updates to system tables are no longer supported. This option, although present, has no effect. | Modify scripts that update system tables directly to use documented commands instead of direct updates. |
| **Network Protocols.** The NWLink IPX/SPX, AppleTalk, Banyan Vines, and Multiprotocol network protocols are no longer supported. | Use TCP/IP sockets, named pipes, VIA, or shared memory. |
| **Rebuildm.exe.** Used for rebuilding system databases, this executable is obsolete. | This executable is replaced by the REBUILDDATABASE option in setup.exe. Modify any scripts that use the rebuildm.exe executable. |
| **Sample Databases.** The Northwind and Pubs sample databases were discontinued. | These sample databases are replaced by the AdventureWorks and the AdventureWorksDW sample databases. However, you can download or move Northwind and Pubs from a SQL Server 2000 or SQL Server 2005 instance. |
| **Remote setup.** Used to install SQL Server on a remote computer, this option is no longer available in the Setup program. | Use a remote connection to run the SQL Server Setup program on the remote computer. |
| **Named pipe backup devices.** No longer supported. | Contact your backup vendor to see whether it has a new version that supports VDI (the replacement for named pipes). Alternatively, you can use SQL Server native tools to back up to disk or tape. |
| **Mail attachments by using SQL Mail.** No longer supported. | Use Database Mail to send mail attachments. |

For a complete list of discontinued functionality in SQL Server 2005 and SQL Server 2008, see the following topics in SQL Server 2008 Books Online:

* [Discontinued Database Engine Functionality in SQL Server 2005](http://go.microsoft.com/fwlink/?LinkID=133134)
* [Discontinued Database Engine Functionality in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133135)

### Breaking Changes

Some settings will prevent the SQL Server 2008 Setup program from starting the upgrade process for the Database Engine. If one of these issues is encountered, the upgrade process will stop, and the legacy system will remain in place. Upgrade Advisor will discover each issue that Table 3-2 lists.

Table 3-2: Issues That Will Prevent an Upgrade

| **Issue** | **Corrective Action** |
| --- | --- |
| **Username of *sys* in a database.** SQL Server 2008 does not permit a username of *sys* in a database. | Create a new user who has a different name, transfer ownership of all database objects to that new user, and drop user *sys* from the database. |
| **Duplicate login SIDs.** SQL Server 2008 does not permit duplicate login SIDs for SQL Server authentication. | Drop and re-create the duplicate SQL Server logins on the legacy system. |
| **Login names matching fixed server role names.** SQL Server 2008 does not allow login names to match fixed server role names. | Rename the logins on the legacy system. |
| **Database ID 32767.** SQL Server 2008 does not permit a database ID of 32767. | Detach and reattach the database and make sure that it gets a new database ID. |
| **Duplicate index names.** SQL Server 2008 does not permit duplicate index names on a table. | Rename the indexes so that all indexes are unique within each table. |

If you are performing a side-by-side upgrade, you must correct only the "username *sys*" and the "duplicate index names" issues on the legacy system. SQL Server 2008 will prevent you from applying any of the other settings, such as using a database ID of 32767, having duplicate login SIDs, and having login names that match fixed server role names.

Some additional issues will prevent you from upgrading a SQL Server 2000 or SQL Server 2005 database to SQL Server 2008. You must resolve these issues before you start an upgrade, or the upgrade will fail. Table 3-3 lists the most likely issues of this kind together with the recommended corrective action.

Table 3-3: Issues to Resolve Before You Start the Upgrade Process

| **Issue** | **Corrective Action** |
| --- | --- |
| **Compressed drives.** SQL Server 2008 cannot create or upgrade relational databases residing on compressed drives. | Verify that the relational databases to be upgraded do not reside or will not reside on compressed drives. READ-ONLY relational databases and files can be placed back on compressed drives after the upgrade is complete. |
| **Read-only file groups.** SQL Server will not upgrade file groups in relational databases set to READ\_ONLY because non-writeable files will not be upgraded. | Make sure that all file groups in relational databases scheduled for upgrade are set to READ\_WRITE. |
| **Disk space.** Additional space is required for data files during an upgrade because of additional system metadata, 40 bytes per column required for large object columns, and the storage of a full-text mapping table for each full-text indexed table within the data file instead of in the file system. | During setup, make sure that each user database is set to autogrow and that the PRIMARY file group of each user database has sufficient disk space. Insufficient disk space will cause an upgrade to fail. |
| **Named pipe naming.** During the upgrade, the Setup program starts the SQL Server 2008 relational Database Engine with shared memory support, a named pipe that accepts only local connections. If the pipe name specified on the server is not blank, it must begin with "\\.\pipe\" to be valid. | If the pipe name is not valid, change it to a valid name. |
| **Service Account.** The SQL Server service must not be running under the Local Service or the Network Service account if SQL Server is running on a Windows Server 2003 domain controller. | Change the service account to Local System, a local user account, or a domain user account. |

For a complete list of breaking changes in SQL Server 2005 and SQL Server 2008, see the following SQL Server Books Online topics.

* [Breaking Changes to Database Engine Features in SQL Server 2005](http://go.microsoft.com/fwlink/?LinkID=133140)
* [Breaking Changes to Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133141)

### Behavior Changes

SQL Server 2008 has several behavior changes that might require you to take corrective action after the upgrade is complete. Table 3-4 lists the behavior changes that you are most likely to see in addition to the recommended corrective action.

**Important**: Backward compatibility with earlier versions was a high priority in SQL Server 2008, so in most cases, applications will behave as in the past.

Table 3-4: Behavioral Changes That Might Require Corrective Action

|  |  |
| --- | --- |
| **Behavior Change** | **Corrective Action** |
| **Log file disk space.** Additional space is required by transaction log files. | Make sure that the log files for each user database are set to autogrow and have sufficient additional disk space or increase the size of each log file manually. Monitor the effect of workloads on transaction log space after upgrade. |
| **tempdb disk space.** Additional space is required by tempdb data and log files because of its use by new features and enhancements to existing features. | Make sure that the data and log files for tempdb are set to autogrow and have sufficient additional disk space or increase the size of each file manually. Monitor the effect of workloads on data and log space after upgrade. |
| **Extended stored procedures.** Extended stored procedures that were previously registered without a full path for the DLL name might fail because the old BINN directory is not added to the new path during upgrade. | Drop the extended stored procedure by using sp\_dropextendedproc, and then register the extended stored procedure with the full path by using sp\_addextendedproc. |
| **Dbo-owned objects.** System objects are now owned by *sys* instead of *dbo*. | Modify scripts that contain statements that query system tables or have search criteria specifying *dbo*. |

**Note**: Table 3-4 lists general behavior changes in SQL Server 2008 related to relational databases. For more information about behavioral changes specific to relational technology, see the remaining sections of this chapter.

For a complete list of behavior changes from SQL Server 2000 and SQL Server 2005 to SQL Server 2008, please review the following SQL Server Books Online topics:

* [Behavior Changes to Database Engine Features in SQL Server 2005](http://go.microsoft.com/fwlink/?LinkID=133143)
* [Behavior Changes to Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133142)

## Preparing for an Upgrade

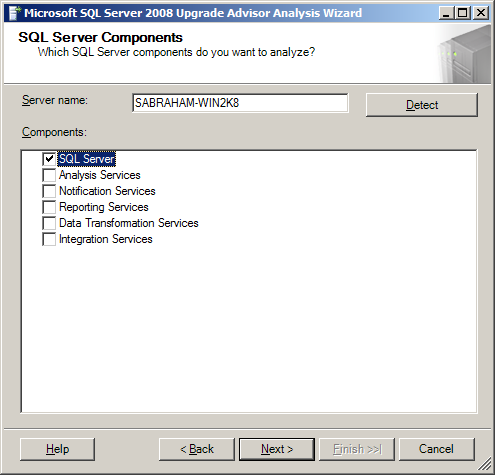
Before you start a relational database upgrade, regardless of type, you must take several steps to prepare for the upgrade and a possible rollback. This section shows the preparation steps that you must follow for an in-place upgrade and then examines the steps for a side-by-side upgrade.

**Important**: For any critical database, you should perform a test run of a side-by-side upgrade to enable extensive application testing before you perform an actual upgrade visible to the online application. This best practice lets you perform any necessary modifications to the application environment in the test environment for verification. Performing a test upgrade can help prevent an unnecessary rollback to the earlier version.

### Preparing for an In-Place Upgrade

You should take the following steps to prepare for an in-place upgrade of an instance of the relational Database Engine and its databases:

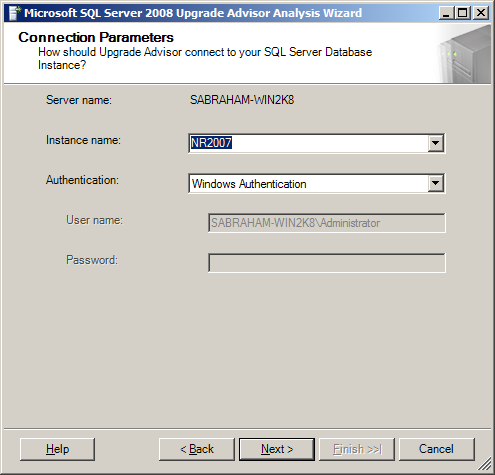
1. Verify that you have met the SQL Server 2008 hardware and software requirements. If you do not meet these requirements, the System Configuration Checker (SCC) part of the SQL Server Setup program will not permit setup to continue.
2. Run the SQL Server 2008 Upgrade Advisor to analyze installed SQL Server 2000 or SQL Server 2005 relational engine components, as the following series of figures shows.
3. First, specify the legacy server name to be analyzed and, for our purposes, select the SQL Server component, as Figure 3-1 shows. If you want to verify the compatibility of other components, you can select them also.



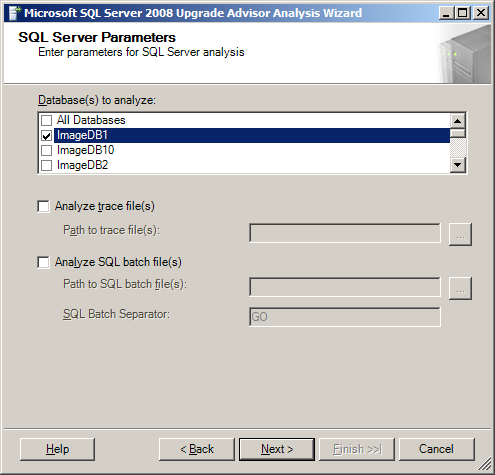
**Figure 3-1:** Selecting components to analyze by using the Upgrade Advisor

1. Next, select the instance name to analyze. In this example, the instance name is NR2007, as Figure 3-2 shows. In this step, you also specify your authentication type and credentials for running the analysis.

**Figure 3-2:** Selecting the instance of SQL Server to analyze

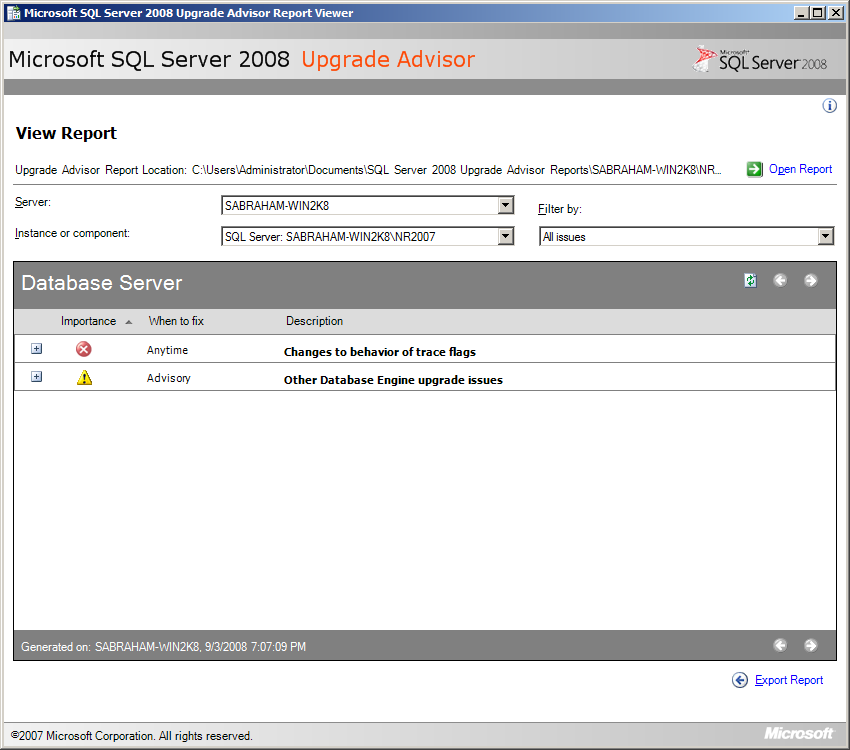


1. Figure 3-3 shows the next Upgrade Advisor window, which lets you select one or more databases to analyze. You also have an option to analyze specified trace files and SQL batch files to help identify any Transact-SQL code that might not port well to SQL Server 2008.



**Figure 3-3:** Selecting a SQL Server 2000 or SQL Server 2005 database to analyze

1. After Upgrade Advisor finishes its analysis, review the generated report. Figure 3-4 shows an example of the upgrade issue report, which shows a list of items to address before you start an upgrade in addition to some items to address after the upgrade. Verify that all pre-setup issues are resolved and that you have a plan for all post-setup issues.



**Figure 3-4:** Reviewing Upgrade Advisor analysis results

1. Back up all system and user databases to make sure that you can roll back the upgrade if it is necessary.
2. Run DBCC CHECKDB on all databases to make sure that they are in a consistent state.
3. Configure the system databases for autogrow and make sure that they have sufficient hard disk space. Additional disk space is required for the system databases in SQL Server 2008 because of changes in system database schema. You can turn off autogrow after the upgrade is complete.
4. Make sure that each user database is set to autogrow and that the PRIMARY filegroup of each user database has sufficient disk space. Additional disk space is required to allow for the additional space that is required for the PRIMARY filegroup when you install SQL Server 2008. You can turn off autogrow after the upgrade is complete.
5. Make sure that the log file for each user database is set to autogrow and has sufficient additional disk space. Additional space is required by transaction log files of user databases. You can turn off autogrow after the upgrade is complete if it is required by your application or database maintenance plan.
6. Set the AUTO\_UPDATE\_STATISTICS option to ON for each database before upgrading to SQL Server 2008 (or run UPDATE STATISTICS after the upgrade is complete rather than wait until the first query hits old statistics). By setting the AUTO\_UPDATE\_STATISTICS option to ON, all statistics are updated when they are first referenced.
7. Disable all startup procedures because they might block the upgrade process. You can re-enable disabled startup procedures after the upgrade is complete.
8. Disable all trace flags before upgrading to SQL Server 2008. Some SQL Server 2000 and SQL Server 2005 trace flags do not exist in SQL Server 2008, and some trace flags have different functionality in SQL Server 2008. If you use trace flags, you should verify after the upgrade that the trace flag has not changed before you enable any previously used trace flags.
9. Stop replication and make sure that the replication log is empty.
10. Prune backup history tables in the msdb database to save time during the upgrade (very large backup history tables can slow down the upgrade process).
11. Exit all applications, including all services that have SQL Server dependencies. Upgrades might fail if local applications are connected to the instance being upgraded.

### Preparing for a Side-by-Side Upgrade

You should take the following steps to prepare for a side-by-side upgrade (the steps will vary somewhat based on the side-by-side method that you select):

1. Run Upgrade Advisor to analyze the database(s) you want to upgrade, as shown in Figures 1-4 earlier. Then review the generated report to verify that you have addressed all issues that must be resolved before the upgrade and to make sure that you understand the upgrade issues that you must resolve after the Setup program is complete.
2. Run DBCC CHECKDB on the databases to be upgraded to make sure that they are in a consistent state.
3. Make sure that the user databases to be upgraded are set to autogrow and that the PRIMARY file group of each user database has sufficient disk space. Additional disk space is required to allow for the additional space that is required for the PRIMARY file group when you install SQL Server 2008. You can turn off this option after the upgrade is complete.
4. Make sure that the log file for each user database is set to autogrow and has sufficient additional disk space. Additional space is required by transaction log files of user databases. You can turn off this option after the upgrade is complete if it is required by your application or database maintenance plan.
5. Set the AUTO\_UPDATE\_STATISTICS option to ON before upgrading to SQL Server 2008. Statistics are not upgraded as part of the upgrade process, and relying on statistics from previous SQL Server releases might result in suboptimal query plans. By setting the AUTO\_UPDATE\_STATISTICS option to ON, all statistics are updated when they are first referenced.
6. Make sure that you have current backups of the databases that you are upgrading by using the BACKUP DATABASE command, and verify their validity by using RESTORE VERIFY ONLY before you start the upgrade process. If you use the detach/attach method, you should make sure that you copy instead of move the data files so that in the unlikely event something should go wrong, you can quickly and easily reattach your data files on the original SQL Server 2000 or SQL Server 2005 instance.

## Performing an Upgrade

After thoroughly preparing to move your SQL Server 2000 or SQL Server 2005 relational Database Engine to SQL Server 2008, you are ready to perform the upgrade. Here are the steps for performing an in-place upgrade and a side-by-side upgrade.

### Performing an In-Place Upgrade

You should take the following steps to perform an in-place upgrade of a relational database:

1. Start the SQL Server 2008 Setup program.
2. Select required SQL Server 2008 components.
3. Select SQL Server Database Services and any other desired components, such as Workstation Components, SQL Server Books Online, and Development Tools.
4. Select the default or named instance of SQL Server 2000 or SQL Server 2005 to be upgraded.
5. Specify the appropriate service account.
6. Specify the logon account information if the instance being upgraded is configured to use Mixed Mode authentication.

**Note**: The in-place upgrade process automatically upgrades all system and user databases.

### Performing a Side-By-Side Upgrade

Regardless of which side-by-side upgrade method that you use, the first step is installing the SQL Server 2008 relational database instance to which you plan to upgrade the SQL Server 2000 or SQL Server 2005 relational database. If you decide to install SQL Server 2008 on the same server together with SQL Server 2000 or SQL Server 2005, make sure that you install SQL Server 2008 with a named instance (you might also verify installed instances during the SQL Server 2008 install process to check which instances are already present on the server) to avoid overwriting existing SQL Server 2000 or SQL Server 2005 instances on the server.

**Note**: In some cases, you might want to copy the system databases, including the master database, from the source SQL Server 2000 or SQL Server 2005 instance to the SQL Server 2008 instance before transferring user databases. For more information, see [Moving System Databases](http://go.microsoft.com/fwlink/?LinkId=134509) in SQL Server 2008 Books Online.

Then you can upgrade the SQL Server 2000 or SQL Server 2005 user databases by following the steps associated with the side-by-side upgrade method that you chose.

#### Backup/Restore Upgrade Method

Take the following steps to upgrade a user database by using the backup/restore upgrade method:

1. Back up the database to be moved from the SQL Server 2000 or SQL Server 2005 instance by using either SSMS or the BACKUP DATABASE Transact-SQL statement.
2. Use SSMS to connect to the SQL Server 2008 relational database instance to which you want to restore the SQL Server 2000 or SQL Server 2005 relational database.
3. Restore the relational database from the backup file, changing the database or file names and locations as necessary.

**Important**: You must manually move to the SQL Server 2008 instance the master and msdb database objects related to the database being upgraded (for example, logins, jobs, alerts).

#### Detach/Attach Upgrade Method

Take the following steps to upgrade a user database by using the detach/attach upgrade method:

1. Detach the database to be moved from the SQL Server 2000 or SQL Server 2005 instance by using SQL Server Enterprise Manager, SSMS, or the sp\_detach\_db stored procedure.
2. Copy (or move) the detached data file(s) and log file(s) to the new server.
3. Attach the copied data and log files to the SQL Server 2008 instance by using SSMS or the CREATE DATABASE Transact-SQL statement with the FOR ATTACH or FOR ATTACH\_REBUILD option.
4. Optionally, if you copied the original data and log files, reattach the original data and log files to the previous instance of SQL Server 2000 or SQL Server 2005.

**Important**: You must manually move to the SQL Server 2008 instance the master and msdb database objects related to the database being upgraded (for example, logins, jobs, alerts).

#### Copy Database Wizard Upgrade Method

Take the following steps to upgrade a user database by using the Copy Database Wizard upgrade method:

1. Make sure that you have the required permissions on the appropriate servers.
   1. For the detach/attach method, you must be a member of the sysadmin fixed server role on both the source and destination servers.
   2. For the SMO transfer method, you must be a database owner for the source database and must either have been granted the CREATE DATABASE permission or be a member of the dbcreator fixed server role on the destination server.
2. Specify the source and destination servers.
3. Specify the databases to be moved or copied.
   1. For the detach/attach method, active sessions must not exist when the copy or move operation is tried, or the Copy Database Wizard will not execute the move or copy operation.
   2. For the SMO transfer method, active connections are allowed because the database is never taken offline.
4. Specify the name of the target database if different from the source database.
5. Specify other objects to be moved, such as logins, shared objects from the master database, jobs, maintenance plans, and user-defined error messages.
6. Specify a schedule for the copy or move operation if you want it scheduled for a later time.
7. If you are not a member of the sysadmin fixed server role, you must specify a SQL Server Agent Proxy account that has access to the SSIS package execution subsystem. For information about how to create a proxy account, see [How to: Create a Proxy (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkId=134510) in SQL Server 2008 Books Online.

**Important**: You must manually move to the SQL Server 2008 instance certain master and msdb database objects related to the database being upgraded (such as backup devices, linked server definitions, and alerts).

## Post-Upgrade Tasks

You should take the following actions after you upgrade a relational database to SQL Server 2008 to make sure that the upgrade ran successfully and to configure the relational Database Engine in addition to the upgraded relational database.

### In-Place Upgrade

For in-place upgrades, execute the following steps:

1. Apply available service packs or updates to the upgraded SQL Server 2008 instance.
2. Reregister your servers. For information about registering servers, see [Registering Servers](http://go.microsoft.com/fwlink/?LinkId=134511) in SQL Server 2008 Books Online.
3. Configure the SQL Server installation. To reduce the attackable surface area of a system, SQL Server selectively installs and enables key services and features. Therefore, you must configure the new instance to meet your specific needs.

### Side-by-Side Upgrade

For side-by-side upgrades, execute the following post-upgrade steps:

1. Configure/update server logins on the new instance and database users in the upgraded database.
2. Configure jobs and database maintenance plans on the new instance.
3. Configure alerts on the new instance.
4. Configure DTS/SSIS packages on the new instance.
5. Update connection strings at clients so that they can connect to the new instance, unless you are replacing the old server with a new server that has the same identity.

### General Post-Upgrade Tasks

Whether you are doing an in-place upgrade or a side-by-side upgrade, you must execute the following post-upgrade steps:

1. Execute DBCC CHECKDB WITH DATA\_PURITY to check the database for column values that are not valid or are out of range. After you have successfully run DBCC CHECKDB WITH DATA\_PURITY against an upgraded database, you do not have to specify the DATA\_PURITY option again because SQL Server will automatically maintain "data purity." This is the only DBCC CHECKDB check that you must run as a post-upgrade task.
2. Execute DBCC UPDATEUSAGE on all attached databases to update usage counters and make sure that correct values exist for table and index row counts.
3. Update statistics on all databases after you upgrade them. Execute UPDATE STATISTICS in user-defined tables in SQL Server databases.
4. Repopulate full-text catalogs. For information about this task, see Chapter 6, "Full- Text Search."
5. Make sure that the relational databases are working correctly by executing a sample set of queries.
6. Update any scripts affected by SQL Server 2008 behavior changes.

**Full-Text Upgrade.** Any databases that were marked full-text enabled or disabled before the upgrade will maintain that status after the upgrade. After the upgrade, the full-text catalogs will be rebuilt and populated automatically for all full-text-enabled databases. This is a time- and resource-consuming operation. For more information, see Chapter 6, "Full-Text Search."

**Database Maintenance Plans.** Database maintenance plans in SQL Server 2000 consisted of Transact-SQL commands executed by SQL Server Agent. Starting with SQL Server 2005, database maintenance plans are SSIS packages. If you upgrade from SQL Server 2000 to SQL Server 2008, your database maintenance plans will still work. However, they will be listed under the legacy branch of SSMS. To upgrade these packages to SQL Server 2008, right-click the plan you want to upgrade and select Migrate. For more information about this process, see Chapter 1, "Upgrade Planning and Deployment," and Chapter 2, "Management and Development Tools." You can also find more information in [How to: Migrate SQL Server 2000 Database Maintenance Plans](http://go.microsoft.com/fwlink/?LinkID=134193) in SQL Server Books Online.

**Changes in Caching Behavior.** There are several changes in caching behavior among SQL Server 2000, SQL Server 2005, and SQL Server 2008. For more information about changes between SQL Server 2000 and SQL Server 2005 RTM and SP2, see [Changes in Caching Behavior between SQL Server 2000, Server 2005 RTM and SQL Server 2005 SP2](http://go.microsoft.com/fwlink/?LinkId=134519), and the [SQL Programmability & API Development Team Blog](http://go.microsoft.com/fwlink/?LinkId=134522). For more information about how plan caching and reuse is handled in SQL Server 2008, see [Execution Plan Caching and Reuse](http://go.microsoft.com/fwlink/?LinkID=120424) in SQL Server Books Online.

### Use Plan Hints

Another post-upgrade task that you have to perform is validating or removing USE PLAN hints that were used by SQL Server 2005 and applied to queries on partitioned tables and indexes. SQL Server 2008 changes the way queries on partitioned tables and indexes are processed. Queries on partitioned objects that use the USE PLAN hint for a plan generated by SQL Server 2005 might contain a plan that is not usable in SQL Server 2008. We recommend the following procedures after you upgrade to SQL Server 2008.

When the USE PLAN hint is specified directly in a query, take the following steps:

1. Remove the USE PLAN hint from the query.
2. Test the query.
3. If the optimizer does not select an appropriate plan, tune the query, and then consider specifying the USE PLAN hint with the desired query plan.

When the USE PLAN hint is specified in a plan guide, follow these steps:

1. Use the sys.fn\_validate\_plan\_guide function to check the validity of the plan guide. Or, you can check for invalid plans by using the Plan Guide Unsuccessful event in SQL Server Profiler.
2. If the plan guide is not valid, drop the plan guide. If the optimizer does not select an appropriate plan, tune the query, and then consider specifying the USE PLAN hint with the query plan that you want.

An invalid plan will not cause the query to fail when the USE PLAN hint is specified in a plan guide. Instead, the query is compiled without using the USE PLAN hint. For more information about query processing on partitioned objects, see [Query Processing Enhancements on Partitioned Tables and Indexes](http://go.microsoft.com/fwlink/?LinkId=134526) in SQL Server 2008 Books Online.

### Important Information About Query Plans

Be aware that how SQL Server determines query plans between versions varies—sometimes significantly. When you upgrade to a new version of SQL Server, it is always important to review changes to query plans to guarantee optimal performance. For more information about query plan tuning, see the following SQL Server 2008 Books Online topics:

* [Migrating Query Plans](http://go.microsoft.com/fwlink/?LinkId=134528)
* [Transact-SQL Statements That Produce Showplans](http://go.microsoft.com/fwlink/?LinkId=134529)
* [Finding and Tuning Similar Queries by Using Query and Query Plan Hashes](http://go.microsoft.com/fwlink/?LinkId=134530)

SQL Server 2008 also includes many improvements to query plans on partitioned tables and indexes. Some new improvements include an improved algorithm for identifying the best parallel execution strategy, an improved seek mechanism for partitioned tables, and additional information displayed in the query execution plans for queries that include partitioned tables. For complete information about these improvements, see [Query Processing Enhancements on Partitioned Tables and Indexes](http://go.microsoft.com/fwlink/?LinkID=133145) in SQL Server 2008 Books Online. Also see [Behavior Changes to Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133142) for the latest information about plans over partitioned tables.

## Connecting Client Applications to SQL Server 2008

After you have upgraded your instance or moved to a new instance and validated that your databases are functioning correctly, you must verify that client applications can connect to your new instance. Table 3-5 highlights some potential issues that might affect client connectivity.

Table 3-5: Issues That Might Affect Client Connectivity

| **Issue** | **Description** |
| --- | --- |
| Network protocols | The only supported network protocols are now TCP/IP Sockets, named pipes, VIA, or shared memory. If your application is using network protocols that are not in this list, it will not work. |
| SQL-DMO-based WMI providers | If your application uses DMO-based management APIs, you must upgrade to either the SMO-based management APIs or the WMI for Configuration management APIs. SMO is written by using the managed code APIs. WMI for Configuration is written by using unmanaged code APIs. |
| DB-Library | Before SQL Server 7.0, the primary mechanism for client/server communication between SQL Server and client applications was DB-Library. Although DB-Library was still included with SQL Server 2000, Microsoft announced that it was being deprecated. With the release of SQL Server 2005, DB-Library support is limited to SQL Server 7.0 features. |
| Network communication | By default network communication might be disabled in new SQL Server 2008 installations and must be enabled through the Server Network Communication tool. |

## Conclusion

SQL Server 2008 offers many improvements over SQL Server 2000 and SQL Server 2005, in addition to many great new features. The first step in taking advantage of these improvements is upgrading your existing databases to SQL Server 2008. As this chapter explains, you have two main choices for upgrading SQL Server 2000 and SQL Server 2005 databases to SQL Server 2008: in-place or side-by-side. And if you select the side-by-side method, you have additional choices to make, including whether to use backup/restore, detach/attach, or the Copy Database Wizard—or another alternative. Each of these options has its pros and cons, so you have to make sure that you understand your organization’s current configuration and needs. Then you have to prepare thoroughly and test extensively to make sure that an upgrade is successful and ready for production.

## Additional References

For an up-to-date collection of additional references for upgrading SQL Server 2008, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkId=132510).

For the latest updates to SQL Server 2008 Books Online, see the [Microsoft SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132511).

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkId=132512)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkId=132515)

# High Availability

## Introduction

It is almost guaranteed that when upgrading any major server hardware component or software version, you will encounter some sort of outage. The problem is that in today's world of 24x7 computing, companies of all sizes have less and less tolerance for downtime. This chapter discusses how to minimize downtime when upgrading from SQL Server 2000 or SQL Server 2005 to SQL Server 2008 and how to upgrade the high-availability features of failover clustering, log shipping, database mirroring, and replication. This chapter does not cover how each high-availability feature works unless it is different in SQL Server 2008 and has a bearing on the upgrade process. Prior basic knowledge of high-availability features is assumed.

## Preparing to Upgrade

Before looking at how to have highly available upgrades to SQL Server 2008, this section details the types of upgrades possible and how each rates with regard to high availability. The three main upgrade options are an in-place upgrade, a side-by-side upgrade, and going to a separate server or new cluster. For a detailed discussion of upgrade types and their advantages and disadvantages for different scenarios, see Chapter 1, "Upgrade Planning and Deployment."

**Important:** During upgrade, the original instance of SQL Server and its databases remain available until the Database Engine upgrade process begins. This means that during the checks and file copies, you can use the instance and allow connections. However, the upgrade process does not let you know when it starts processing the database or that other objects might be in use. It is therefore always best to ensure that all connections are terminated and transactions are complete before the setup or upgrade process has started. You do not want to run the risk of someone issuing a query after a final backup has already been performed, thus invalidating it.

### In-Place Upgrade

An in-place upgrade uses the SQL Server 2008 Setup program to directly upgrade an instance of SQL Server 2000 or an instance of SQL Server 2005 to SQL Server 2008. The "Upgrade Strategies" section in Chapter 1 describes the in-place upgrade process in detail.

An in-place upgrade has the highest risk of long downtime compared with the other upgrade methods because it upgrades the existing instance of SQL Server on the same server. Because the upgrade is happening to the source instance on the same hardware, there is no way to confine downtime to only a "switch" from old to new. The instance and user databases will be unavailable during the entire time of the upgrade. For some organizations, this factor might outweigh the benefits of hardware reuse and not having to perform other required tasks (such as scripting logins) when switching to another server.

However, a bigger risk with an in-place upgrade is that there is no easy way to determine the state of the instance or a particular database if something fails during the upgrade. The worst-case scenario is a full reinstall of Windows and SQL Server because the file versions might be in a mixed state. After the reinstallation, you would have to restore the databases and objects from backups and scripts. Any fallback plan must include steps to potentially rebuild the server from scratch to the way it was before the upgrade process started. There is no other fallback plan than backup and restore.

### Side-by-Side Upgrade on the Same Server or Cluster

This type of side-by-side upgrade moves all or some data from an instance of SQL Server 2000 or SQL Server 2005 to a separate instance of SQL Server 2008 that resides on the same server or cluster. The "Upgrade Strategies" section in Chapter 1 describes this process in detail.

A side-by-side upgrade on the same server or cluster offers slightly better protection than an in-place upgrade because the old configuration for the most part is still available if something fails. The real cost is the downtime in moving data and objects to the new instance. However, depending on the method used, the downtime could be just a matter of minutes.

Problems with a side-by-side upgrade on the same server include the following:

* It is impossible to go back to the original configuration as it existed before the upgrade process began. Installing SQL Server 2008 creates a server that replaces some components of the old SQL Server version with those of SQL Server 2008. This should not pose any major problems, but you need to realize that although your old configuration might appear to still be intact, it really is not.
* You will still encounter downtime because some SQL Server 2008 components require a reboot to install.
* SQL Server 2008 is still bound by the same rules as SQL Server 2000 or SQL Server 2005 regarding instance names; there can be only one default instance, and everything else is a named instance. If the existing instance is the default instance, the SQL Server 2008 instance must be a named instance. That might pose problems for some applications that do not support named instances. Through thorough testing, you can easily determine if this is an issue in your implementation and, if so, account for it. Otherwise, users or applications might not be able to connect to the upgraded database.
* Related to the previous bullet point, redirecting users and applications to the new instance or database might prove challenging. Make sure to work out any such issues long before the upgraded solution goes into production.
* Perhaps the biggest challenge in a side-by-side upgrade is going back to the previous installation, if it is necessary. If an organization uses the new system in production and then determines that the new environment is not suitable for production, getting data from the new system and back into the old one will not be a trivial task. Many applications do not provide ways to extract data easily (if at all), and a backup of SQL Server cannot be applied to a later version of SQL Server.

### Side-by-Side Upgrade to a Separate Server or Cluster

This variation of a side-by-side upgrade moves all or some data from an instance of SQL Server 2000 or SQL Server 2005 to a separate instance of SQL Server 2008 that resides on a different server. This strategy potentially offers the best of both worlds: using separate servers or clusters leaves the old environment intact and available for use in a fallback plan, and you can minimize downtime by using whatever method is best for moving the data to the new instance. Other advantages to this option are that you can get the hardware up and running as well as tested long before a cutover date and you can use the new environment to test the process of the actual data move. This ability is invaluable to a smooth transition.

Using a separate server has the same drawbacks as a same-server side-by-side upgrade. Application and end-user redirection, and default versus named instances are still issues. A new wrinkle in this scenario is that although you are using new hardware and can create a new default instance, you cannot have two objects with the same name in the same domain. Assuming that the old SQL Server default instance is on a stand-alone server named MYSERVER, for example, the new server (and default instance) could not use MYSERVER. It could be MYNEWSERVER, but that is clearly a different name. The correct way to solve this issue is to decommission the old server or instance and then rename the new server or instance to MYSERVER so that applications and uses can continue to use that name. You could also consider other workarounds such as using DNS aliases, but that option puts a burden on network administrators and is only a patch that obscures the underlying issue.

### Decommissioning and Disabling the Original Instance or Database in a Side-by-Side Upgrade

After your new instance has passed acceptance tests and the newly upgraded server is successfully in production, you will likely want to schedule a time to either uninstall the instance or fully decommission the server, as discussed in "Decommission and Uninstall After a Side-by-Side or New Hardware Upgrade," in Chapter 1. The key is to ensure that the new instance or server is correct and stable before tearing down the old environment. Decommissioning and disabling the original instance too soon could be a mistake, and your downtime will depend on your fallback plan. It is cheaper to switch back to the old, known, already configured environment than to have to restore from scratch.

However, before cutting over to the new instance, you must first disable the old one so that users and applications will not accidentally connect to the wrong server. The business is expecting to use the newly upgraded instance and databases, and moving the data if this occurred would be painful.

### Methods for Side-by-Side Upgrades to a Separate Server or Cluster

There are several methods for moving your SQL Server databases in a side-by-side upgrade involving new hardware. Chapter 1, "Upgrade Planning and Deployment," and Chapter 3, "Relational Databases," cover these methods in detail. This section discusses the high-availability aspects of each one.

#### Backup and Restore

The downtime associated with upgrading a database by using a database backup is related to the size of the database as well as the efficiency of the underlying network and disk I/O. It will take time to copy the database backup from Server A to Server B (the network and the disk I/O of both servers will come into play), and the restore speed will depend on the hardware configuration (including disk) of Server B. You should not have any downtime related to actually performing the backup because all SQL Server backups can be made while the database is online; if any transactions occur after the backup is initiated, a differential backup or series of transaction log backups can be performed to "catch up" the destination database.

If the database is small to mid-sized, it is relatively easy to back up, copy, and restore it in a reasonable amount of time. However, if you have a very large database (VLDB), in the hundreds of gigabytes or terabyte range, waiting until the cutover time to first do the backup, copy, and restore might exceed the allowed outage window. Just copying a terabyte database could take a day or more. That is definitely *not* a highly available upgrade and will need to be accounted for in your upgrade plan. For more information about VLDBs, see "Upgrading Very Large Databases," in Chapter 1.

There are a few things you can do to increase the speed of the backup, copy, and restore process:

* Use third-party compression tools, which not only shrink the backup size (which means a quicker copy time) but generally speed up the backup and restore process as well.
* Use hardware-based (i.e., SAN-based) backups to make the backup, and then attach the backup (and restore it) using lower-level technologies that are transparent to the hardware. This strategy assumes that the source and destination servers are on the same storage unit. Hardware-based backups are not an option if this configuration is not already set up. See "Hardware-Based Installation" below for more information.

#### Detach and Attach

Detaching and attaching databases is not the same as performing a backup, but both methods involve physically copying files. The big difference is that the former method copies actual data and log files. There will be complete downtime during the detach and copy because once the database is detached, it is no longer part of the source instance until it is reattached. This method might not be an optimal solution for minimizing downtime and could pose risk. In addition, if the database is large, this approach is impractical.

#### Log Shipping

Log shipping is traditionally used only as a high-availability or disaster recovery solution, but it is also a useful upgrade option. The log shipping method is based on backup and restore and requires that the source database be configured to be able to make transaction log backups (so the Simple recovery mode would not work). It offers a way to minimize downtime because the only necessary outage occurs during the switch from Server A to Server B. Most of the work—including getting a new server up and running and performing the initial backup, copy, and restore—can be done well in advance of the actual cutover.

The problem is that while SQL Server 2008 can restore backups from SQL Server 2000 or SQL Server 2005, the log shipping features of those earlier versions of SQL Server cannot be used to configure log shipping to SQL Server 2008. Custom log shipping scripts (such as the ones listed in "Additional References" at the end of the chapter) would need to be used.

#### Hardware-Based Installation

Besides the traditional options for moving databases, another approach has become increasingly cheaper over the years: hardware-based moves via shared storage, such as a SAN. Many companies deploy a large portion of their servers on one or more storage units. Failover clustering requires such shared storage. Assuming that the source and target servers are on the same storage unit and that the appropriate options are configured on the hardware, this option opens up a wide range of possibilities for minimizing downtime.

A hardware-assisted backup is generally known by most storage vendors as a "snapshot," "clone," or something similar. What happens is that a backup is initiated outside of SQL Server, and the disks being used by SQL Server are essentially cloned and snapped off. The clone can then be attached to another server on the same storage unit. This process is nearly instantaneous, and you can use the snapshot or clone for a restore (traditional or hardware-based). Within SQL Server, I/O is "frozen" briefly and then "thawed" to ensure consistency behind the scenes. Although this process takes a few seconds, it should be completely transparent.

It's important to note that for this hardware-based approach to be properly implemented, a storage vendor must support SQL Server's Virtual Device Interface (VDI). Otherwise, the process could damage the SQL Server database. The storage vendor might impose some limitations, such as allowing only one database or file per disk, which must be verified before any implementation. Work with the storage administrators to not only investigate whether this is an option but to make sure SQL Server is implemented properly to take advantage of the feature. For more information about this strategy, see the [SQL Server 2005 Virtual Device Interface (VDI) Specification](http://go.microsoft.com/fwlink/?LinkId=132170) white paper.

### Which SQL Server Upgrade Method Should You Use?

Ultimately, the upgrade method chosen should meet not only an organization's technical needs but also minimize the downtime needed for the upgrade. If a side-by-side upgrade is necessary, log shipping or hardware-based methods should strongly be considered to reduce the outage. There will never be zero downtime in an upgrade; the issue is how much downtime can be tolerated. For valuable information about choosing an upgrade strategy, see "Considerations for Choosing an Upgrade Strategy" in Chapter 1.

## Minimizing Downtime

This section covers the key strategies for promoting highly available upgrades for all installations of SQL Server, minimizing downtime in the switch to SQL Server 2008.

### Prepare for SQL Server 2008

Before putting together an upgrade plan or executing an upgrade to SQL Server 2008, you must take into account the following considerations.

#### Learn What Has Changed

When it comes to high availability, one of the best ways to minimize downtime is to anticipate changes and account for them accordingly. Although some changes might appear to be only performance-related or involve feature changes that do not seem relevant to your organization's product usage, if one of those changes causes a performance or operational issue, it will be perceived as downtime. If an upgrade is properly planned and executed, it should not introduce large performance degradations or problems. And a database administrator should never be caught by surprise. For a complete list of backward-compatibility issues in SQL Server 2008, see [Backward Compatibility](http://go.microsoft.com/fwlink/?LinkId=132171) in SQL Server 2008 Books Online.

#### Prepare Applications

For information about how to prepare applications for a database upgrade, see the "Application and Connection Requirements" section in Chapter 1. Treating the SQL Server upgrade as an isolated event that will not affect anything else is a mistake and may cause downtime.

#### Update Skills

Ensuring that the database administrator staff is fully trained and prepared for the upgrade will increase uptime. For information about how to make sure that those administering or deploying SQL Server 2008 are ready, see the "Treat the Upgrade as an IT Project" section in Chapter 1.

#### Check Minimum Hardware Requirements

Make sure that the hardware you plan to use for SQL Server 2008 meets the minimum requirements documented in the "Minimum Hardware Requirements," in Chapter 1, as well as in [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132172) in SQL Server 2008 Books Online. It is a bad idea when it comes to availability to assume that the current server will meet a company's future needs. New software generally has higher requirements. If current servers are near capacity or exceed it, you will be facing availability and performance issues. These issues must be resolved prior to deployment, or they will cause a much larger upgrade outage than necessary.

#### Disk Space

Disk space, or lack thereof, is one of the major causes of downtime. During the installation of SQL Server 2008, SQL Server will tell you how much disk space is needed. For more information, see the "Hard Disk Space Requirements" section in Chapter 1. [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132172) also lists how much disk space the program files consume.

Besides accounting for the program files and system databases, follow the recommended guidelines to account for the disk space needed during the upgrade. The downtime spent recovering and then attempting to continue upgrading a VLDB that failed due to lack of disk space can easily be avoided through proper planning. You also need to size all databases and their files appropriately after the upgrade, especially tempdb.

In addition, make sure you account for backup space during the upgrade, as noted in "Perform Database Backups" later in this chapter, and in the "Plan for Backups" section in Chapter 1.

### Devise an Upgrade Plan

The "Developing an Upgrade Plan" section in Chapter 1 provides extensive resources for how to put together a plan for the upgrade to SQL Server 2008. The goal is to have a solid plan that is straightforward, minimizes downtime, and is successful. Because databases are more than just data and log files sitting on some storage, it takes the coordination of not only the database administrators, but all teams involved (application, storage, operating system, network, and so on) to minimize downtime. Attempting to do an upgrade without a plan—whether you are upgrading one database or a million—is a surefire way to increase your risk of downtime. Upgrading a major version of any software should not be treated lightly. The key is having the appropriate measures in place to be able to recover to how the system was before the upgrade took place. This is commonly known as having a fallback plan.

**Important:** As part of the fallback plan, you need to set "go" and "no-go" points. For example, if you need to be up and running for Monday's business but your upgrade looks like it will exceed the window you planned for, put your contingency plan into place. Know how long the contingency plan will take to execute. If you know that it takes five hours to execute and the business needs to be up by 7AM Monday, your no-go point would be around 2AM Monday. Make sure the "go" points are listed steps or items that when checked, verify that the upgrade can happen. Missing one of the steps could cause problems down the road.

### Test the Upgrade Plan

The best defense is a good offense, and one of the best things you can do to minimize upgrade downtime is to test the upgrade plan. Most downtime is directly related to the lack of testing. Testing requires a big commitment from the entire organization. How easy it is to test the upgrade plan is directly related to the selected upgrade method and the complexity of your applications and environment.

If you find that the plan is flawed halfway through the upgrade, it is too late to fix problems. Finding the problems and either accounting for or correcting them before beginning the upgrade is essential for a successful upgrade. One goal of testing the upgrade and fallback plans is to know approximately how long the process should take to perform. If an unforeseen problem is encountered during the upgrade and the fallback plan is set into motion, all that management will want to know is how long it will take to get back to a usable state. Without testing, it is anyone's best guess, and chances are, management will not want to hear, "I don't know." It is also important that all plans include application testing steps; upgrades are not just an IT exercise.

Testing does not provide a 100 percent guarantee that problems will not be encountered. There might be some differences in production that cannot be replicated in the test environment. For example, many companies do not use SQL Server failover clustering in their test environment, but it is implemented in production. Performing tests against a stand-alone server will validate that the databases work after an upgrade to SQL Server 2008, but the upgrade process for a stand-alone server is different than that for upgrading a cluster. Comprehensive testing helps mitigate such risks and reduces downtime.

### Prepare Servers and Instances for SQL Server 2008

To minimize overall downtime during the upgrade window, you can perform several tasks before the actual upgrade, as follows.

#### Check SQL Server Versions

The first step to minimizing downtime is to ensure that the instances containing the databases targeted for upgrade are at the proper SQL Server versions. Discovering only at upgrade time that the source instance and database are at an incompatible version could cancel the upgrade or increase downtime by requiring you to apply the appropriate—and most likely untested—updates to the source. "Allowable Upgrade Paths," in Chapter 1, describes the versions of SQL Server that can be upgraded to SQL Server 2008.

#### Install Prerequisites

As noted in Chapter 1, SQL Server 2008 requires [.NET Framework 3.5 SP1](http://go.microsoft.com/fwlink/?LinkID=120550) and [Windows Installer (MSI) 4.5](http://go.microsoft.com/fwlink/?LinkID=49112). The .NET Framework is a side-by-side architecture, so installing .NET Framework 3.5 SP1 should cause little downtime on a server; earlier versions of the .NET Framework can be updated but not removed. Any software that depends on the previous .NET Framework version should not, in most cases, be affected by the installation of the newer version. The same applies for MSI 4.5.

However, installing .NET Framework 3.5 or MSI 4.5 will cause some downtime. That outage can happen either during a scheduled maintenance window before the SQL Server 2008 upgrade or during the upgrade itself. Installing these components during a scheduled maintenance window will minimize the downtime and reboots required during the upgrade. Remember to test this configuration before a production rollout to ensure that there are no conflicts with .NET Framework 3.5 SP1 or MSI 4.5.

No administrator should install these two components whenever they like. Many instances of SQL Server now host more than one database application, so an unplanned, unscheduled, and unknown upgrade could wreak havoc with multiple applications. These installations might be easier to perform on a cluster with more than two nodes, where one (or more) is a dedicated failover node. The server acting as a dedicated failover node can be updated and rebooted without affecting anything else, assuming that the node does not currently own any instances.

Consider the following example of a four-node cluster with five instances of SQL Server installed on it. Instance A is on Node A, instances B and C are on Node B, and instances D and E are on Node C. Node D is purely a failover node. On Node D, you could easily install both .NET Framework 3.5 SP1 and MSI 4.5 with no associated downtime because that node is the primary failover node. The other nodes are actively hosting instances, so unless there is a scheduled maintenance window, you do not want to install on any of them and cause an availability outage unless absolutely necessary. Of course, you could fail the instance or instances to another server, but that would be an outage. Do you then fail that instance back? You need to consider all these types of issues.

#### Run a Full DBCC CHECKDB

SQL Server provides the Transact-SQL DBCC CHECKDB command to check and ensure the health of SQL Server databases. A healthy database generally ensures higher availability. For information about using DBCC CHECKDB in conjunction with an upgrade, see Chapter 2, "Management and Development Tools." The problem with running a full DBCC CHECKDB is that it generally affects a database's availability, and how long it takes to run is directly related to the size of the database. The benefits generally outweigh the downtime, but there might be service-level agreements (SLAs) that prevent DBCC CHECKDB from being run or that make it very difficult to complete within the set window.

It is also recommended that you run DBCC CHECKDB after the database is upgraded to SQL Server 2008 to ensure that nothing went wrong or was introduced as a result of the upgrade. That means more downtime during the upgrade process. Although some people might consider the downtime required to run DBCC CHECKDB before and after the upgrade excessive, the assurance it provides might be worth it.

#### Run SQL Server 2008 Upgrade Advisor

SQL Server 2008 Upgrade Advisor is a free tool that Microsoft provides to help detect problems before the upgrade. Running this tool is an essential part of your upgrade-preparation phase, as described in the "SQL Server 2008 Upgrade Advisor" section in Chapter 1.

#### Run SQL Server 2008 Upgrade Assistant

SQL Server 2008 Upgrade Assistant is an external tool that uses Upgrade Advisor, along with baseline and trace replay, in a test environment to help identify compatibility issues between SQL Server 2000 or SQL Server 2005 and SQL Server 2008. For more information, see the "SQL Server 2008 Upgrade Assistant" section in Chapter 1.

#### Run the Appropriate Best Practices Analyzer

The Best Practices Analyzer (BPA), which comes in versions for SQL Server 2000 and SQL Server 2005, is an invaluable tool for finding issues with current deployments so that they do not cause an upgrade to fail—or worse, come along for the ride and cause a potential outage down the road. For more information, see the "Best Practices Analyzer for SQL Server 2000 and SQL Server 2005" section in Chapter 1.

#### Perform Database Backups

The more important thing you can do to protect data in an upgrade is to perform backups at the appropriate times. The "Plan for Backups" section in Chapter 1 covers this essential topic. How many backups you need to perform depends on the point to which the data might need to be recovered.

Never decommission an instance or database until a final full backup is made. During the upgrade process, it is also recommended that you perform backups at certain points so that in case something goes wrong, the upgrade process will not have to start from the beginning. For example, suppose application changes occur before the upgraded database goes live in production and something goes wrong. If a database backup was made before those changes occurred, you will experience some downtime but not nearly as much as if you had go back 18 hours to the initial backup, copy, and restore. In any catastrophic failure, the only way to recover is with good, proper backups. Not making them will most certainly increase, not decrease, downtime. The worst thing about backups is that they consume disk space, but you can delete them some time in the future when you no longer need them.

#### Script or Export All Objects

No SQL Server high-availability method except failover clustering accounts for objects that reside outside the database. (Failover clustering provides instance-level protection.) Chapter 2, "Management and Development Tools," and Chapter 13, "Integration Services," discuss how to use scripting or use tools such as SSIS to move objects from one database or instance to another. These objects include instance-level logins, linked servers, SQL Server Agent jobs, user-created stored procedures, and so on.

To minimize downtime, prepare any scripts or SSIS packages before the upgrade and in cases where sensitive information may be stored (such as passwords), secure them properly. Upgrades can fail when an application no longer functions properly because an object such as a linked server is not configured.

Even if you do not use any of the high-availability technologies discussed in this chapter, using scripts or SSIS packages for objects involved in the upgrade provides an insurance policy like backups in case a complete failure occurs. Having these objects scripted or exported out of the original system could mean the difference between some downtime and trying to track down the consultant who implemented the system years ago to see if he or she remembers those key elements.

#### Upgrade Common Components

Another way to minimize downtime is to upgrade the common components to the versions SQL Server 2008 requires ahead of the actual SQL Server 2008 upgrade. Do not install any upgrades without performing due diligence and having reasonable confidence that the new elements will not destabilize the current production environment and cause downtime.

#### Full-Text Search

Chapter 6, "Full-Text Search," covers the strategies for upgrading full-text search, which is now integrated into the Database Engine in SQL Server 2008. Thus, there will no longer be a dedicated cluster resource for full-text in the resource group with SQL Server if failover clusters are deployed. In addition, if you use full-text indexes, you will have more to upgrade, so plan accordingly for more downtime.

#### Replication

Unlike SQL Server 2005 Setup, which let you skip installation of replication, replication is a required and included component of SQL Server 2008, even if replication was not deployed as part of an existing SQL Server 2005 instance. For details about upgrading replicated databases, see "Upgrading Replicated Databases" later in this chapter.

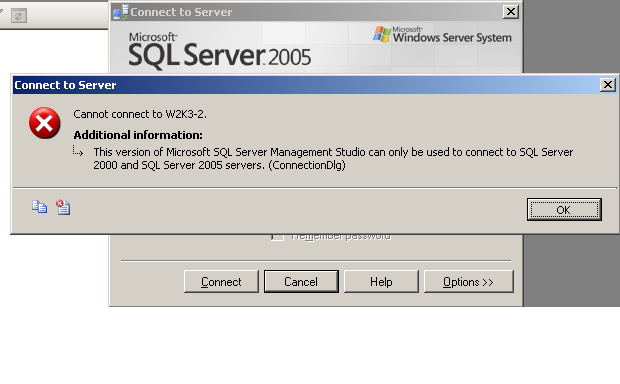
#### Service Accounts

During an in-place upgrade, Setup will not prompt you to change the existing instance's service accounts. This means that the existing SQL Server 2000 or SQL Server 2005 service accounts will be used for SQL Server 2008. This means that any previous version's service account will have full access to the new SQL Server 2008 instance.

We do not recommend using the existing SQL Server 2000 or SQL Server 2005 service accounts for the upgraded instances of SQL Server 2008. Create new domain-based service accounts with the correct privileges on each server instead. After the upgrade, use SQL Server Configuration Manager (covered in Chapter 5, "Database Security") to update the services to use these new accounts.

#### Management Tools and Utilities

After the upgrade, only SQL Server 2008 tools and utilities can manage SQL Server 2008 instances. If you try to use SQL Server 2005 Management Studio to manage SQL Server 2008, you will see the message that Figure 4-1 shows. However, you can use the SQL Server 2008 tools to manage instances of SQL Server 2005. If instances of SQL Server are managed remotely, install these tools on a workstation to make sure you can manage SQL Server 2008 after it is installed.



**Figure 4-1:** SQL Server 2005 Management Studio is not able to connect to an instance of SQL Server 2008

**Note:** When you upgrade in-place to SQL Server 2008, the SQL Server 2008 install process might not remove the existing management tools for the legacy version of SQL Server. After the upgrade, check to see if the old tools versions still exist and, if it is necessary, uninstall them.

## Upgrading Failover Clusters

This section describes the considerations and processes for upgrading existing SQL Server 2000 or SQL Server 2005 failover clustering instances to SQL Server 2008. This is not intended as a formal SQL Server 2008 failover clustering white paper. As of this writing, no SQL Server 2008 failover clustering white papers had been published, but check the [Microsoft SQL Server Web site](http://go.microsoft.com/fwlink/?LinkID=116427) to see if any have been released.

For comprehensive information about upgrading failover clusters, see [Upgrading a SQL Server 2008 Failover Cluster](http://go.microsoft.com/fwlink/?LinkId=132180) in SQL Server 2008 Books Online.

### Feature Changes in SQL Server 2008 Failover Clustering

SQL Server 2008 failover clustering works essentially the same way as it has since SQL Server 6.5, providing availability through failover based on the underlying Windows cluster. However, how it is deployed has changed. Both the upgrade process and the installation process are completely different than the setup processes in SQL Server 2000 or SQL Server 2005. Instead of having all nodes configured during one session, in SQL Server 2008, each instance on every node must be installed or upgraded separately. Thus, for a new installation, there is one main install for the SQL Server instance, and everything else is an add-node operation. For an in-place upgrade, the process is similar, as described later in this section.

Table 4-1 shows an example of how many install processes would be executed for up to four nodes and four instances of SQL Server.

Table 4-1: Number of Installs or Upgrades per Node Depending on the Number of Instances Required

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **1 Instance** | **2 Instances** | **3 Instances** | **4 Instances** |
| 2 Nodes | 2 | 4 | 6 | 8 |
| 3 Nodes | 3 | 6 | 9 | 12 |
| 4 Nodes | 4 | 8 | 12 | 16 |

These numbers might be surprising to some, but the installation change was implemented with these important goals in mind: increased stability, improved reliability, and more granular control. In terms of an in-place upgrade, the ability to upgrade instances in a rolling fashion, one node at a time, increases application and database availability and minimizes downtime during an upgrade. Other nodes and their individual components as they relate to an instance can be upgraded independently.

Another major feature change is that if you are implementing a new failover clustering instance under Windows Server 2008, SQL Server 2008 does not require the domain groups that were introduced with SQL Server 2005. SQL Server 2008 under Windows Server 2008 can use Service SIDs, which are recommended. In-place upgrades (either Windows Server 2003 or Windows Server 2008) and all failover clustering deployments under Windows Server 2003 will still use domain groups. For information about Service SIDs, see Cyril Voisin's [Per-service SID](http://go.microsoft.com/fwlink/?LinkId=132188) Security blog entry.

A welcome addition to failover clustering functionality is that not only can you do setup via the command line or an .ini file, as you can do in SQL Server 2005, but you can also now perform the upgrade via the command line or an .ini file. This upgrade process is described in "Using the Command Line to Upgrade a Failover Clustering Installation," later in this chapter.

### Instance IDs and Failover Clustering

During an in-place upgrade to SQL Server 2008, Setup will prompt for the Instance ID on the Instance Configuration screen, which is new in SQL Server 2008. (See Chapter 1 for information about the new Instance ID and paths that SQL Server 2008 uses.) The Instance ID will default to MSSQLSERVER for a default instance and to the named part of a named instance for named instances. For example, if the instance name is MY\INS1, the Instance ID value will be INS1. The value used for the Instance ID must be the same for that particular instance across all nodes of the cluster. Because a clustered instance is not the same as its underlying server, it might make more sense to rename the Instance ID to the actual name of the instance or to the first part of the named instance.

### Windows Server 2008 and SQL Server 2008 Failover Clustering

If you want to upgrade the operating system on the cluster nodes to Windows Server 2008, you will face some challenges. The biggest challenge is that a rolling upgrade (that is, an upgrade performed one cluster node at a time) is not supported as it was with Windows Server 2003. With Windows Server 2008, the operating system-level clustering feature that SQL Server is built on was completely recoded, so a direct upgrade is not possible.

To reuse existing cluster nodes, you must install a fresh copy of Windows Server 2008, meaning that the existing configuration will be erased. You can then reinstall SQL Server after the installation of Windows Server 2008 is complete. This means that before tearing down the old configuration, performing backups of all databases and scripting all objects is essential. Although these tasks are part of a normal upgrade process, there is additional pressure to do them because the old configuration will no longer exist. If you are going to deploy Windows Server 2008, consider purchasing new hardware and implementing an upgrade strategy that would allow the configuration of the existing cluster to remain.

Be aware that with Windows Server 2008, Windows is calling the availability clustering a failover cluster, just as SQL Server does. Before Windows Server 2008, the operating system-level feature was known as a server cluster. Every attempt will be made to distinguish the difference between Windows and SQL Server in this portion of the upgrade guide.

**Important:** You can find more information about upgrading a Windows cluster at the [Migration Options for Hardware with Failover Clustering in Windows Server 2008](http://go.microsoft.com/fwlink/?LinkId=132442) Microsoft Failover and Network Load Balancing Clustering Team blog entry. However, note the following caveat. The blog talks about using the Migrate a Cluster Wizard. This Wizard *cannot* be used to migrate your existing clustered SQL Server instances. The Migrate a Cluster Wizard can be used to assist in the migration or upgrade of the operating system-level cluster. You will still need to do a full installation of SQL Server and then upgrade the databases either using a database attach if the files are on the SAN LUNs associated with the cluster or using backup restores. Do not try "disabling" SQL Server, unclustering, upgrading the operating system, and then trying to re-enable SQL Server. Follow the supported upgrade paths that are outlined in this document and its links. Do not put yourself in a position where you will have a hard time getting support if you run into problems.

#### SQL Server 2008 Failover Clustering: Compatibility with Windows Server 2008 Failover Clustering Features

Windows Server 2008 provides a mixture of new and existing features for failover clustering, but SQL Server 2008 does not support them all.

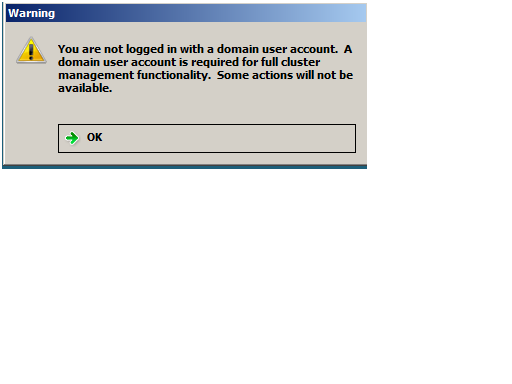
**Windows Server 2008 Failover Clustering Features**

* Windows Server 2008 failover clusters are no longer bound by the old Windows Server Catalog and the cluster solutions defined in them. Windows Server 2008 has a built-in process called Cluster Validation. The concept is simple: if the hardware passes the tests, a cluster can be configured and is considered supported. If the validation fails, the cluster is not considered a supported or valid cluster. It is possible to deceive Cluster Validation because some tests can be disabled; do not do this. We strongly recommend that you run all tests, especially in a production environment. Do not take this capability as carte blanche to take two or more odd pieces of hardware and cobble them together as a valid cluster. The recommendation is still to configure nodes that are similar (i.e., same brand and type of server). The biggest change to implementers is that they will have to rely on the other hardware vendors (SAN, HBA) to provide correct information about which drivers are certified for clusters using Windows Server 2008.

Even though you still need to run Cluster Validation, Microsoft offers a list of vendor pre-certified cluster solutions through the [Failover Cluster Configuration Program](http://go.microsoft.com/fwlink/?LinkId=132443).

**Important:** It is crucial to ensure that the Windows Server 2008 failover cluster has no errors during Cluster Validation. SQL Server 2008 Setup relies on the validation results. If an error is found, Setup will not proceed. If you have deployed Windows Server 2008 without Hyper-V RTM, you might encounter an error during cluster validation that says the operating SKU is not valid, as described in [When you run the Validate a Configuration Wizard on a Windows Server 2008-based computer or on a Windows Vista-based computer, the validation does not pass](http://go.microsoft.com/fwlink/?LinkId=132444) in the Microsoft Knowledge Base. A fix for this issue is available for download. After applying the patch, re-run Cluster Validation.

* Windows Server 2008 failover clustering supports separate subnets for cluster nodes. SQL Server does not support this feature, so if you want to do geographically dispersed clusters, you must implement them the traditional way—by using a virtual LAN (VLAN).
* The quorum is now known as the witness, and there are four models (up from two). They are:
  + No Majority. This is the same as the old disk-based quorum, where the quorum disk is a single point of failure.
  + Node Majority. This is the same as the old Majority Node Set quorum, where one less than half the number of nodes rounded up can be tolerated as a failure. So for example, two nodes have no failure tolerance; three and four nodes can tolerate one node failure; and five nodes can tolerate two nodes failing. Node majority is best for an odd number of nodes.
  + Node and Disk Majority. This is a combination of both the witness disk and a majority of nodes, giving you more protection. It can tolerate the failure of half the nodes rounded down if the witness disk is available, and it can tolerate the failure of one less than half the nodes rounded up if the witness is unavailable. So for example, two nodes have one-node failure tolerance if the witness is available, but no failure tolerance if the witness is unavailable.
  + Node and File Share Majority. Similar to the Node and Disk Majority, this uses a file share instead of a witness disk. This is a good choice for a geographically dispersed cluster.
* To cluster the Microsoft Distributed Transaction Coordinator (DTC), you must configure the Application Server role on each cluster node.
* All cluster nodes must be in the same domain to implement SQL Server 2008 with Windows Server 2008.
* The private network, also known as the heartbeat, is now unicast.
* All disks must still be Basic disks (not Dynamic); however, GPT disks are supported for sizes over 2 TB.
* Cluster Administrator has been completely revamped. It is now a Microsoft Management Console (MMC) snap-in and has been renamed Failover Cluster Management.
* To properly install a Windows Server 2008 failover cluster, a domain account with local administrator rights must be designated on each node with rights to create objects in the domain. The domain account after installation is used only for administrative purposes and is no longer used to run the Windows failover cluster itself. If Failover Cluster Manager is not started with that domain account, you will see the dialog box that Figure 4-2 shows.



**Figure 4-2:** Attempting to use Failover Cluster Management with a non-domain account

**SQL Server 2008 Failover Clustering Features**

* New SQL Server 2008 Enterprise failover clustering installations support up to 16 nodes with Windows Server 2008 failover clustering; SQL Server 2008 Standard still supports only two-node clusters.
* Drive letters are still required for all clustered SQL Server instances. Although using mount points for clusters was supported even by Windows Server 2003, SQL Server 2008 still only supports mount points on clustered SQL Server instances if they are associated with a drive letter. This is slated to be fixed in a future version of SQL Server.
* SQL Server 2008 supports the new DHCP functionality of Windows Server 2008 failover clustering. But we recommend that for predictability, you use a static IP address for SQL Server.

### Considerations for Upgrading a SQL Server 2000 Failover Cluster to SQL Server 2008

Because many SQL Server 2000 installations are old and the hardware might not be viable, the easiest method for upgrading a SQL Server 2000 failover cluster to SQL Server 2008 is most likely a side-by-side upgrade to a new cluster. Performing an in-place upgrade is possible only if the hardware is viable and you are using Windows Server 2003. For information about an in-place failover clustering upgrade, see the section "Upgrading a SQL Server 2000 or SQL Server 2005 Failover Cluster to SQL Server 2008" later in this chapter as well as "Additional References for Clustering Upgrades."

A side-by-side upgrade requires additional dedicated cluster disks and enough disk space for the SQL Server 2008 implementation (including space for the backups or other method used to initialize the data), a new IP address for the new SQL Server 2008 failover clustering instance, and a new instance name. For more information about how to install a new SQL Server 2008 failover cluster, see "Additional References for Clustering Upgrades" later in this chapter.

All new implementations under Windows Server 2003 will require the configuration of the domain service account and cluster groups, as mentioned earlier. SQL Server 2000 did not require these, so an in-place upgrade from SQL Server 2000 to SQL Server 2008 failover clustering will require their configuration as well. These features must be configured properly before the upgrade begins; for more information, see [Before Installing Failover Clustering](http://go.microsoft.com/fwlink/?LinkId=132446) in SQL Server 2008 Books Online.

**Important:** If you implemented the IA64 version of SQL Server 2000 Enterprise on Windows 2000 Advanced Server Limited Edition, there is no direct upgrade path to SQL Server 2008. You will need to install a fresh installation of Windows Server 2003 or Windows Server 2008, install SQL Server 2008, and then use one of the standard methods such as database attach or a restore to upgrade the databases. This limitation is documented in [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkId=132447) in SQL Server 2008 Books Online: "Upgrade of SQL Server 2000 (64-Bit) IA64 failover clusters is not supported."

Upgrade of SQL Server 2000 Analysis Services to SQL Server 2008 is also not supported on failover clusters. SQL Server 2000 never supported the clustering of SSAS 2000 as a feature of the installer. [How to cluster SQL Server 2000 Analysis Services in Windows 2000 and in Windows Server 2003](http://go.microsoft.com/fwlink/?LinkId=132448) in the Microsoft Knowledge Base, describes how to cluster SQL Server 2000 Analysis Services by creating a generic resource in the Windows server cluster. The clustering process is done after the initial installation. SQL Server 2005 is the first version of SQL Server that provided as part of the Setup process the ability to cluster SQL Server Analysis Services.

#### Upgrading SQL Server Failover Clusters That Use WOW64

SQL Server 2008 does not support installing a 32-bit failover clustering instance under the Windows On Windows 64 (WOW64) Edition of 64-bit Windows. In-place upgrade of a WOW64-based SQL Server 2000 or SQL Server 2005 failover cluster to SQL Server 2008 is not supported. Therefore, to upgrade to SQL Server 2008 if a WOW64-based SQL Server 2000 or SQL Server 2005 failover cluster is deployed, you must use a side-by-side approach.

### Considerations for Upgrading a SQL Server 2005 Failover Cluster to SQL Server 2008

There are no specific considerations when upgrading from a SQL Server 2005 failover cluster to SQL Server 2008.

### Upgrading a SQL Server 2000 or SQL Server 2005 Failover Cluster to SQL Server 2008

The process for upgrading a SQL Server 2000 or SQL Server 2005 failover clustering instance to SQL Server 2008 is the same. You can perform the upgrade via the command prompt or through the graphical Setup application. Step-by-step instructions for using the Setup program for an in-place upgrade can be found in [How to: Upgrade a SQL Server Failover Cluster Instance (Setup)](http://go.microsoft.com/fwlink/?LinkId=132450) in SQL Server 2008 Books Online. This section walks through the main steps of an in-place upgrade but does not cover every Setup screen. This section also covers the tasks to perform and considerations to take into account before, during, and after upgrade, especially those not already documented or that might need further context.

To install a new clustered instance, follow the instructions in [Installing a SQL Server 2008 Failover Cluster](http://go.microsoft.com/fwlink/?LinkId=132470) in SQL Server 2008 Books Online.

**Important:** The *Advanced cluster preparation* option in the SQL Server Installation Center is for use only with new installations, not in-place upgrades, and will not be covered in this chapter.

#### Before Upgrading (Setup or Command Prompt)

1. Make sure the underlying Windows Server 2003 or Windows Server 2008 cluster is properly configured to standard best practices. The Windows Server 2003 solution must appear in the [Windows Server Catalog](http://go.microsoft.com/fwlink/?LinkID=71739) under the [Cluster Solutions category](http://www.windowsservercatalog.com/results.aspx?bCatID=1291&cpID=0&avc=10&OR=1). If it is not there, it is not a supported cluster solution for Window Server 2003.

If this is Windows Server 2008 cluster, the cluster must pass Cluster Validation; otherwise, SQL Server 2008 Setup will fail because it relies on a successful outcome of the Windows configuration files.

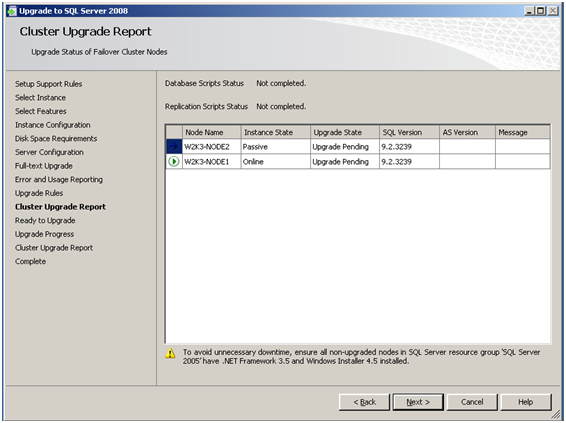
1. Make sure that the MS DTC is configured properly. It should not be configured in the group that contains SQL Server, and it also should not be placed in the group with the quorum or witness disk.
2. Create the security accounts and, if necessary, groups in the domain for SQL Server 2008. Because it might not be possible to alter the service accounts during the upgrade, we recommend that you change the SQL Server 2000 and SQL Server 2005 service accounts to SQL Server 2008-specific ones after upgrade.
3. Make sure that there are no errors, that all resources can be failed over to each node, and that all names and IP addresses can be accessed. Troubleshooting a poor Windows installation is more difficult after SQL Server is added to it.

#### In-Place Upgrade for a Single-Instance Failover Cluster

1. On the node that is not hosting the SQL Server 2000 or SQL Server 2005 instance, install both .NET Framework 3.5 SP1 and MSI 4.5 because this node can tolerate a reboot and will not affect the SQL Server instance. These installs can be done at any point leading up to the upgrade of SQL Server. Consider upgrading the shared components, but be aware that this will represent a major difference in underlying connectivity in the event a failover. If this might introduce risk to the cluster, upgrade the shared components only when the actual upgrade is performed.
2. Before upgrading the instance itself, upgrade the node(s) not hosting the SQL Server instance either via Setup or the command prompt (see "Using the Command Line to Perform an In-Place Upgrade for a Failover Clustering Installation," later in this chapter). This node must be upgraded first so that during the actual instance upgrade, the instance can be failed over to this node. Although this node is not running the clustered instance of SQL Server, it will upgrade *all* components on that server.

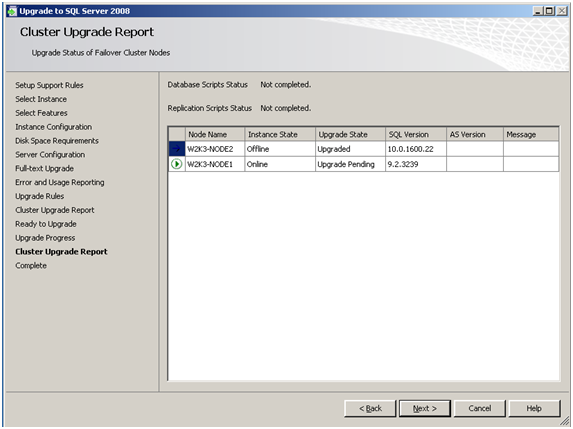
**Important:** When you get to Step 11 of [How to: Upgrade a SQL Server Failover Cluster Instance (Setup)](http://go.microsoft.com/fwlink/?LinkID=132450) in SQL Server 2008 Books Online, make sure to note the instance ID that is used (assuming the default is chosen) because it must be the same for each upgrade run on every node for the running SQL Server instance. If the instance ID is changed according to "Instance IDs and Failover Clustering," earlier in this chapter, make sure you use that value during the upgrade of each node.

1. Before the upgrade begins, Setup will display the Cluster Upgrade Report, as shown in Figure 4-3.



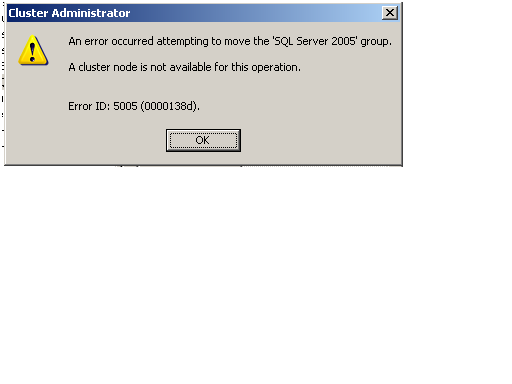
**Figure 4-3:** Cluster Upgrade Report

After the upgrade of the node is complete, the Cluster Upgrade Report will be displayed again with an updated status, as shown in Figure 4-4.



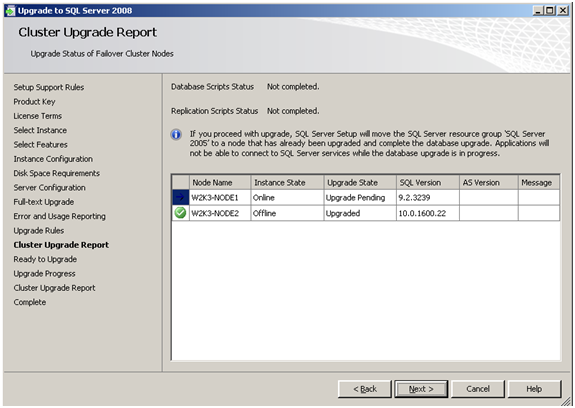
**Figure 4-4:** Updated Cluster Upgrade Report

**Warning:** At this point, it is not possible to fail over the existing SQL Server 2000 or SQL Server 2005 instance to the node that was upgraded. For two-node clusters, if the instance itself is not upgraded immediately and the instance fails without any nodes for it to fail over to, a bigger outage could occur. Time the upgrades of each node to minimize exposure to this risk. Figure 4-5 shows what happens if an instance attempts to fail over to an upgraded node.

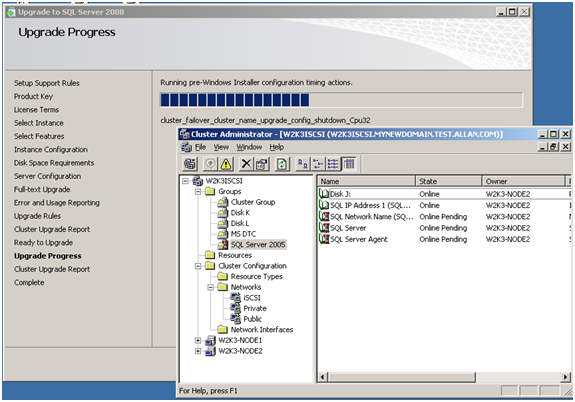


**Figure 4-5:** Warning after trying to fail the resource group

1. Stop all traffic to the instance of SQL Server to ensure that no one tries to access the instance during the upgrade process. SQL Server will be unavailable during the duration of the upgrade. Sometimes there is no need to stop the traffic. Setup will automatically handle the failover to the upgraded node. The only downtime for the clustered instance will be failover time + upgrade script execution time. One the average his takes about 2 minutes depending on the hardware and other configurations.
2. Follow the instructions in [How to: Upgrade a SQL Server Failover Cluster Instance (Setup)](http://go.microsoft.com/fwlink/?LinkID=132450) to upgrade the failover clustering instance. Use the same instance ID that was used for upgrading the other node(s). During the upgrade, the instance will be failed over to the already upgraded node, as the notification in Figure 4-6 shows. Figure 4-7 shows the failover during the upgrade. Note that the full-text search resource has already been removed from the resource group.

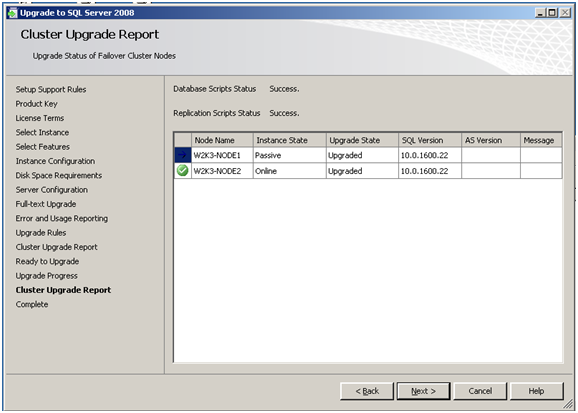


**Figure 4-6:** Uncompleted Upgrade Cluster Upgrade Report when the instance itself is upgraded



**Figure 4-7:** Showing the failover during the upgrade

When the upgrade is complete, Setup will show a final Cluster Upgrade Report, similar to the one in Figure 4-8.



**Figure 4-8:** Final Cluster Upgrade Report after success

The upgrade is now complete. Note that the upgraded instance was not automatically moved back to the original node. You can manually fail the resource group back to the original node by using the proper Windows cluster administration tool for the operating system version deployed.

1. Check for pending reboots and any errors in the logs to see if a reboot is necessary on the node. Reboot as needed.
2. Manually test failover of the resource group containing the SQL Server resource between all nodes of the cluster.
3. Ping the network name of the clustered SQL Server instance from all nodes inside the cluster as well as outside the cluster.
4. Ping the IP address of the clustered SQL Server instance from all nodes inside the cluster as well as outside the cluster.
5. Make sure that the compatibility level of the databases is set to 100.
6. We recommend that you run all the necessary health checks including DBCC CHECKDB to ensure the well-being of the newly upgraded databases.
7. Open the instance for testing and, if all goes well, production use.

#### In-Place Upgrade for a Multiple-Instance Failover Cluster

Upgrading a cluster that has multiple instances is more challenging because it requires more coordination. Each instance will be affected by an outage at some point, even if it is not being upgraded, and will be left alone in a side-by-side configuration with SQL Server 2008. This section walks through an example of a cluster with multiple instances of SQL Server.

The cluster in this example has three Windows Server 2003 nodes (A, B, and C) and three instances of SQL Server 2005 (1, 2, and 3). Node C is a dedicated failover node. The diagram in Figure 4-9 illustrates this cluster configuration.



**Figure 4-9:** Pre-upgrade cluster configuration

Because there are three instances of SQL Server, the upgrade process will be executed nine times (three per node). The real question is, outside of coordinating with business and application owners as well as other IT staff involved, how should you approach the upgrade? Remember that SQL Server 2005 and SQL Server 2008 can coexist side-by-side in the same cluster, so even if only one or two instances are upgraded, the remaining one or two SQL Server 2005 instances will still work.

The easiest thing to do first is to upgrade the shared components and install MSI 4.5 and .NET Framework 3.5 SP1 on Node C. Because Node C is the dedicated failover node, no outages will be incurred for any of the SQL Server instances themselves. And it will not pose any major risk if one of the instances needs to fail over to Node C after reboots because the actual upgrades have not been started. If both Node A and Node B have scheduled maintenance windows, those would also be opportune times to install at least MSI 4.5 and .NET Framework 3.5 SP1, and possibly upgrade the shared components. Remember to fully test even this partial upgrade of components elsewhere to ensure that there will be no incompatibilities with applications. If you cannot install MSI 4.5 and .NET Framework 3.5 SP1 during a maintenance window, you can do so during the actual upgrade of SQL Server.

At this point, you must choose an instance to upgrade based on whatever factors you are using to determine which instance goes first. For this example, assume that we want to upgrades the SQL 1 instance first. Upgrade the SQL 1 instance's components on Node C first and reboot that node if necessary. At this point, Node C is not an option for SQL 1 to fail over to. SQL 2 and SQL 3 can still fail over to Node C with no problems.

Next, upgrade SQL 1 on Node B. Here is where things get a bit sticky. SQL 2 and SQL 3 also exist on Node B. Consider coordinating a scheduled failover to Node C for the SQL 2 and SQL 3 instances so that the installation can occur. This failover would only mean minutes of downtime for those two instances and most users and applications would not even notice. With this strategy, if something goes wrong with the installation, SQL 2 and SQL 3 will not unexpectedly fail over, and the install process can occur and not affect any processes on Node B. This approach also assumes that Node C has the capacity to run the instances simultaneously. There should be no need at this point to schedule any failovers for SQL 2 or SQL 3 and disrupt business again. Assuming the failover is complete, the configuration at this stage now looks like the one in Figure 4-10.



**Figure 4-10:** SQL 1 upgraded on Nodes B and C but not yet on Node A

Finally, upgrade SQL 1 on Node A. After the instance is upgraded, it can fail over to either Node B or Node C. The cluster now has one SQL Server 2008 instance (SQL 1) and two SQL Server 2005 instances (SQL 2 and SQL 3) peacefully coexisting. For this example, let's say that SQL 1 failed over to Node B; the configuration now looks like the one in Figure 4-11.



**Figure 4-11:** Upgrade of SQL 1 is complete

The cluster configuration can stay this way indefinitely, or if you need to, you can upgrade SQL 2 and SQL 3 to SQL Server 2008. The considerations and technical processes will be the same as the ones for SQL 1: You need to determine who the upgrade affects, how to minimize risk and downtime for the other instances, and the correct set of steps to get the work done.

In our example, let's next upgrade SQL 2, which has been coexisting on Node C with SQL 3 for a few weeks while SQL 1 remains on Node B. All the nodes have .NET Framework 3.5 SP1 as well as the shared components, so this upgrade should be more straightforward and incur fewer overall outages. First, upgrade SQL 2 on Node A and reboot if necessary. Next, schedule a planned failover of both SQL 1 and SQL 3 to Node A. The configuration will look like the one in Figure 4-12.



**Figure 4-12:** Partial upgrade of SQL 2

Upgrade SQL 2 on Node B and then on Node C, which upgrades the SQL Server instance. Assuming that SQL 2 fails over to Node B, the configuration after the successful upgrade of SQL 2 looks like the one in Figure 4-13.



**Figure 4-13:** Completed upgrade of SQL 2

Finally, it is time to upgrade SQL 3. First upgrade SQL 3 on Node C. Schedule a failover of SQL 1 and SQL 2 to Node C and upgrade Node B. Last, upgrade SQL 3 on Node A. The final configuration would look like that in Figure 4-14, assuming a failover to Node B of SQL 3 after its upgrade.



**Figure 4-14:** All three instances of SQL Server upgraded

If everything is working, do not fail the instances back to their original configuration, which Figure 4-9 above shows. But if you want to fail back to the original configuration, schedule an outage to accomplish this task.

#### Side-by-Side Upgrade in the Same Cluster or to a New Cluster

Performing a side-by-side upgrade—whether on the same cluster where the current instance lives or using a newly deployed Windows cluster—follows basically the same steps as an in-place upgrade. Use one of the methods listed in "Methods for Side-by-Side Upgrades to a Separate Server or Cluster" to move and upgrade the databases in conjunction with installing a fresh instance of SQL Server 2008. Remember that with failover clustering, the new clustered instance name must be completely unique in the domain. This means that after the upgrade, all users and applications will need to be redirected to the new instance.

#### Using the Command Prompt to Perform an In-Place Upgrade for a Failover Clustering Installation

This section provides the syntax for upgrading an instance via the command-line Setup. It is assumed that.NET Framework 3.5 SP1 and MSI 4.5 are already installed. There are two options for using the command line: either enter all the commands directly on the same line as setup.exe or use a configuration file. This section covers only using setup.exe. Table 4-2 shows the command-line Setup upgrade options for failover clustering.

Table 4-2: Command-Prompt Setup Upgrade Options for Failover Clustering

| **Option** | **Value** | **Notes** |
| --- | --- | --- |
| ACTION | Upgrade | This tells setup.exe that an upgrade will be performed. |
| AGTDOMAINGROUP | This is the name of the domain-based group associated with SQL Server Agent. | This is used for SQL Server 2000 upgrades because SQL Server 2000 did not require domain groups and SQL Server 2008 does. |
| FAILOVERCLUSTERROLLOWNERSHIP | 0 = Will not roll ownership to already upgraded nodes, nor will it add the node to the list of possible owners when upgrade is completed.  1 = Will roll ownership to upgraded nodes  2 = SQL Server determines if 0 or 1 is needed. | This is optional. If not specified, the default value will be 2. The parameter controls the failover behavior of the upgrade. |
| FTSVCACCOUNT | This is the account used for the SQL Full-text Filter Daemon Launcher. |  |
| FTSVCPASSWORD | A valid password for the account specified FTSVCACCOUNT. | This will be in plain text. |
| FTUPGRADEOPTION | Import – Import existing full-text catalogs instead of rebuilding them.  Rebuild – Rebuild the full-text catalogs using the new word breakers.  Reset – Reset the full-text catalogs. | These are the same values as on the Full-text Upgrade screen of the GUI. |
| INSTANCEID | Valid value for the Instance ID. See the earlier section "Instance IDs and Failover Clustering" for an explanation and recommendations. | If a value is not specified, it will default to MSSQLSERVER for a default instance and the value after the slash for a named instance (i.e., if the instance name is MY\INS1, the value will be INS1). |
| INSTANCENAME | Valid value for the instance to be upgraded. | Use MSSQLSERVER for a default instance and the value after the slash for a named instance (i.e., if the instance name is MY\INS1, the value will be INS1). |
| INDICATEPROGRESS | TRUE shows a verbose output that can also be redirected to a text file.  FALSE suppresses any messages. | This is optional. If it is not set, INDICATEPROGRESS defaults to FALSE. See Figures 15 and 16 for examples of what successful completed upgrades look like with both options. Setting to TRUE will be invaluable to most, especially because at the end, it indicates if a reboot is needed or not. |
| Q(UIET) | N/A | This tells setup.exe to be "silent" and not display any of the GUI interface. |
| SQLDOMAINGROUP | This is the name of the domain-based group associated with the SQL Server service. | This is used for SQL Server 2000 upgrades because SQL Server 2000 did not require domain groups and SQL Server 2008 does. |

You can also use other options on the command prompt, such as ISSVCACCOUNT and RSSVCACCOUNT, which correspond to SQL Server Integration Services (SSIS) and SQL Server Reporting Services (SSRS), respectively. For more information, see [How to: Install SQL Server 2008 from the Command Prompt](http://go.microsoft.com/fwlink/?LinkId=132474) in SQL Server 2008 Books Online.

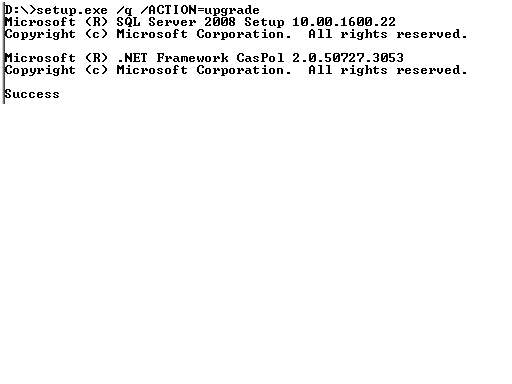
To use the command prompt, use a slash (/) before each option and separate with a space. For strings that might contain spaces or odd characters, use double quotation marks. Below is the sample syntax for upgrading a named instance of SQL Server 2005:

D:\>setup.exe /q /ACTION=Upgrade /INSTANCENAME=INS1 /INDICATEPROGRESS=True /FAILOVERCLUSTERROLLOWNERSHIP=2 /INSTANCEID=SQLUPGRADE /FTSVCACCOUNT="domainname\ftssvcaccount" /FTSVCPASSWORD="password" /FTUPGRADEOPTION=Reset

Below is the sample syntax for upgrading a default instance of SQL Server 2000:

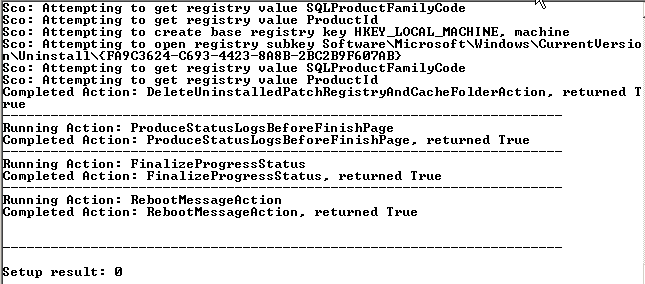
D:\>setup.exe /q /ACTION=Upgrade /AGTDOMAINGROUP="DOMAIN\Domain Group" /INSTANCEID=UPGRADEME /INSTANCENAME=INS1 /FAILOVERCLUSTERROLLOWNERSHIP=2 /FTSVCACCOUNT="domainname\ftssvcaccount" /FTSVCPASSWORD="password" /FTUPGRADEOPTION=Import /INDICATEPROGRESS=True /INSTANCEID=SQLUPGRADE /SQLDOMAINGROUP="DOMAIN\Domain Group"

As noted in Table 4-2, Figure 4-15 shows success for an example command-prompt upgrade if INDICATEPROGRESS is not set.



**Figure 4-15:** Example of a successful command-prompt upgrade of shared components only, with INDICATEPROGRESS not set

And Figure 4-16 shows an example of a successful command-prompt upgrade of a node when INDICATEPROGRESS is set to TRUE, indicating a reboot is necessary.



**Figure 4-16:** Example of a successful command-prompt upgrade of a node using INDICATEPROGRESS set to TRUE, indicating a reboot is necessary.

### Additional References for Clustering Upgrades

For an up-to-date collection of additional references for upgrading failover clusters—including Knowledge Base articles, blogs, white papers, and other resources—see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkId=132498).

## Upgrading Log Shipped Databases

This section covers how to upgrade databases that are using the built-in log shipping feature. This section assumes an in-place upgrade of the instance containing either the primary or secondary database.

### Feature Changes in SQL Server 2008 Log Shipping

Unlike in failover clustering, any objects that reside outside the database itself must be manually accounted for. This means that items such as SQL Server Agent jobs, SQL Server-level logins, linked servers, and anything not self-contained within the database must be either scripted or moved to the destination server in another way. Think of this process as a disaster recovery drill: you need to take the same steps to prepare for the upgrade as you would if preparing to perform a role change. See "Script or Export All Objects " above for information about getting objects out of SQL Server and how to perform such tasks as synchronizing logins.

Whether you want to upgrade a SQL Server 2000 database participating in log shipping or a SQL Server 2005 database participating in log shipping, if an upgrade occurs and the secondary database is left in either STANDBY or NORECOVERY mode, the secondary database will not be upgraded to a SQL Server 2008 database until it is brought online WITH RECOVERY.

SQL Server 2008 supports one new log shipping feature not in SQL Server 2005 log shipping: backup compression. Backup compression, available only in SQL Server 2008 Enterprise, by default is not enabled after the upgrade to SQL Server 2008. To enable it, go into the log shipping configuration and manually enable it on the primary database.

### Log Shipping Upgrade Scenarios

SQL Server 2000 and SQL Server 2005 have similar upgrade scenarios for log shipping. The two scenarios are upgrading with or without a role change. A role change is the log shipping-specific terminology for the process of switching the current primary database to promote the secondary database as the new primary.

**Note:** When upgrading without a role change and either preserving or reconfiguring log shipping, as long as the primary database and any secondary databases are at the same point, the secondary will be upgraded once log shipping is restarted or reconfigured. This works because the database upgrade is a logged operation: Once the transaction logs start flowing again to the secondary database, it will be upgraded as part of the log shipping process. This scenario greatly speeds up the upgrade process and reduces the number of steps required.

#### Upgrade with a Role Change

In this upgrade scenario for log shipping, a role change is performed to shorten downtime. When upgrading with a role change, the only time the database should be unavailable is during the actual switch from the primary database to the secondary database. In theory, the process sounds simple, but in reality, it is more complex.

When the instance containing the secondary database is upgraded, the secondary database will not be upgraded because it is still in a state to accept the loading of transaction log backups. This means that after the instance is upgraded, there is the ability to apply the transaction logs from the primary database. However, once the cutover is started, the primary database can no longer be used because there is no way to keep the two databases synchronized: when a database is brought online WITH RECOVERY, it can no longer accept transaction logs. In addition, after the secondary becomes the new primary, all users and applications would need to be redirected to the new instance and database. If that process is not centrally managed, many desktops would need to be touched to achieve the switch. Thus, this is not a recommended upgrade scenario.

At this point, although the old primary database could be upgraded, log shipping would no longer be configured. You would have to completely reconfigure log shipping, which means making a backup from the new primary and restoring it on the "new" secondary (the old primary). If the database is small, that might not be difficult. But if you are dealing with a VLDB or limited network bandwidth for copying a full database backup—not to mention the time it would take to restore—upgrading without a role change might be a better option.

On the plus side, besides reducing downtime, upgrading with a role change provides a built-in fallback plan. If the SQL Server 2008 upgrade does not work for any reason or is not accepted after a few days, the original instance and database would be nearly ready to go. You would just need to use a manual method of getting the updated data out of SQL Server 2008.

#### Upgrade without a Role Change

When upgrading without a role change, an upgrade is performed on an existing instance of SQL Server without doing the role change. This will preserve the state of the databases. But depending on the version of SQL Server you are upgrading (see the specific sections for SQL Server 2000 and SQL Server 2005 below), it might or might not upgrade log shipping in the process. This option generally has more downtime associated with it, but for those who cannot afford to reinitialize log shipping, upgrading without a role change might be a better option.

#### Unconfigure Log Shipping

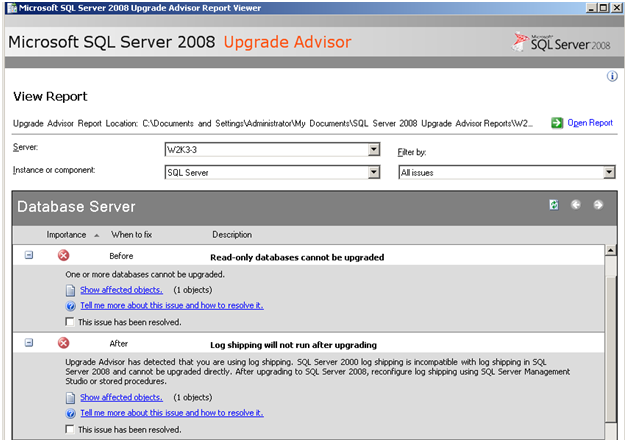
Similar to upgrading without a role change is the option of completely removing the log shipping configuration for a particular database. This would yield a result similar to upgrading without a role change. But in the case of SQL Server 2000, it might leave a cleaner upgrade, as we will discuss in the next section. Because a SQL Server 2008 transaction log cannot be applied to a SQL Server 2000 or SQL Server 2005 database, log shipping would need to be reinitialized.

### Upgrading from SQL Server 2000 to SQL Server 2008 Log Shipping

There is no direct upgrade path for log shipping configurations on a SQL Server 2000 database. However, you can migrate from SQL Server 2000 log shipping to SQL Server 2008 log shipping. Microsoft first introduced log shipping in SQL Server 2000 Enterprise. Log shipping was configured as part of the Database Maintenance Plan Wizard. Although it was mainly used to expose stored procedures triggered by SQL Server Agent jobs as well as xp\_sqlmaint, it could not be set up via stored procedures. When Microsoft released SQL Server 2005, it changed how log shipping could be configured, and log shipping was no longer integrated and associated with a database maintenance plan.

If the Log Shipping Monitor is configured on an instance other than the primary or secondary database, it can be migrated at any time because the functionality is not integral to the migration of the primary or secondary database. After you migrate the instance that is running the Log Shipping Monitor, be sure to remove the remnants of the SQL Server 2000 log shipping configuration.

Figure 4-17 shows the output of Upgrade Advisor for SQL Server 2000 if log shipping is configured.



**Figure 4-17:** Upgrade Advisor results

**Important:** The SQL Server 2000 secondary database that has been migrated to SQL Server 2008 must be reconfigured to be restored WITH NORECOVERY if it was configured to restore WITH STANDBY.

#### In-Place Migration with a Role Change

Follow these steps to perform a role change during the migration from SQL Server 2000 Enterprise to SQL Server 2008 when log shipping is involved.

1. Disable the copy and restore jobs on the secondary database.
2. Upgrade theinstance of SQL Server 2000 that contains the secondary database. Note that even though the instance will be upgraded to SQL Server 2008, the log shipped database is in a state where it is restoring transaction logs. Thus, the log shipped database will not be upgraded to SQL Server 2008 yet.
3. Manually copy over and restore (WITH NORECOVERY) all transaction log backups generated during the SQL Server 2008 upgrade from the primary database to the secondary.
4. On the primary database, stop all incoming traffic and verify that there are no connections to ensure that the data cannot be changed. To do this, consider using single-user mode.
5. On the primary database, make a final transaction log backup (also known as the tail of the log), using the WITH NORECOVERY clause of the BACKUP LOG command to place the database in a state where it can accept transaction logs. Do this only if you want to reconfigure log shipping or make this database the primary again. If not, just perform a normal transaction log backup and do not use the WITH NORECOVERY option.
6. Copy and restore the tail of the log WITH RECOVERY on the secondary database to bring it online. It is important that WITH RECOVERY is only used for this transaction log restore. At this point, the log shipped database will have been upgraded to SQL Server 2008.
7. Make sure the compatibility level of the database is set to 100.
8. Synchronize any logins and ensure that all jobs and objects residing outside the database are restored via scripts.
9. Delete any remnants of the old log shipping configuration on the secondary database.
10. It is recommended that you now run all the necessary health checks, including DBCC CHECKDB, to ensure the well-being of the newly upgraded databases.
11. Configure administration such as backup jobs for the database. It is especially important to configure transaction log backups.
12. Redirect all users and applications to the newly upgraded instance of SQL Server 2008 and database. The old secondary is now the new primary database.
13. Upgrade the instance that was formerly the primary and any other instance that might have been a secondary.
14. Remove all remnants of SQL Server 2000 log shipping from any databases that were involved the previous SQL Server 2000 log shipping topology, as noted in Table 4-3 later in this chapter.
15. If the old primary is going to become the new secondary or the primary again, configure log shipping from the new primary server to the old primary server. For more information, see [How to: Enable Log Shipping (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkId=132480) in SQL Server 2008 Books Online.
16. If the old primary is going to be the primary again and not a secondary, perform a role change back to that server. This is not recommended because it will cause another outage and increase downtime. For informaton about how to perform a role change with SQL Server 2008 log shipping, see [Changing Roles Between Primary and Secondary Servers](http://go.microsoft.com/fwlink/?LinkId=132481) in SQL Server 2008 Books Online.

#### In-Place Migration without a Role Change

Follow these steps to perform a migration from SQL Server 2000 to SQL Server 2008 without a role change.

1. Disable the copy and restore jobs on the secondary database.
2. Upgrade the instance of SQL Server 2000 that contains the secondary database. Note that even though the instance will be upgraded to SQL Server 2008, the log shipped database is in a state where it is restoring transaction logs. Thus, the log shipped database will not be upgraded to SQL Server 2008 yet.
3. Manually copy over and restore (WITH NORECOVERY) all transaction log backups that were generated during the SQL Server 2008 upgrade from the primary to the secondary.
4. At this point, there are two options: continue to manually copy and restore the transaction logs for a period of time or upgrade the instance containing the primary. This is when downtime will begin.
5. When you are ready to upgrade the primary, stop all incoming traffic and verify that there are no connections to ensure that the data cannot be changed. To do this, consider using single-user mode.
6. Make a final transaction log backup (also known as the tail of the log) on the primary. Copy and restore the tail to the secondary along with any remaining transaction log backups that should be restored before the tail of the log.
7. Upgrade the instance of SQL Server 2000 containing the primary instance. At this point, both the primary and secondary databases are at the same point in time, as well as upgraded to SQL Server 2008.
8. Make sure the compatibility level of the online and upgraded primary database is set to 100.
9. It is recommended that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
10. Remove any remnants of the old log shipping configuration on the primary and secondary.
11. Configure administration such as backup jobs for the primary database if necessary.
12. Reconfigure log shipping under SQL Server 2008 from the primary database to the secondary.

#### Migrate the Log Shipping Monitor

SQL Server 2000 allows the configuration of the Log Shipping Monitor on another instance of SQL Server that is not necessarily the primary or secondary. Treat this instance as if you were upgrading a typical SQL Server 2000 instance unless it is also the primary or secondary. Remember to remove the SQL Server 2000 log shipping objects from the monitor instance (refer to Table 4-3).

#### Side-by-Side Migration to a New Server or Cluster

There are two scenarios for a side-by-side upgrade: new hardware for both the primary and the secondary or new hardware only for the primary. Because secondaries can easily be initialized, those steps are not covered here.

**New Hardware for Both the Primary and Secondary**

1. Install the new SQL Server 2008 instances.
2. Take a full backup of the SQL Server 2000 primary database and restore two copies WITH NORECOVERY: one on the new primary and one on the new secondary.
3. When you are ready to upgrade, stop all traffic and kill all connections to the instance containing the original primary. This will ensure that the data cannot be updated during the upgrade.
4. If transaction logs were made after the full backup was taken, copy and restore them WITH NORECOVERY on both the primary and secondary.
5. Back up the tail of the log on the primary.
6. Copy the tail log backup to the new primary and restore WITH RECOVERY.
7. Copy the tail log backup to the new secondary and restore WITH NORECOVERY.
8. If necessary, synchronize any logins and ensure that all jobs and objects residing outside the database are restored via scripts.
9. We recommend that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
10. Set the compatibility level of the primary database to 100.
11. Configure SQL Server 2008 log shipping between the new SQL Server 2008 primary and secondary databases.
12. Redirect all users and applications to the new SQL Server 2008 instance containing the principal.

**New Hardware for the Primary Only**

1. Install the new SQL Server 2008 instance.
2. Take a full backup of the SQL Server 2000 primary database and restore the database WITH NORECOVERY on the new primary.
3. When you are ready to upgrade, stop all traffic and kill all connections to the instance containing the original primary. This will ensure that the data cannot be updated during the upgrade.
4. If transaction logs were made after the full backup was taken, copy and restore them WITH NORECOVERY on both the primary and secondary.
5. Back up the tail of the log on the primary.
6. Copy the tail log backup to the new primary and restore WITH RECOVERY.
7. Copy the tail log backup to the secondary and restore WITH NORECOVERY.
8. Upgrade the secondary in-place.
9. If necessary, synchronize any logins and ensure that all jobs and objects residing outside the database are restored via scripts.
10. We recommend that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
11. Set the compatibility level of the primary database to 100.
12. Configure SQL Server 2008 log shipping between the new SQL Server 2008 primary and secondary databases.
13. Redirect all users and applications to the new SQL Server 2008 instance containing the principal.

#### Post-Migration Tasks

There might be objects left behind by the old SQL Server 2000 log shipping configuration. If the objects shown in Table 4-3 exist, delete them.

Table 4-3: SQL Server 2000 Log Shipping Objects

| **Object** | **Location** | **Primary** | **Monitor** | **Secondary** |
| --- | --- | --- | --- | --- |
| Maintenance Plan | N/A | X |  |  |
| Transaction Log Backup Job for DB Maintenance Plan ‘<Plan Name>' | SQL Server Agent Jobs | X |  |  |
| Log Shipping Alert Job – Backup | SQL Server Agent Jobs |  | X |  |
| Log Shipping Alert Job – Restore | SQL Server Agent Jobs |  | X |  |
| Log Shipping copy for <instance>.<DBName>\_logshipping | SQL Server Agent Jobs |  |  | X |
| Log Shipping Restore for <instance>.<DBName>\_logshipping | SQL Server Agent Jobs |  |  | X |
| log\_shipping\_databases table | msdb | X | X | X |
| log\_shipping\_monitor table | msdb | X | X | X |
| log\_shipping\_plan\_databases table | msdb | X | X | X |
| log\_shipping\_plan\_history table | msdb | X | X | X |
| log\_shipping\_plans table | msdb | X | X | X |
| log\_shipping\_primaries table | msdb | X | X | X |
| log\_shipping\_secondaries table | msdb | X | X | X |

### Upgrading from SQL Server 2005 Log Shipping to SQL Server 2008 Log Shipping

Unlike SQL Server 2000, SQL Server 2005 and SQL Server 2008 use the same underlying methods for log shipping deployments. Log shipping in SQL Server 2005 was also expanded beyond Enterprise, so as long as the version that you are upgrading from is supported, the log shipping configuration can be left intact. For version information, see [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) [in](file:///C:\Documents%20and%20Settings\subon\Local%20Settings\Temporary%20Internet%20Files\OLKC3\in) SQL Server 2008 Books Online. Whether log shipping is left intact also depends on which upgrade path (with or without a role change) you choose.

Another difference from SQL Server 2000 is that both SQL Server 2005 and SQL Server 2008 support the WITH NORECOVERY clause of the BACKUP LOG Transact-SQL statement. This clause allows a database to be put in a state where it can receive transaction log backups after a full backup is made. Using this clause makes upgrades flexible, as detailed below.

### Upgrade log shipping with a role change

This option will make the former log shipped secondary database the new primary database. This results in a smaller outage window for users of the database system.

1. Disable the copy and restore jobs on the secondary.
2. Disable the alert job on the monitor (if it is used).
3. Upgrade the SQL Server 2005 instance containing the secondary database. As noted earlier, because the database is in a state where it is restoring transaction logs, it will not be upgraded to SQL Server 2008 yet.
4. Manually copy over and restore (WITH NORECOVERY) all transaction log backups generated during the SQL Server 2008 upgrade from the primary to the secondary.
5. On the primary, stop all incoming traffic and verify that there are no connections to ensure that, at this point, the data cannot be changed. To do this, consider using single-user mode.
6. Disable the transaction log backup job on the primary.
7. On the primary database, manually make a final transaction log backup (also known as the tail of the log) using the WITH NORECOVERY option. This will put the SQL Server 2005 database in the RESTORING state. If this instance will not be the primary again, do not use the WITH NORECOVERY clause of the BACKUP LOG command.
8. Copy and restore the tail WITH RECOVERY on the secondary database, which has not yet been upgraded. Once the restore is done and the database is online, it will be considered upgraded to SQL Server 2008.
9. Synchronize any logins and ensure that all jobs and objects residing outside the database are restored via scripts.
10. It is recommended that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
11. Configure administration such as backup jobs for the database.
12. Set the database compatibility level of upgraded databases to 100.
13. Delete the old copy and restore jobs on the former secondary database.
14. Upgrade the monitor instance to SQL Server 2008.
15. Delete the alert job from the monitor instance.
16. Upgrade the original primary to SQL Server 2008. Because there is a new primary, to save time during the upgrade, consider dropping the old primary database so that it will not go through the upgrade process.
17. If the old primary is going to become the new secondary, make a new full backup of the new primary database, copy it, and restore it on the old primary server.
18. Configure log shipping from the new primary to the new secondary.
19. Redirect all users and applications to the newly upgraded SQL Server 2008 log shipped primary database. The old secondary is now the new primary database.

### Upgrade log shipping without a role change

If you want to retain the original log shipping configuration, the following steps are applicable. Be warned that this upgrade path will cause an outage on the primary that you need to account for. If you choose this option, consider taking a longer single outage and performing the upgrade without failover.

1. Disable the copy and restore jobs on the secondary.
2. Disable the alert job on the monitor (if it is used).
3. Upgrade the SQL Server 2005 instance containing the secondary database. As noted earlier, because the database is in a state where it is restoring transaction logs, it will not be upgraded to SQL Server 2008 yet.
4. Upgrade the monitor instance to SQL Server 2008.
5. Manually copy over and restore (WITH NORECOVERY) all transaction log backups generated during the SQL Server 2008 upgrade from the primary to the secondary.
6. On the primary, stop all incoming traffic and verify that there are no connections to ensure that, at this point, the data cannot be changed. To do this, consider using single-user mode.
7. Disable the transaction log backup job on the primary.
8. Upgrade the instance containing the primary database.
9. It is recommended that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
10. Enable the existing Log Shipping backup, copy, and restore jobs that were previously disabled during the upgrade window. Verify each job is working properly. Note that when the first transaction log from the upgraded primary database is applied to the secondary database server, it will upgrade the log shipped secondary database to SQL Server 2008.
11. Verify the database compatibility level of upgraded databases is 100.
12. Direct all users and applications to the original primary database.

#### Side-by-Side Upgrade to a New Server or Cluster

There are two scenarios for a side-by-side upgrade from SQL Server 2005: new hardware for both the primary and the secondary or new hardware only for the primary. Because secondaries can easily be initialized, those steps are not covered here.

**New Hardware for Both the Primary and Secondary**

1. Install the new instances of SQL Server 2008.
2. Take a full backup of the SQL Server 2005 primary database and restore two copies WITH NORECOVERY: one on the new primary and one on the new secondary.
3. When you are ready for the upgrade, stop all traffic and kill all connections to the instance containing the primary. This will ensure that the data cannot be updated during the upgrade.
4. If transaction logs were made after the full backup was taken, copy and restore them WITH NORECOVERY on both the primary and secondary.
5. Back up the tail of the log on the primary.
6. Copy the tail log backup to the new primary and restore WITH RECOVERY.
7. Copy the tail log backup to the new secondary and restore WITH NORECOVERY.
8. If necessary, synchronize any logins and ensure that all jobs and objects residing outside the database are restored via scripts.
9. It is recommended that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
10. Set the compatibility level of the primary database to 100.
11. Configure log shipping.
12. Redirect all users and applications to the new SQL Server 2008 instance containing the principal.

**New Hardware for the Primary Only**

1. Install the new SQL Server 2008 instance that will be used for the primary.
2. Take a full backup of the SQL Server 2005 primary database and restore it to the new SQL Server 2008 instance WITH NORECOVERY.
3. When you are ready for the upgrade, stop all traffic and kill all connections to the instance containing the primary. This will ensure that the data cannot be updated during the upgrade.
4. If transaction logs were made after the full backup was taken, copy and restore them WITH NORECOVERY on the SQL Server 2008 instance. Also ensure that the secondary is caught up with its transaction log restores.
5. Back up the tail of the log on the primary.
6. Copy the tail log backup to the new primary and restore WITH RECOVERY.
7. Copy the tail log backup to the secondary and restore WITH NORECOVERY.
8. Upgrade the secondary to SQL Server 2008.
9. If necessary, synchronize any logins and ensure that all jobs and objects residing outside the database are restored via scripts.
10. It is recommended that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
11. Set the compatibility level of the primary database to 100.
12. Configure log shipping.
13. Redirect all users and applications to the new SQL Server 2008 instance containing the principal.

### Upgrading with Multiple Secondaries

The log shipping functionality in both SQL Server 2000 and SQL Server 2005 log shipping functionality allows you to configure multiple secondaries for a single primary, although this is not a common configuration. Of course, having multiple secondaries will complicate the upgrade scenario. With more than just the primary, secondary, and monitor involved in the configuration, you will need to consider the order in which the instances will be upgraded. The two main log shipping upgrade scenarios—with a role change or without a role change—still apply. If you have multiple secondaries and perform a role change, you will need to reinitialize every secondary; if the upgrade is performed without a role change, the secondaries should be fine.

### Additional References for Upgrading Log Shipping

For more information about upgrading log shipping, see the following topics in SQL Server 2008 Books Online:

* [Migrating a SQL Server 2000 Log Shipping Configuration to SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132489)
* [Upgrading SQL Server 2005 Log Shipping to SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132491)
* [How to: Enable Log Shipping (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkID=132480)
* [How to: Enable Log Shipping (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=132494)
* [How to: Add a Secondary Database (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkId=132495)
* [How to: Add a Secondary Database (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=132496)

For an up-to-date collection of additional references for upgrading log shipping, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkId=132498).

## Upgrading Mirrored Databases

This section covers how to upgrade instances of SQL Server 2005 to SQL Server 2008 where database mirroring is involved. This section does not apply to SQL Server 2000, which does not include database mirroring.

Database mirroring supports rolling upgrades. The order in which the instances of SQL Server are upgraded depends on the mode of database mirroring being used: high performance (asynchronous) or high safety (synchronous). This is further complicated if there is a Witness server with high-safety mode.

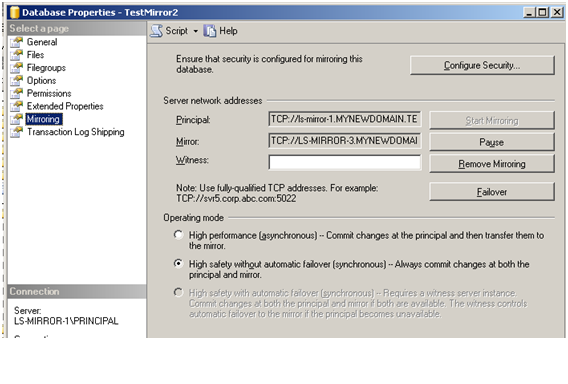
### Feature Changes in SQL Server 2008 Database Mirroring

Database mirroring works the way it did in SQL Server 2005 and still has a limit of one principal to one mirror. The two biggest changes are compression of the log stream between the principal and mirror and automatic page repair. Automatic page repair detects if either the principal or mirror has a corrupt page and fixes it from the side that is not corrupted. These new features are automatic and do not affect the upgrade process.

### In-Place Upgrade – High-Performance Mode

To upgrade a high-performance mode configuration, you must first change it to high-safety mode without a witness. This not only assures that quorum is not achieved during the upgrade, but it ensures that all transactions are committed and there is no data loss in the switch. After the upgrade is complete, you can switch the configuration back to high-performance mode. Below are the steps for an in-place upgrade in a high-performance mode configuration.

1. To change the mode from high performance to high safety, use SQL Server Management Studio or Transact-SQL. Figure 4-18 shows an example using the Database Properties page in SQL Server Management Studio.



**Figure 4-18:** Changing the mirroring mode in SQL Server Management Studio

The Transact-SQL command to perform the same task is as follows:

ALTER DATABASE database SET PARTNER SAFETY FULL

1. Upgrade the instance containing the mirror first and wait for the mirroring session to synchronize.
2. Stop all traffic into the Principal.
3. On the Principal, manually fail over to the upgraded mirror instance. At this point, the mirroring session will be suspended and will not resume until the Principal is upgraded.
4. Synchronize any logins and ensure that all jobs and objects residing outside the database are restored via scripts.
5. It is recommended that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
6. Configure administration such as backup jobs for the database.
7. Set the database compatibility level of upgraded databases to 100.
8. Redirect all users and applications to the new SQL Server 2008 instance containing the principal (the old mirror). This will minimize an outage because the principal will not be available during its upgrade.
9. Upgrade the instance containing the old principal (the new mirror). Database mirroring will be re-established because both the principal and mirror are at the same version.
10. To change the mode from high-safety to high-performance use SQL Server Management Studio or Transact-SQL. The Transact-SQL command to use is as follows:

ALTER DATABASE <database> SET PARTNER SAFETY OFF

### In-Place Upgrade – High-Safety Mode

1. If a Witness is configured, remove it. This will prevent a failover during the upgrade because the Principal will be unavailable during the upgrade of that instance.
2. Upgrade the instance containing the Mirror first and wait for the mirroring session to synchronize.
3. Stop all traffic into the Principal.
4. On the Principal, manually fail over to the upgraded Mirror instance. At this point the mirroring session will be suspended and will not resume until the Principal is upgraded.
5. Synchronize any logins and ensure that all jobs and objects residing outside the database are restored via scripts.
6. It is recommended that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded Principal database.
7. Configure administration such as backup jobs for the database.
8. Set the database compatibility level of upgraded databases to 100.
9. Redirect all users and applications to the new SQL Server 2008 instance containing the Principal (the old Mirror). This will minimize an outage because the Principal will not be available during its upgrade.
10. Upgrade the instance containing the Principal. Database mirroring will be re-established because both the Principal and Mirror are at the same version.
11. Finally, add the Witness back if it was configured.

### Side-by-Side Upgrade to a New Server

The following steps describe how to upgrade to SQL Server 2008 when you have database mirroring deployed and will be using a new server.

1. Install the new SQL Server 2008 instances.
2. Take a full backup of the SQL Server 2005 principal database and restore two copies WITH NORECOVERY: one on the new principal and one on the new Mirror.
3. When you are ready to upgrade, stop all traffic and end all connections to the instance containing the principal. This will ensure that the data cannot be updated during the upgrade.
4. If transaction logs were made after the full backup was taken, copy and restore them WITH NORECOVERY on both the principal and mirror.
5. Back up the tail of the log on the principal.
6. Copy the tail log backup to the new principal and restore WITH RECOVERY.
7. Copy the tail log backup to the new mirror and restore WITH NORECOVERY.
8. If necessary, synchronize any logins and ensure that all jobs and objects residing outside the database are restored via scripts.
9. It is recommended that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded database on the principal.
10. Configure administration such as backup jobs for the database.
11. Set the database compatibility level of upgraded databases to 100.
12. Configure database mirroring from the principal to the mirror.
13. Test the failover from the principal to the mirror and from the mirror back to the principal.
14. Redirect all users and applications to the new SQL Server 2008 instance containing the principal.

### Additional References for Upgrading with Mirrored Databases

For more information about upgrading with mirrored databases, see the following SQL Server 2008 Books Online topics:

* [How to: Manually Fail Over a Database Mirroring Session (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkId=132641)
* [How to: Manually Fail Over a Database Mirroring Session (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=132642)
* [How to: Minimize Downtime for Mirrored Databases When Upgrading Server Instances](http://go.microsoft.com/fwlink/?LinkId=132643)

For an up-to-date collection of additional references for upgrading with mirrored databases, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkID=132510).

## Upgrading Replicated Databases

This section covers upgrading databases and instances that are participating in replication. Unlike the other availability features, replication is generally upgraded with little additional effort during the upgrade of the instance to SQL Server 2008. This section covers both general and specific considerations for each form of replication.

### Feature Changes in SQL Server 2008 Replication

Upgrading replication can become complicated because, depending on the replication topology, there might be a lot of moving parts and dependencies. A simple replication topology is always easier to upgrade than one that has multiple Distributors, multiple levels of Publishers, and changes occurring at multiple locations. Using such capabilities as updating subscribers and peer-to-peer replication is not recommended because they allow changes to occur outside of the Publisher. When upgrading to SQL Server 2008, it might be time to consider instituting a ban on allowing anyone other than the Publisher to update data for a period of time. If this is not done, the Subscribers must be treated as Publishers.

SQL Server 2008 can use a SQL Server 2000 instance as part of its replication topology, but that instance must be at a minimum of SQL Server 2000 SP3a. If versions of SQL Server participating in the replication topology are older than SQL Server 2000 SP3a, they must be upgraded or they cannot participate in replication. All versions of SQL Server 2005 can participate in a replication topology with SQL Server 2008.

The Distributor is the key to how much downtime will be encountered during the upgrade. If there are multiple Distributors in the topology, there will be multiple outages. Where possible, assuming the hardware is not out of date, performing an in-place upgrade instead of installing a new instance of SQL Server and reconfiguring the Distributor is the preferred upgrade method. When upgrading in an environment that has multiple Distributors, upgrade them in order of magnitude: Do not upgrade the biggest and most important Distributor first. Schedule the upgrades to have the least impact on the end users and applications.

**Tip:** Always upgrade the Distributors first because they can push changes from a Publisher to a Subscriber that are both down level from the SQL Server 2008 Distributor. Ensure that the Publishers and Subscribers are at the right SQL Server patch levels; otherwise, upgrading the Distributor first will be pointless.

In general, performing in-place upgrade is preferred when it comes to all aspects of replication because much like log shipping, reinitializing a Subscriber can be very costly from a time perspective as well as in disk space, network bandwidth, and so on. If someone forgets to script out the replication or does not update the script, starting from scratch is not a good option. Avoid this if at all possible.

The Distributor's SQL Server version must be greater than or equal to the version of the Publisher. So if the Publisher is a SQL Server 2008 database, the Distributor must be SQL Server 2008. The Publisher version can be lower than the Distributor version.

If there are SQL Server CE Subscribers, additional actions must be executed on the IIS server. SQL Server 2008 client connectivity components must be installed along with SQL Server Compact components (SQL Server Compact 3.5 Service Pack 1 Server Tools) on the IIS server. Sqlcesa30.dlll, sqlcerp30.dll, and all the replication components on the IIS server must also be replaced.

SQL Server 2008 introduces some new data types, which will need to be mapped to compatible old data types if you will be replicating to or from earlier versions of SQL Server. For more information, see "Mapping New Data Types for Earlier Versions" in [Using Multiple Versions of SQL Server in a Replication Topology](http://go.microsoft.com/fwlink/?LinkId=132645) in SQL Server 2008 Books Online. Table 4-4 shows the mappings of old to new data types.

Table 4-4: Data Type Mappings

| **SQL Server 2000 Data Type** | **SQL Server 2005 Data Type** | **SQL Server 2008 Data Type** |
| --- | --- | --- |
| Image | Common Language Runtime (CLR) user-defined type (UDT) | UDT (8000 bytes or less) |
| Image | Varbinary(max) | UDT (8000 bytes or more) |
| Nvarchar(10) | Nvarchar(10) | Date |
| Nvarchar(27) | Nvarchar(27) | Datetime2 |
| Nvarchar(34) | Nvarchar(34) | Datetimeoffset |
| Not supported | Varbinary(max) | FILESTREAM attribute |
| Image | Varbinary(max) | Geography and geometry |
| Image | Varbinary(max) | Hierarchyid |
| Ntext | Nvarchar(max) | Nvarchar(max) |
| Nvarchar(16) | Nvarchar(16) | Time |
| Text | Varchar(max) | Varchar(max) |
| Image | Varbinary(max) | Varbinary(max) |
| Ntext | Xml | Xml |

If you are upgrading from SQL Server 2005, there are minor changes to replication from an upgrade perspective. If upgrading from SQL Server 2000, to see all aspects of replication, you must account for features that have been deprecated or discontinued as well as for breaking and behavior changes in SQL Server 2008. For comprehensive information about these changes, see [Replication Backward Compatibility](http://go.microsoft.com/fwlink/?LinkId=132646) in SQL Server 2008 Books Online.

#### Deprecated Features

SQL Server 2008 deprecates the PublisherAddress, PublisherNetwork, DistributorNetwork, and DistributorAddress parameters of the Distribution Agent. This applies to all forms of replication. We recommend using an alias with the client protocols to map the IP address to server name and then use the server name in the agents.

#### Behavior Changes

SQL Server 2005 implemented a new security model for replication. If you are using SQL Server 2005, you should already know about this change and have accounted for it. If you are upgrading from SQL Server 2000, see "New Replication Agent Security Model" in [Considerations for Upgrading Replicated Databases](http://go.microsoft.com/fwlink/?LinkId=132650) in SQL Server 2008 Books Online for information about the security model.

**Important:** Before any upgrade, always script the existing replication configuration. You should probably upgrade the scripts if they will be applied to SQL Server 2008. You will see errors in the execution of the scripts and, ultimately, failure if they are not upgraded and executed by anyone other than a sysadmin.

### Snapshot Replication

This section covers the specifics of upgrading snapshot replication to SQL Server 2008.

#### Changes to Snapshot Replication in SQL Server 2008

When using row and/or page compression with a SQL Server 2008 source, the Snapshot Agent will generate a schema that uses compression on the table and the indexes, not one or the other.

#### In-Place Upgrade from SQL Server 2000 or SQL Server 2005

As noted earlier, performing an in-place upgrade is the least intrusive way of upgrading all servers participating in snapshot replication.

1. Generate scripts for the entire existing SQL Server 2000 or SQL Server 2005 replication topology and store them in a safe place.
2. If the Distribution Agent is configured on the Subscriber for a pull subscription, disable the SQL Server Agent jobs related to the pull.
3. Upgrade the Distributor to SQL Server 2008. This may or may not be the same as the Publisher. Before upgrading the Distributor, disable any SQL Server Agent jobs related to replication, including any push subscriptions. Figure 4-19 shows an example of SQL Server Agent jobs involved in snapshot replication.

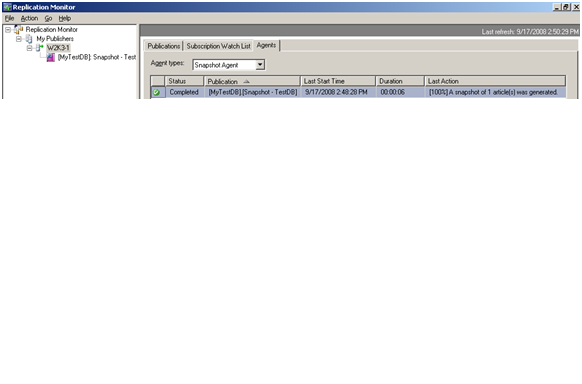


**Figure 4-19:** Distributor with push subscriptions for snapshot replication

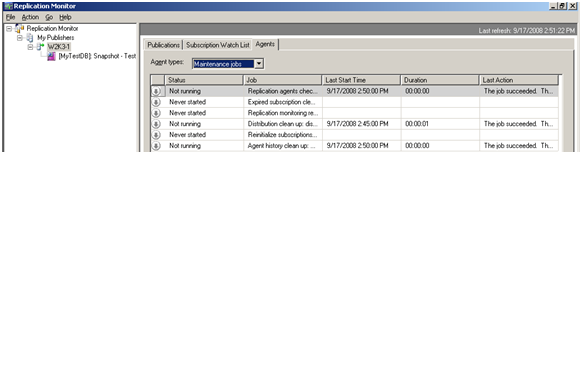
1. Ensure that each Subscriber is at a version of SQL Server that is compatible with SQL Server 2008 replication.
2. Check to see that SQL Server Agent is started on the Distributor.
3. If the Publisher is not the same as the Distributor and the Publisher or Subscriber will not be upgraded in the same outage, enable all of the replication jobs at the newly upgraded Distributor to allow replication to work until all components are upgraded. When the upgrade process is initiated for either the Publisher or Subscriber, it is recommended that you disable all SQL Server Agent jobs relating to distribution to ensure that no attempts to propagate data will occur during the upgrade.
4. At this point, as long as each Subscriber's version of SQL Server is compatible with SQL Server 2008, you can upgrade the Publisher. To upgrade the Publisher:
   1. Stop all traffic and end any connections into the Publisher.
   2. Upgrade the Publisher to SQL Server 2008.
   3. Ensure that SQL Server Agent is started.
   4. We recommend that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
   5. Set the database compatibility level of upgraded databases to 100.
   6. If the Subscriber will not be upgraded at this time, enable all the SQL Server Agent jobs related to the Publisher and Distributor.
5. If there are multiple Subscribers, consider doing a phased upgrade and do selected ones each outage. To upgrade the Subscriber, follow these steps:
   1. If the Subscriber is pulling the subscription, disable the SQL Server Agent jobs related to the subscription. This will ensure that the Subscriber cannot be updated during the upgrade process.
   2. Upgrade the instance containing the Subscriber to SQL Server 2008.
   3. Run DBCC CHECKDB against all upgraded databases.
   4. Set the database compatibility level of upgraded databases to 100.
   5. Ensure that SQL Server Agent is started.
   6. After the Subscriber is upgraded, enable all the SQL Server Agent jobs for the subscription and enable all SQL Server Agent jobs on the Distributor.
6. Verify that the replication is working properly. The easiest way to do this is to manually kick off the SQL Server Agent jobs involved in snapshot replication and view the status in the Replication Monitor. Figure 4-20, Figure 4-21, and Figure 4-22 show examples of a successful snapshot replication upgrade.



**Figure 4-20:** Overall status of snapshot replication



**Figure 4-21:** Status of the Snapshot Agent



**Figure 4-22:** Status of the SQL Server Agent jobs

1. Generate new scripts for the upgraded replication architecture.
2. The newly upgraded SQL Server environment is now ready for use by applications and end users.

#### Side-by-Side Upgrade to a New Server or Cluster

Snapshot replication tends to be the easiest in terms of a non-in-place upgrade because it is the least complex process.

1. In the replication topology, where an in-place upgrade of an existing instance to SQL Server 2008 will not be done, install a new instance of SQL Server 2008. This might be for a Publisher, a Distributor, or a Subscriber.
2. Generate scripts for the entire existing SQL Server 2000 or SQL Server 2005 replication topology.
3. Upgrade the scripts to SQL Server 2008, and make sure that the instance names and databases are updated to reflect their new locations. Before upgrading the scripts, make copies so that the old environment can be restored if necessary.
4. The Distributor must be SQL Server 2008. Run the appropriate upgraded replication script at the Distributor to create publication and distribution. At this point, the old replication topology is still in use.
5. To do the cutover, stop all traffic and kill any connections into the Publisher to ensure that no one tries to access it during the upgrade.
6. Assuming the Publisher is also going to be the main application database, upgrade it using one of the methods described in "Methods for New Hardware and Side-by-Side Upgrades," above. Just using a snapshot will not necessarily move the entire database. Also, ensure that moving the database to a new instance follows whatever approved and supported procedures the application vendor provides. Do not just move it.
7. Run the appropriate upgraded script at the Publisher.
8. Run the appropriate upgraded scripts at the Subscriber to connect it to the new replication topology. If the Subscriber is not a new instance of SQL Server 2008, make sure it is at a version of SQL Server that can be supported in a SQL Server 2008 replication topology. The Subscriber will need to be completely initialized. Remember to script and migrate any objects that are required by the database, but which are not accounted for by replication.
9. Verify that the replication agents and jobs are started with no errors.
10. Verify that transactional replication is working via the Replication Monitor.
11. Configure administration such as backup jobs for the database.
12. Generate new scripts for the upgraded replication architecture.
13. To ensure that no one will connect to the old environment, it is recommended that you stop the services if possible on the original topology after it has been moved to the new instance. For more guidance, see the section "Decommissioning and Uninstalling After a Side-by-Side or New Hardware Upgrade" earlier in this chapter.
14. The newly upgraded SQL Server environment is now ready for use by applications and end users. Point them to the new locations if necessary.

### Merge Replication

This section covers the specifics of upgrading merge replication to SQL Server 2008.

#### Changes

Merge replication uses the publication compatibility level to determine which features can be used. This compatibility level is set as the @publication\_compatibility\_level of sp\_addmergepublication on the Subscriber Types page of the New Publication Wizard or on the General page of the Publication Properties. Do not use any new SQL Server 2008-only features if there are down-level versions in the topology. An example of this is data compression. For more information, see "Compatibility Level for Merge Publications" in [Using Multiple Versions of SQL Server in a Replication Topology](http://go.microsoft.com/fwlink/?LinkID=132645) in SQL Server 2008 Books Online.

Table 4-5 lists the feature changes to merge replication and any recommended actions.

Table 4-5: SQL Server 2008 Merge Replication Changes

| **Feature** | **Description** | **Recommendation** |
| --- | --- | --- |
| "No sync" subscriptions to merge publications | Deprecated | A subscription is a "no sync" subscription if a value of none is specified for the @sync\_type parameter of sp\_addmergesubscription or sp\_addmergepullsubscription. This type of subscription is not recommended for merge replication. |
| -ParallelUploadDownload parameter¹ | Deprecated | This parameter of the Merge Agent is used to perform simultaneous upload and download of changes in a merge replication session. This parameter provides a performance gain, but it is outweighed by the amount of metadata that must be transferred over the network. |
| @allow\_partition\_realignment property in sp\_addmergepublication | Deprecated | This parameter is used to control the delete operations that must be sent to Subscribers if a row moves out of the Subscriber's partition. |
| -ExchangeType parameter | Deprecated | This parameter is used to control whether the Merge Agent goes through the upload phase or the download phase or both. This defaults to 3 to perform both upload and download. We do not recommend upload-only because it would not replicate schema changes or initialization processes. Download-only functionality can be achieved by using @subscriber\_upload\_options for an article. For more information, see [sp\_addmergearticle (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=132653) in SQL Server 2008 Books Online. |
| @delete\_tracking property in sp\_addmergearticle | Deprecated | This property is used to stop tracking deletes when deletes should be sent down to the Publisher or Subscriber. This can be implemented by using the DeleteHandler in the BusinessLogicModule. For more information, see [Executing Business Logic During Merge Synchronization](http://go.microsoft.com/fwlink/?LinkId=132654) in SQL Server 2008 Books Online. |
| Logical Records | Deprecated | This feature is used to send a set of related rows in a single transaction. In most cases, this feature adds significant performance overhead to replication when it is used. For more information, see [Grouping Changes to Related Rows with Logical Records](http://go.microsoft.com/fwlink/?LinkId=132655) in SQL Server 2008 Books Online. |
| PublisherAddress , PublisherNetwork, DistributorNetwork, and DistributorAddress parameters in Merge Agents | Deprecated | To initialize a subscription from a backup in SQL Server 2005. Use alias at the client protocols to map the IP address to server name and use the server name in the agent. |
| Initializing a transactional subscription from a backup | Breaking change | To initialize a subscription from a backup in SQL Server 2008, a user must be a member of the dbcreator server role. In SQL Server 2005, membership in the db\_owner database role was sufficient. For more information about how to initialize a subscription from a backup, see [Initializing a Transactional Subscription Without a Snapshot](http://go.microsoft.com/fwlink/?LinkId=132656) in SQL Server 2008 Books Online. |

#### Upgrading from SQL Server 2000 or SQL Server 2005

If upgrading from SQL Server 2000, after upgrading an instance and database to SQL Server 2008, the Snapshot Agent must be run for each merge replication publication. You do not need to do this for SQL Server 2005 upgrades. Do not apply a new snapshot; running the Snapshot Agent will update publication metadata because merge replication stores publication metadata in system tables. For more information about running the Snapshot Agent, see the following SQL Server 2008 Books Online topics:

* [How to: Create and Apply the Initial Snapshot (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkId=132658)
* [How to: Start and Stop a Replication Agent (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkId=132660)
* [How to: Create the Initial Snapshot (Replication Transact-SQL Programming)](http://go.microsoft.com/fwlink/?LinkId=132661)
* [Replication Agent Executables Concepts](http://go.microsoft.com/fwlink/?LinkId=132662)

Besides running the Snapshot Agent, you must also run the Merge Agent because it will update metadata stored for the subscriptions. For more information about running the Merge Agent, see the following SQL Server 2008 Books Online topics:

* [How to: Synchronize a Pull Subscription (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkId=132663)
* [How to: Synchronize a Push Subscription (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkId=132664)
* [How to: Synchronize a Pull Subscription (Replication Programming)](http://go.microsoft.com/fwlink/?LinkId=132665)

#### In-Place Upgrade

1. Generate scripts for the entire existing SQL Server 2000 or SQL Server 2005 replication topology and store them in a safe place.
2. If the Merge Agent is configured on the Subscriber for a pull subscription, disable the SQL Server Agent jobs related to the pull.
3. Upgrade the Distributor to SQL Server 2008. The Distributor may or may not be the same as the Publisher. Before upgrading the Distributor, disable any SQL Server Agent jobs related to replication, including any push subscriptions. The jobs involved will be similar to the ones that Figure 4-19, above, shows.
4. Ensure that each Subscriber is at a version of SQL Server that is compatible with SQL Server 2008 replication.
5. Check to see that SQL Server Agent is started on the Distributor.
6. If the Publisher is not the same as the Distributor and the Publisher or Subscriber will not be upgraded in the same outage, enable all the replication jobs at the newly upgraded Distributor to allow replication to work until all components are upgraded. When the upgrade process is initiated for either the Publisher or Subscriber, either disable all SQL Server Agent jobs relating to publication and distribution or disable networking to ensure that no attempts to propagate data will occur during the upgrade. Disabling networking may be easier, but both are valid options.

Having said the above, it is strongly recommended that you upgrade the Distributor and Publisher during the same outage. Publishers and Subscribers can be upgraded in stages, and data changes can happen on a Subscriber while the Publisher is being upgraded (and vice versa).

1. To upgrade the Publisher, follow these steps:
   1. Stop all traffic and kill any connections into the Publisher and Subscriber because both can update data. This should only be done on the instance or server being upgraded.
   2. Make sure that all existing transactions have been replicated and that no data is flowing via replication. This will ensure that in the upgrade, no data changes.
   3. Upgrade the Publisher to SQL Server 2008.
   4. Ensure that SQL Server Agent is started.
   5. We recommend that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
   6. Set the database compatibility level of upgraded databases to 100.
2. Upgrading the Subscriber(s) can be done in a staged fashion. Ensure that the Publisher and Distributor have been upgraded before the Subscriber.
3. After the Subscriber is upgraded, enable all the SQL Server Agent jobs for the subscription and enable all SQL Server Agent jobs on the Distributor.
4. Verify that SQL Server Agent is started at the Publisher, Distributor, and Subscriber.
5. Verify that the replication agents and jobs are started with no errors.
6. Verify that merge replication is working via the Replication Monitor.
7. Generate new scripts for the upgraded replication architecture.
8. The newly upgraded SQL Server environment is now ready for use by applications and end users.

#### Side-by-Side Upgrade to a New Server or Cluster

When not upgrading merge replication in-place, the upgrade process complexity will depend on the complexity of the replication topology and which components will be changed to SQL Server 2008. Tackle the switch to the new instances in logical groups, starting from the top down.

1. In the replication topology, where an in-place upgrade of an existing instance to SQL Server 2008 will not be performed, install a new instance of SQL Server 2008. This may be for a Publisher, a Distributor, or a Subscriber.
2. Generate scripts for the entire existing SQL Server 2000 or SQL Server 2005 replication topology.
3. Upgrade the scripts to SQL Server 2008 and make sure that the instance names and databases are updated to reflect their new locations. Before upgrading the scripts, make copies so that the old environment can be restored if necessary.
4. Ensure that the existing Publisher is at a version that can participate in the replication topology when SQL Server 2008 is added.
5. The Distributor must be SQL Server 2008. Run the appropriate upgraded replication script at the Distributor to create publication and distribution. At this point, the old replication topology is still being used.
6. If necessary, run the appropriate upgraded replication script at the Publisher. At this point, you are still actively using the old replication topology.
7. To upgrade one tier of merge replication, follow these steps:
   1. Initialize the new Publisher. Assuming the Publisher is going to be the main application database as well, migrate it using one of the methods described in "Methods for New Hardware and Side-by-Side Upgrades" earlier in this chapter. Just using a snapshot will not necessarily capture the entire database in the same way a backup will. Also, ensure that migrating the database to a new instance follows whatever approved and supported procedures the application vendor or developers provide. Do not just move it.
   2. Configure a new subscription from the old Publisher to the new Publisher.
   3. To do the cutover, stop all traffic and kill any connections into the Publisher to ensure that no one tries to access it during the upgrade. Because both sides can update data, no traffic or connections should be allowed at either side.
   4. Make sure that no transactions are left to replicate.
   5. After all transactions have been replicated, delete the subscription from the old Publisher to the new Publisher.
   6. Run the appropriate upgraded script at the Publisher to add it to the new replication topology.
   7. Run the appropriate upgraded scripts at the Subscriber to connect it to the new replication topology. If the Subscriber is not a new instance of SQL Server 2008, make sure it is at a version of SQL Server that can be supported in a SQL Server 2008 replication topology. We recommend that you delete the old subscription to ensure a clean configuration. This will mean a complete re-synchronization of the Subscriber.
   8. Verify that the replication agents and jobs are started with no errors.
   9. Verify that merge replication is working via the Replication Monitor. Also, insert some "dummy" data and make sure it propagates to all Subscribers.
8. Remember to script and migrate any objects required by the database that are not accounted for by replication.
9. Configure administration such as backup jobs for the database.
10. Generate new scripts for the upgraded replication architecture.
11. To ensure that no one will connect to the old environment, it is recommended that you stop the services if possible on the original topology after it has been migrated to the new instance. For more information, see the section "Decommissioning and Uninstalling After a Side-by-Side or New Hardware Upgrade" earlier in this chapter.
12. The newly upgraded SQL Server environment is now ready for use by applications and end users. Point applications and clients to the new locations if necessary.

### Transactional Replication

This section covers the specifics of upgrading transactional replication to SQL Server 2008.

#### Deprecated Features

Not much has changed in transactional replication since SQL Server 2005. To see what was deprecated or discontinued as well as breaking or behavior changes, review [Replication Backward Compatibility](http://go.microsoft.com/fwlink/?LinkId=133159) in SQL Server 2008 Books Online. Table 4-6 lists deprecated features in SQL Server 2008 transactional replication.

Table 4-6: Deprecated Features in SQL Server 2008 Transactional Replication

| **Feature** | **Description** | **Recommendation** |
| --- | --- | --- |
| Updatable subscriptions, including immediate updating and queued updating with snapshot and transactional publications | Deprecated | Use peer-to-peer replication |
| Replicating to Oracle 8 subscribers and from Oracle 8 publishers¹ | Deprecated | See [Oracle Publishing Overview](http://go.microsoft.com/fwlink/?LinkId=132687) and [Oracle Subscribers](http://go.microsoft.com/fwlink/?LinkId=132688) in SQL Server 2008 Books Online. |

#### Behavior Changes

When using row and/or page compression with a SQL Server 2008 source, the Distribution Agent does not check whether the Subscribers are a different version of SQL Server, so table creation will fail with the compression options.

**Important:** Do not enable data compression if SQL Server 2000 or SQL Server 2005 are part of the topology.

#### In-Place Upgrade

As noted earlier, performing an in-place upgrade is probably the least intrusive way of upgrading all servers participating in transactional replication.

**Upgrading with One-Way Transactional Replication.** Upgrading when there is only one Publisher (or Publisher and Republisher) is simpler than if the Subscribers are also Publishers.

1. Generate scripts for the entire existing SQL Server 2000 or SQL Server 2005 replication topology.
2. If the Distribution Agent is configured on the Subscriber for a pull subscription, disable the SQL Server Agent jobs related to the pull.
3. Upgrade the Distributor to SQL Server 2008. The Distributor may or may not be the same as the Publisher. Before upgrading the Distributor, disable any SQL Server Agent jobs related to replication, including any push subscriptions. The jobs involved will be similar to the ones shown in Figure 4-19 above.
4. Ensure that each Subscriber is at a version of SQL Server that is compatible with SQL Server 2008 replication.
5. Check to see that SQL Server Agent is started on the Distributor.
6. If the Publisher is not the same as the Distributor and the Publisher or Subscriber will not be upgraded in the same outage, enable all the replication jobs at the newly upgraded Distributor to allow replication to work until all components are upgraded. When the upgrade process is initiated for either the Publisher or Subscriber, either disable all SQL Server Agent jobs relating to publication and distribution or disable networking to ensure that no attempts to propagate data will occur during the upgrade. Disabling networking may be easier, but both are valid options.

Having said the above, it is strongly recommended that you upgrade the Distributor and Publisher during the same outage. Publishers and Subscribers can be upgraded in stages, and data changes can happen on a Subscriber while the Publisher is being upgraded (and vice versa).

1. To upgrade the Publisher, follow these steps:
2. Stop all traffic and kill any connections into the Publisher to ensure that no one tries to access it during the upgrade. This should only be done on the server or instance being upgraded.
3. Although the Publisher can be upgraded even when not all transactions have been replicated, it would be better to ensure that all existing transactions have been replicated from the Publisher before upgrading. For transactional replication, run sp\_repltrans at the Publisher to get the outstanding transactions marked for publication. Once the result set is empty, the upgrade can begin.
4. Upgrade the Publisher to SQL Server 2008.
5. Ensure that SQL Server Agent is started.
6. We recommend that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
7. Set the database compatibility level of upgraded databases to 100.
8. After the Subscriber is upgraded, enable all the SQL Server Agent jobs for the subscription and enable all SQL Server Agent jobs on the Distributor.
9. Verify that the replication agents and jobs are started with no errors.
10. To upgrade the Subscriber, follow these steps:
11. You do not need to stop all traffic or kill connections into the Publisher during an upgrade of the Subscriber. However, be aware that the Publisher will not be able to flush transactions until the Subscriber is back online. It will also affect the Publisher's ability to back up the transaction log.
12. Upgrade the Subscriber to SQL Server 2008.
13. Ensure that SQL Server Agent is started.
14. We recommend that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
15. Set the database compatibility level of upgraded databases to 100.
16. After the Subscriber is upgraded, enable all the SQL Server Agent jobs for the subscription and enable all SQL Server Agent jobs on the Distributor.
17. Verify that the replication agents and jobs are started with no errors.
18. Verify that transactional replication is working via the Replication Monitor. Also, insert some "dummy" data and make sure it propagates to all Subscribers.
19. Generate new scripts for the upgraded replication architecture.
20. Archive the SQL Server 2000 or SQL Server 2005 replication scripts.
21. The newly upgraded SQL Server environment is now ready for use by applications and end users.

**Upgrading Bi-Directional Transactional Replication, Updating Subscribers, or Peer-to-Peer.** Upgrading where bi-directional transactional replication or peer-to-peer is involved is similar to upgrading with merge replication because both the Publisher and Subscriber can be updating data.

1. Generate scripts for the entire existing SQL Server 2000 or SQL Server 2005 replication topology.
2. If the Distribution Agent is configured on the Subscriber for a pull subscription, disable the SQL Server Agent jobs related to the pull.
3. Upgrade the Distributor to SQL Server 2008. The Distributor may or may not be the same as the Publisher. Before upgrading the Distributor, disable any SQL Server Agent jobs related to replication, including any push subscriptions. The jobs involved will be similar to the ones shown in Figure 4-19 above.
4. Ensure that each Subscriber is at a version of SQL Server that is compatible with SQL Server 2008 replication.
5. Check to see that SQL Server Agent is started on the Distributor.
6. If the Publisher is not the same as the Distributor and the Publisher or Subscriber will not be upgraded in the same outage, enable all the replication jobs at the newly upgraded Distributor to allow replication to work until all components are upgraded. When the upgrade process is initiated for either the Publisher or Subscriber, either disable all SQL Server Agent jobs relating to publication and distribution or disable networking to ensure that no attempts to propagate data will occur during the upgrade. Disabling networking may be easier, but both are valid options.

Having said the above, it is strongly recommended that you upgrade the Distributor and Publisher during the same outage. Publishers and Subscribers can be upgraded in stages, and data changes can happen on a Subscriber while the Publisher is being upgraded (and vice versa).

1. To upgrade the Publisher and Subscriber, follow these steps:
   1. Stop all traffic and kill any connections into the Publisher and Subscriber because both can update data.
   2. Although the Publisher can be upgraded even when not all transactions have been replicated, it is better to ensure that all existing transactions have been replicated from the Publisher before upgrading. This will ensure that in the upgrade, no data changes. For transactional replication, run sp\_repltrans at the Publisher to get the outstanding transactions marked for publication. Once the result set is empty, the upgrade can begin.
   3. Upgrade the Publisher to SQL Server 2008.
   4. Ensure that SQL Server Agent is started.
   5. We recommend that you run all the necessary health checks, including DBCC CHECKDB, at this point to ensure the well-being of the newly upgraded databases.
   6. Set the database compatibility level of upgraded databases to 100.
   7. After the Subscriber is upgraded, enable all the SQL Server Agent jobs for the subscription and enable all SQL Server Agent jobs on the Distributor.
2. Verify that the replication agents and jobs are started with no errors.
3. Verify that transactional replication is working via the Replication Monitor. Also, insert some "dummy" data and make sure it propagates to all Subscribers.
4. Generate new scripts for the upgraded replication architecture.
5. Archive the SQL Server 2000 or SQL Serve 2005 replication scripts.
6. The newly upgraded SQL Server environment is now ready for use by applications and end users.

**Side-by-Side Upgrade to a New Server or Cluster.** When transactional replication will not be upgraded in-place, determine the complexity of the topology. Tackle the switch to the new instances in logical groups, starting from the top down. The complexity of this form of upgrade will depend on which pieces are going to be changed.

In the replication topology, where you are not doing an in-place upgrade of an existing instance to SQL Server 2008, install a new instance of SQL Server 2008. This may be for a Publisher, a Distributor, or a Subscriber.

1. Generate scripts for the entire existing SQL Server 2000 or SQL Server 2005 replication topology.
2. Upgrade the scripts to SQL Server 2008 and make sure that the instance names and databases are updated to reflect their new locations. Before upgrading the scripts, make copies so that the old environment can be restored if necessary.
3. Ensure that the existing Publisher is at a version that can participate with SQL Server 2008 in the topology.
4. The Distributor must be SQL Server 2008. Run the appropriate upgraded replication script at the Distributor to create publication and distribution. At this point, the old replication topology is in use.
5. If necessary, run the appropriate upgraded replication script at the Publisher. At this point, the old replication topology is still being used.
6. To upgrade one tier of transactional replication, follow these steps:
   1. Initialize the new Publisher. Assuming the Publisher is going to be the main application database as well, migrate it using one of the methods described in "Methods for New Hardware and Side-by-Side Upgrades" earlier in this chapter. Just using a snapshot will not necessarily capture the entire database. Also, ensure that migrating the database to a new instance follows whatever approved and supported procedures the application vendor or developers provide. Do not just move it.
7. Configure a new subscription from the old Publisher to the new Publisher.
8. To do the cutover, stop all traffic and kill any connections into the Publisher to ensure that no one tries to access it during the upgrade. If peer-to-peer, updating Subscribers, and/or bi-directional transactional replication are being used, ensure that neither side has any connections to the Publishers.
9. Make sure that no transactions are left to replicate. Run sp\_repltrans at the Publisher to get a list of the outstanding transactions marked for publication.
10. Once the result set is empty, delete the subscription from the old Publisher.
11. Run the appropriate upgraded script at the Publisher to add it to the new replication topology.
12. Run the upgraded scripts at the Subscriber to connect it to the new replication topology. If the Subscriber is not a new instance of SQL Server 2008, make sure it is at a version of SQL Server that can be supported in a SQL Server 2008 replication topology. We recommend that the old subscription be erased to ensure a clean configuration. This will mean a complete re-synchronization of the Subscriber.
13. Verify that the replication agents and jobs are started with no errors.
14. Verify that transactional replication is working via the Replication Monitor. Also, insert some "dummy" data and make sure it propagates to all Subscribers.
15. Remember to script and migrate any objects required by the database that are not accounted for by replication.
16. Configure administration such as backup jobs for the database.
17. Generate new scripts for the upgraded replication architecture.
    * 1. To ensure that no one will connect to the old environment, it is recommended that you stop the services if possible on the original topology after it has been migrated to the new instance. For more guidance, see the section "Decommissioning and Uninstalling After a Side-by-Side or New Hardware Upgrade" in this chapter.
      2. The newly upgraded SQL Server environment is now ready for use by applications and end users. Point applications and clients to the new locations if necessary.

### Additional Information for Upgrading Replicated Databases

See the following SQL Server 2008 Books Online topics for more information about upgrading replicated databases:

* [Behavior Changes in SQL Server Replication](http://go.microsoft.com/fwlink/?LinkId=132689)
* [Breaking Changes in SQL Server Replication](http://go.microsoft.com/fwlink/?LinkId=132690)
* [Considerations for Upgrading Replicated Databases](http://go.microsoft.com/fwlink/?LinkID=132650)
* [Deprecated Features in SQL Server Replication](http://go.microsoft.com/fwlink/?LinkId=132691)
* [Discontinued Functionality in SQL Server Replication](http://go.microsoft.com/fwlink/?LinkId=132692)
* [How to: Enable Initialization with a Backup for Transactional Publication (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkId=132693)
* [How to: Initialize Transactional Subscription from a Backup (Replication Transact-SQL Programming)](http://go.microsoft.com/fwlink/?LinkId=132694)
* [How to: Initialize a Subscription Manually (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkId=132695)
* [How to: Initialize a Subscription Manually (Replication Transact-SQL Programming)](http://go.microsoft.com/fwlink/?LinkId=132696)
* [How to: Synchronize a Pull Subscription (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkID=132663)
* [How to: Synchronize a Pull Subscription (Replication Programming)](http://go.microsoft.com/fwlink/?LinkID=132665)
* [How to: Synchronize a Push Subscription (SQL Server Management Studio)](http://go.microsoft.com/fwlink/?LinkID=132664)
* [How to: Synchronize a Push Subscription (Replication Programming)](http://go.microsoft.com/fwlink/?LinkId=132697)
* [How to: Upgrade Replication Scripts (Replication Transact-SQL Programming)](http://go.microsoft.com/fwlink/?LinkId=132698)
* [Considerations for Installing SQL Server Replication](http://go.microsoft.com/fwlink/?LinkId=132699)
* [Replication Backward Compatibility](http://go.microsoft.com/fwlink/?LinkID=132646)
* [Using Multiple Versions of SQL Server in a Replication Topology](http://go.microsoft.com/fwlink/?LinkID=132645)

For an up-to-date collection of additional references for upgrading replication, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkID=132510).

### Upgrade Considerations for SQL Server Analysis Services

Similarly to SQL Server relational engine you have several choices upgrading Analysis Services. The considerations for upgrade are also very similar to once mentioned above in this paper for SQL Server Relational engine: In-Place Upgrade, Side-by-Side and so on. Though Analysis Services uses different set of technologies and therefore your options for implementing highly available solutions are different from once for Relational Engine.

### Methods for New Hardware and Side-by-Side Upgrades

There are various methods of moving and subsequently upgrading your SQL Server databases if you are going to use new hardware or perform a side-by-side upgrade.

#### Backup and Restore

Similarly to SQL Server Database Engine, you can perform a backup of your Analysis Services databases on the live server and then restore then on the newly installed instance.

Using a database backup and restoring it is the most traditional method of migrating a database from one server to another. No downtime is involved because all Analysis Services backups are performed as online operations. The limiting factors are time and speed of your hardware, especially your disk subsystem and network.

#### Detach and Attach

Analysis Services 2008 introduced the ability to detach and attach databases. Detaching and attaching is *not* the same as performing a backup, but it does share some of its cons. Because this functionality is not available for earlier versions of Analysis Services you can only use it going forward in upgrade from Analysis Services 2008 RTM to the future versions.

#### Synchronization

Many Analysis Services customers use Synchronization functionality to implement high-availability and disaster recover solutions. Setting up Synchronization between two servers requires database administrator to set up external job or recurring task that is sending synchronization command to the target server telling it to synchronize itself with the source server. You can setup such recurring jobs using SQL Server Agent or SQL Server Integration Services.

Synchronizing servers require them to be of the same version. One of the scenarios for upgrade you would use will be to start upgrade of the target server immediately after synchronizing it to the server. Once that happen you can start upgrading your source server while target server is available for query.

#### Multiple Instances and Multiple Components of the Same Instance

SQL Server setup allows for installing side by side several server instances. You can upgrade each Analysis Services instance independently.

When SQL Server and Analysis Services installed using the same instance name you upgrade both at the same time.

When planning for high available solution you often have different requirements for SQL Server relational engine and Analysis Services. In some cases - for instance when installing on the failover cluster you would like to keep SQL Server relational engine installed on a different cluster node from Analysis Services and therefore you would run setup multiple times installing the SQL Server relational engine and Analysis Services using different instance names.

Accessing SQL Server relational engine and Analysis Services is different on the failover cluster. When installed on failover cluster giving an instance name Analysis Services would appear to the user as default instance. For instance when installing on the cluster group with network name MyServer and instance name Instance1 you access SQL Server Relational engine using MyServer\Instance1 , whereas Analysis Services would accessed by just MyServer.

## Upgrading Failover Clusters

This section describes the considerations and processes for upgrading existing SQL Server 2000 or SQL Server 2005 failover clustering instances to SQL Server 2008.

### Considerations for Upgrading a SQL Server 2000 Failover Cluster to SQL Server 2008

Upgrade of SQL Server 2000 Analysis Services to SQL Server 2008 is not supported on failover clusters. Clustering of Analysis Services 2000 is performed manually by using information in the Microsoft Knowledge Base article [How to cluster SQL Server 2000 Analysis Services in Windows 2000 and in Windows Server 2003](http://go.microsoft.com/fwlink/?LinkID=132702). To upgrade to Analysis Services 2008, we recommend that you install Analysis Services 2008 side-by-side with Analysis Services 2000 and use Migration Wizard to migrate your metadata. After that you must process your Analysis Services cubes to load data into it.

### Considerations for Upgrading a SQL Server 2005 Failover Cluster to SQL Server 2008

Analysis Services 2005 clustering installation is done using SQL Server 2005 Setup. To upgrade from Analysis Services 2005, you run SQL Server 2008 Setup and it will detect existing installation and will prompt you for instance of Analysis Services to upgrade. Analysis Services 2008 is fully backward compatible and you should be able to query data in Analysis Services database after upgrade. You can find the step-by-step instructions for upgrading via Setup through an in-place upgrade in [How to: Upgrade a SQL Server Failover Cluster Instance (Setup)](http://go.microsoft.com/fwlink/?LinkID=132450) in SQL Server 2008 Books Online. This section walks you through the process of an in-place upgrade but not through each Setup screen because the process overlays the Setup steps.

### Additional References for Clustering Upgrades

The list below includes links to more information in SQL Server 2008 Books Online related to failover clustering:

* [Upgrading a SQL Server 2008 Failover Cluster](http://go.microsoft.com/fwlink/?LinkID=132180)
* [How to: Install Analysis Services on a Failover Cluster](http://go.microsoft.com/fwlink/?LinkId=132704)

## Conclusion

It is possible to deploy and upgrade to SQL Server 2008 with minimal downtime. Although it is impossible to completely avoid times when the databases will be unavailable, preparation and testing will be the ultimate defenses against failures and extended outages. If you are using the high-availability features of SQL Server 2000 or SQL Server 2005 and will be upgrading, each feature and version will have different considerations and processes that you must take into account when upgrading to SQL Server 2008.

## Additional References

For an up-to-date collection of additional SQL Server 2008 high-availability references, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkID=132510).

[SQL Server High Availability Web site](http://go.microsoft.com/fwlink/?LinkId=132705)

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkId=132512)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132514)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkId=132515)

# Database Security

## Introduction

The number of security factors affecting the upgrade of the SQL Server relational engine from SQL Server 2000 or SQL Server 2005 to SQL Server 2008 is fairly small. And for most systems, the changes will be nearly invisible. Almost all existing SQL Server security configurations will upgrade automatically without intervention by the database administrator. In this chapter, you will learn about any security-related upgrade issues you need to consider for the Database Engine, including new and enhanced features as well as security functionality that is no longer part of SQL Server.

## New Security Features

Security in SQL Server 2008 has been enhanced and strengthened. But database administrators must remove any blocking issues and follow up in newly upgraded environments to take advantage of these enhancements. If you address any issues early on in the upgrade planning process, your upgrade experience will be much smoother. For a comprehensive discussion of upgrading database security, see [Security Considerations for a SQL Server Installation](http://go.microsoft.com/fwlink/?LinkId=133079) in SQL Server 2008 Books Online. Let's look at how the new and enhanced security features in SQL Server 2008 could affect your upgrade planning and process.

Table 5-1 provides a summary of the new security features in SQL Server 2008 and how they might affect the upgrade process from SQL Server 2000 or SQL Server 2005 to SQL Server 2008.

Table 5-1: SQL Server 2008 Security Features

| **New SQL Server 2008 Security Features** | **Benefit** | **When Available** | **Affects the Upgrade Process** |
| --- | --- | --- | --- |
| Services and configurations  "off by default" | Greater security at install time | Immediately after upgrade | No for in-place upgrade;  yes for side-by-side upgrade |
| Metadata visibility configuration | Less exposed surface area | Immediately after upgrade | Yes |
| Strong password policies for SQL Server authentication | Integrated Windows and SQL Server behavior | Immediately after upgrade | No |
| All permissions grantable | Permissions easier to manage and more granular | Immediately after upgrade | No |
| User/schema separation | No recording when staff changes, separates dbo and developers | Immediately after upgrade | No |
| Keys and encryption | Compliance with privacy requirements, secure communications | Minimal work to leverage | No |
| Transparent Data Encryption | Encrypts database files and backups without requiring changes to any application | Minimal work to leverage | No |
| Execution context,  signed procedures | Principle of least privilege, audit ability | Design and architect | No |
| Extensible Key Management | Lets third-party vendors register their devices in SQL Server | Design and architect | No |
| SQL Server Audit | Creates customized audits of Database Engine events to meet compliance | Design and architect | No |
| Service hardening via per-service SID | Isolation across services and defense in-depth on Windows Vista and Windows Server 2008 | Immediately after upgrade | No |

As you can see from this table, the new security features that might affect your upgrade arise from having features and services "off by default" and from strengthened metadata visibility. The other features are enhancements that you can take advantage of after your upgrade, but they will not block or impede the upgrade process.

### New Configuration Tools

Another security-related feature change you need to be aware of is that the Surface Area Configuration (SAC) tool has been removed from SQL Server 2008 and replaced by SQL Server Configuration Manager and SQL Server Policy-Based Management. SQL Server Configuration Manager is a Microsoft Management Console (MMC) snap-in that you can start through the Start menu or add to custom Management Console displays. You use SQL Server Configuration Manager to set protocol, connection, and startup options. For more information about this tool, see [SQL Server Configuration Manager](http://go.microsoft.com/fwlink/?LinkId=133081) in SQL Server 2008 Books Online.

You use the new Policy-Based Management capability in SQL Server 2008 to help set Database Engine, Analysis Services, and Reporting Services configurations. You can import the best practice policies that SQL Server 2008 provides to evaluate the configurations for your instances, instance objects, databases, and database objects.

**Note**: As in SQL Server 2005, you can also use sp\_configure to set Database Engine features.

### Configuring Services and Connections

SQL Server Configuration Manager lets you configure SQL Server 2008 services and (when relevant) remote connections. If you use an in-place upgrade, SQL Server 2008 Setup preserves your source SQL Server instance's service configurations, except for the new service for SQL Server 2008 full-text search. Even for an in-place upgrade, service setting for SQL Server 2005 full-text search will not be preserved.

**Note**: If you are performing an in-place upgrade from SQL Server 2000, the new Browser service will be installed with default settings, as Table 5-2 shows. For more information about the SQL Server Browser service, see [SQL Server Browser Service](http://go.microsoft.com/fwlink/?LinkId=133083) in SQL Server 2008 Books Online.

If you use a side-by-side upgrade method, you might need to adjust the default service settings for all services. To change the service accounts, password, service startup type, or other properties of any SQL Server–related service, use SQL Server Configuration Manager. In addition to changing service accounts, SQL Server Configuration Manager changes other configurations such as permissions in the Windows registry. Changing service accounts by using the Windows Service Control Manager is not supported. For more information, see [SQL Server Configuration Manager](http://go.microsoft.com/fwlink/?LinkId=133081) in SQL Server 2008 Books Online.

Table 5-2 lists the SQL Server 2008 services and their status after an initial installation.

Table 5-2: SQL Server 2008 Services Default Status After Installation

| **SQL Server Service** | **Optional Accounts** | **Startup Type** |
| --- | --- | --- |
| SQL Server | SQL Server Express: Domain User, Local System, Network Service;  All other editions: Domain User, Local System, Network Service | Automatic;  manual in failover cluster installations |
| SQL Server Agent | Domain User, Local System, Network Service | Manual;  disabled in SQL Server Express and SQL Server Express with Advanced Services |
| Analysis Services | Domain User, Network Service, Local Service, Local System | Automatic |
| Reporting Services | Domain User, Local System, Network Service, Local Service | Automatic |
| Integration Services | Domain User, Local System, Network Service, Local Service | Automatic |
| Full-text Filter Daemon Launcher | Use an account different than the account for the SQL Server service. The account will default to Local Service on Windows Server 2008 and Windows Vista. | Automatic |
| SQL Server Browser | Local Service | Disabled;  automatic in failover cluster and named instances installations |
| SQL Server Active Directory Helper | Local System, Network Service | Disabled |
| SQL Writer | Local System | Automatic |

If you upgrade from SQL Server 2000, you will find the following new services: the SQL Server Browser service (now separate from the SQL Server service), the Integration Services service, and SQL Writer. SQL Server 2008 now also includes an instance-specific Full-text Filter Daemon Launcher service and no longer uses the MS Search service.

The Full-text Filter Daemon Launcher service replaces the instance-specific Full Text Search services in SQL Server 2005. For more information about the Full-text Filter Daemon Launcher service, see Chapter 6, "Full-Text Search," in this guide.

Some of these services might be optional for your instance. You can enable and disable these services, depending on the functionality your SQL Server instance requires, by using SQL Server Configuration Manager. You should review these new service accounts and their security requirements before attempting an upgrade to SQL Server 2008. Use the principle that if a service is not needed, it should be disabled. You can also enable and disable remote connections from many of the services.

For details about service accounts and services, see [Setting Up Windows Service Accounts](http://go.microsoft.com/fwlink/?LinkId=133085) in SQL Server 2008 Books Online.

### Service Account Security

Unlike SQL Server 2000 service accounts, SQL Server 2005 and SQL Server 2008 do not require local server administrative rights on the database server. You should assign domain accounts to each SQL Server service instead of using the Local System or Network Service accounts.

**Note**: Although the Network Service account has low privileges, other services use it, and therefore it violates isolation guarantees.

If you create the accounts ahead of time, the SQL Server 2008 Setup program will grant the minimal privileges to the accounts for the services to run properly.

For stand-alone instances, SQL Server Setup creates local security groups for the different services, grants the needed privileges to these security groups, and adds the service accounts or per-service SID available only on Windows Vista and Windows Server 2008 to these groups. Per-service SIDs are used to enable service isolation when SQL Server 2008 is installed on Windows Server 2008 or Windows Vista. When installing on Windows 2003 or Windows XP, the service accounts are used.

On failover cluster installations, you need to create domain groups before installing the instance on Windows Server 2003 or Windows XP and then assign them to the services during setup. With installations on Windows Server 2008, SQL Server Setup installation uses per-service SIDs. If you perform an in-place upgrade, Setup preserves the existing domain groups and access control list (ACL) configuration.

For more information about service account security, see [Setting Up Windows Service Accounts](http://go.microsoft.com/fwlink/?LinkId=133085) in SQL Server 2008 Books Online.

For more information about per-service SIDs in Windows Server 2008 and Windows Vista, see the "Security improvements to Windows services" in [What's New for Operating System Hardening and Integrity for Windows Server 2008](http://go.microsoft.com/fwlink/?LinkId=133086) on MSDN.

For details about the new SQL Server Full-text Filter Daemon Launcher services, see "Understanding the Filter Daemon Process" in the [SQL Server 2008 Full-Text Search: Internals and Enhancements](http://go.microsoft.com/fwlink/?LinkId=133087) white paper.

### Configuring Features

Through the new Policy-Based Management capability in SQL Server 2008, you can enable or disable many Database Engine features as well as options for the Analysis Services and Reporting Services components. You can also set options for the Database Engine by using the sp\_configure stored procedure.

All these options are off by default with a new installation, although the Windows Data Access Components (DAC) on clusters will have remote use enabled by default. The purpose of having these options off by default is to reduce the potential attack surface of a new installation. You can selectively enable these options as required for your SQL Server 2008 instance.

With an in-place upgrade, all feature configurations stay in their pre-upgrade state. But you can use Policy-Based Management to disable features that you are not using anymore and reduce the attack surface of the upgraded instance.

Here are the Database Engine configuration features that you can set through the Surface Area Configuration facet of Policy-Based Management:

* Ad hoc remote queries
* Common Language Runtime (CLR) integration
* DAC (remote use of the Dedicated Administrator Connection)
* Database Mail
* Native XML Web services (if HTTP endpoints are defined)
* OLE automation
* Service Broker
* SOAP endpoints
* SQL Mail
* Web Assistant stored procedures
* xp\_cmdshell

You can find out-of-the-box security best practices policies, including Surface Area Configuration, in the SQL Server 2008 installation path in the \100\Tools\Policies\DatabaseEngine\1033 folder. You can also download the policies as part of the [Microsoft SQL Server 2008 Feature Pack August 2008](http://go.microsoft.com/fwlink/?LinkId=133088).

For more information about Policy-Based Management, see [Administering Servers by Using Policy-Based Management](http://go.microsoft.com/fwlink/?LinkId=133089) in SQL Server 2008 Books Online.

### Metadata Visibility

SQL Server 2008, like SQL Server 2005, has an entirely new approach to metadata compared to SQL Server 2000 and earlier releases. And although the new version has mechanisms to preserve backward compatibility, there are some important behavioral changes you need to be aware of.

The most important change is the limitation on viewing metadata. Access to catalog views and system metadata is no longer available by default to guest users or members of the PUBLIC role. This restriction is also reflected in users' ability to inspect metadata using SQL Server Management Studio. You can see SQL catalog metadata for objects you own or have some permission on, and you can see information in dynamic management views (DMVs) if you have been granted the VIEW SERVER STATE permission.

In addition, in SQL Server 2008, the base system tables are hidden. For backward compatibility, SQL Server exposes legacy system base tables as views called *compatibility views*. These views expose the same metadata as the legacy versions but are read-only. For example, in SQL Server 2000, sysindexes is a table in each database, but in SQL Server 2005 and SQL Server 2008, sys.sysindexes is a compatibility view.

**Note**: Some SQL Server 2008 compatibility views have behavioral changes that differ from the legacy system tables. The rowmodctr column in the sys.sysindexes compatibility view, for example, displays values somewhat differently from the way the sysindexes system table does in previous versions of SQL Server. For more information about these changes, see [sys.sysindexes (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=133090) in SQL Server 2008 Books Online.

If you have code that refers to legacy system tables, note that some behavior will change. You cannot update system base tables, and therefore, you cannot update compatibility views. If you have any code that updates system base tables in a prior version of SQL Server, you need to remove it because the code will fail in SQL Server 2005 and SQL Server 2008.

In addition, some code will not work if it accesses undocumented tables or columns in system objects. For example, the following security-related columns in compatibility views will return NULL or 0 in SQL Server 2008:

* sysremotelogins.status
* sysoledbusers.rmtpassword

Therefore, you should change all code that accesses legacy system tables to access system views and functions instead. For information about mapping legacy system tables to system views and functions in SQL Server 2005 and SQL Server 2008, see [Mapping System Tables to System Views](http://go.microsoft.com/fwlink/?LinkId=133091) in SQL Server 2008 Books Online.

## Preparing to Upgrade

After you become familiar with the new features you need to prepare for in your upgrade plan and process, you need to understand which features have been deprecated or discontinued in SQL Server 2008 and review the changes that could block your upgrade or change the behavior of your applications after the upgrade.

### Deprecated Features

Some security-related features are deprecated in SQL Server 2008. Although they continue to operate in SQL Server 2008, they will be removed in a future version of SQL Server. These deprecated features have no immediate effect on your upgrade to SQL Server 2008, but they will affect your upgrade to a later version.

For comprehensive information about these features, see [Deprecated Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133092) in SQL Server 2008 Books Online. The Books Online lists of deprecated security features consist primarily of system stored procedures that are now replaced by Transact-SQL commands. For example, sp\_adduser and sp\_dropuser are replaced by CREATE USER and DROP USER, respectively, and SETUSER is replaced by EXECUTE AS. These procedures are deprecated because they do not work with user/schema separation. As soon as you take advantage of new security commands such as CREATE USER and CREATE SCHEMA, you should also switch from using compatibility views such as sysobjects and use catalog views such as sys.objects instead.

Table 5-3 lists the deprecated security features in SQL Server 2008 and their replacements.

Table 5-3: Deprecated Security Features

| **Deprecated Feature** | **Replacement** |
| --- | --- |
| sp\_addapprole  sp\_dropapprole | CREATE APPLICATION ROLE  DROP APPLICATION ROLE |
| sp\_addlogin  sp\_droplogin | CREATE LOGIN  DROP LOGIN |
| sp\_adduser  sp\_dropuser | CREATE USER  DROP USER |
| sp\_grantdbaccess  sp\_revokedbaccess | CREATE USER  DROP USER |
| sp\_addrole  sp\_droprole | CREATE ROLE  DROP ROLE |
| sp\_approlepassword  sp\_password | ALTER APPLICATION ROLE  ALTER LOGIN |
| sp\_changeobjectowner | ALTER SCHEMA or ALTER AUTHORIZATION |
| sp\_defaultdb  sp\_defaultlanguage | ALTER LOGIN |
| sp\_denylogin  sp\_grantlogin  sp\_revokelogin | ALTER LOGIN DISABLE  CREATE LOGIN  DROP LOGIN |
| USER\_ID | DATABASE\_PRINCIPAL\_ID |
| xp\_grantlogin  xp\_revokelogin  xp\_loginConfig | CREATE LOGIN  DROP LOGIN  SERVERPROPERTY('IsIntegratedSecurityOnly') |
| sp\_change\_users\_login | ALTER USER |
| sp\_srvrolepermission  sp\_dbfixedrolepermission | The output does not reflect changes to the permissions hierarchy implemented in SQL Server 2008. For more information, see [Permissions of Fixed Server Roles (Database Engine)](http://go.microsoft.com/fwlink/?LinkId=133094) in SQL Server 2008 Books Online. |
| sp\_addremotelogin  sp\_addserver  sp\_dropremotelogin  sp\_helpremotelogin  sp\_remoteoption | Use linked servers instead of remote servers. |
| @@remserver | Use linked servers instead of remote servers. |
| sp\_dropalias | Replaces aliases with a combination of database users and roles. Remove aliases with sp\_dropalias. |
| sys.database\_principal\_aliases | Use roles instead of aliases. |
| GRANT ALL  DENY ALL  REVOKE ALL | GRANT, DENY, and REVOKE specific permissions. |
| PERMISSIONS intrinsic function | sys.fn\_my\_permissions |
| SETUSER | EXECUTE AS |

To fully understand these security upgrade issues, make sure you review [SQL Server 2008 Database Engine Backward Compatibility](http://go.microsoft.com/fwlink/?LinkId=133097) in SQL Server 2008 Books Online.

### Discontinued Features

A number of security features are discontinued in SQL Server 2008, and you need to adjust your applications accordingly or they will not work properly. You can address many of these issues before the upgrade process, but they will not block an upgrade. Table 5-4 lists currently documented security-related features that are discontinued in SQL Server 2008.

Table 5-4: Discontinued Security-Related Features

| **Discontinued Feature** | **Explanation/Replacement** |
| --- | --- |
| Remote logins are no longer trusted. | SQL Server 2005 and SQL Server2008 no longer support setting remote logins as trusted. Scripts using the sp\_remotelogin system stored procedure to mark remote logins as trusted must be modified. |
| The following network libraries are no longer supported in SQL Server 2005 or SQL Server 2008: Multiprotocol, Banyan VINES, AppleTalk, and IPX/SPX (see note below). | Change the network library for connections to use TCP/IP (preferred) or named pipes. Data transmission encryption can be accomplished by using a certificate and SSL. |
| The following security-related system stored procedures do not function in SQL Server 2005 or SQL Server 2008:   * xp\_eventlog * xp\_GetAdminGroupName * xp\_GetFileDetails * xp\_GetLocalSystemAccountName * xp\_IsNTAdmin * xp\_MSLocalSystem * xp\_MSnt2000 * xp\_SetSecurity | Remove references to security-related undocumented system stored procedures. |
| The sysxlogins table is no longer supplied. | Replace references to sysxlogins in SQL Server 2005 and SQL Server 2008 with references to the sys.server\_principals catalog view or the syslogins compatibility view. |
| Column-level permissions on system objects are not supported in SQL Server 2005 or SQL Server 2008. | Remove statements from your application that grant, deny, or revoke column-level permissions on system objects. |
| SQL Mail in SQL Server 2005 and SQL Server 2008 no longer supports attachments. | Use Database Mail for attachments. |
| Unconverted SQL Server 6.5 passwords not supported in SQL Server 2008. | SQL Server 6.5/7.0 logins that were upgraded but never activated on SQL Server 2000 or SQL Server 2005 must have their passwords reset to function on SQL Server 2008. |
| Winsock Proxy configuration not supported. | SQL Server 2005 and SQL Server 2008 cannot configure Windows components. |
| sp\_addalias | SQL Server 2008 replaces aliases with a combination of database users and roles. Remove aliases by using sp\_dropalias. |
| sp\_addgroup  sp\_changegroup sp\_dropgroup sp\_helpgroup | Use roles instead of groups. |
| Surface Area Configuration tool | Use SQL Server Configuration Manager and Policy-Based Management features instead. |

**Note:** The SQL Server service will not start if it fails to listen on at least one network protocol or if there are errors related to SSL communication. If such failures occur, SQL Server will log an error message to the SQL Server error log indicating the problem.

### Breaking Changes

The SQL Server 2008 Setup program can detect three security-related breaking changes that will prevent an in-place upgrade from starting. In a side-by-side upgrade, these issues are still blockers, but you can deal with them slightly differently. Table 5-5 lists these blocking security issues.

Table 5-5: Security Issues That Will Block an Upgrade

|  |  |
| --- | --- |
| **Blocking Issue** | **Solution** |
| User name of *sys* | The database being upgraded cannot have a user with the name of *sys*. The *sys* name is reserved in SQL Server 2008, and any database user with this name must be renamed before starting the upgrade process. |
| Duplicate login SIDs | SQL Server 2008 does not allow duplicate SIDs. Database administrators must remove one of the duplicate logins and its associated users before starting the upgrade process. |
| Login name that is the same as a fixed server role | Fixed server role names are reserved in SQL Server 2008. You must rename the login before starting the upgrade process. |

In the case of a side-by-side upgrade, a user name of *sys* in a system database (master, model, or msdb) will block the upgrade, and you must remove the user name before the upgrade process.

If you have a user named *sys* in a user database, the attach process will result in a suspect database. You will have to bring the database online in single-user mode, transfer objects owned by that user name to a new user name, and then drop the user name. For more details, see the "Rename user sys" topic in the Upgrade Advisor Help file. You cannot create logins on a new SQL Server 2008 instance that have duplicate SIDs or that match fixed server role names.

You can find a full list of reserved keywords in [Reserved Keywords](http://go.microsoft.com/fwlink/?LinkId=133099) in SQL Server 2008 Books Online. Although it is syntactically possible to use SQL Server reserved keywords as identifiers and object names in Transact-SQL scripts, you can do this only by using delimited identifiers.

### Behavior Changes

After you address the new, deprecated, or discontinued security functionality and blocking issues in your upgrade plan, you need to investigate the security-related features that might not cause an error but that behave differently in SQL Server 2008. You need to address these issues as well because the behavior change might lead to application problems and user complaints. Table 5-6 lists these security-related behavior changes.

Table 5-6: Security-Related Behavior Changes in SQL Server 2008

| **Behavior Change** | **Explanation** |
| --- | --- |
| By default, system metadata is no longer viewable by PUBLIC. | By default, access to virtual tables and system metadata is no longer available to guest users or members of the PUBLIC role. Scripts and processes that view system metadata in virtual tables or system objects should be modified accordingly or sufficient permissions granted to the person or process running the scripts. |
| The ALL permission is deprecated. | This option is deprecated and maintained only for backward compatibility. It does not grant all possible permissions. For a list of granted permissions, see [GRANT (Transact-SQL)](http://msdn.michttp/go.microsoft.com/fwlink/?LinkId=133101) in SQL Server 2008 Books Online. |
| Case-sensitive passwords are required. | Case-insensitive password comparisons are no longer supported. You must modify users and applications that use SQL Server authentication to use case-sensitive password submissions. |
| Metadata visibility permission is strengthened. | SQL Server 2008 requires users to have the VIEW SERVER STATE or VIEW DATABASE STATE permission to access DMVs and the VIEW (ANY) DEFINITION permission for system catalog metadata. |
| Application roles are constrained to a given database. | Application roles cannot read data from DMVs and cannot read server-level metadata in compatibility views. |
| Schema names are shown in INFORMATION\_SCHEMA views. | SQL Server 2008 schema names (not user names) are now returned in the schema column of INFORMATION\_SCHEMA views. |
| Additional permission is required for sp\_changeobjectowner. | Users executing the sp\_changeobjectowner system stored procedure must also have the CONTROL permission set for target objects as well as membership in the db\_ddladmin and db\_securityadmin fixed database roles. |
| Permission for running sp\_addtype is strengthened. | Users must be a member of the db\_ddladmin or the db\_owner fixed database role to execute the sp\_addtype system stored procedure. |
| Permissions for BCP are strengthened. | Users must have the ALTER permission in addition to the INSERT and SELECT permissions to insert data into a table if they disable CHECK constraints on the target table. Disabling CHECK constraints is the default BCP behavior. |
| User ID range is expanded. | SQL Server 2008 allows higher user IDs, which was not allowed in previous versions of SQL Server. Modify any code that makes assumptions about user ID numbers. Some compatibility views might return error messages on databases with more than 32767 users, groups, and roles. For more information, see [Compatibility views (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=133104) in SQL Server 2008 Books Online. |

#### Metadata Visibility Issues

In addition, a number of behavior changes related to metadata visibility can affect your upgraded applications and need to be addressed. Let's look at each in turn.

**Limited user access to metadata about other users' objects.** By default in SQL Server 2005 and SQL Server 2008, users do not have visibility to metadata about each other's objects. A given user cannot see catalog view metadata about objects belonging to other users unless the user is privileged or has explicit permissions granted, such as CONTROL, IMPERSONATE, ALTER, or VIEW DEFINITION. You can write a data definition language (DDL) trigger to automatically grant those permissions to users when they are newly created.

**Users cannot see each other's metadata.** A given least-privileged user cannot see the metadata of other least-privileged users. For example, least-privileged user User1 cannot grant permissions to least-privileged user User2 because the second user is not visible to User1 in the sys.database\_principals view. If you want to override this behavior, you can write a DDL trigger to automatically grant VIEW DEFINITION permission on newly created users.

**Ownership chaining does not apply to metadata views.** Even if a user has permissions to execute a stored procedure that inserts into another user's table, that user cannot view metadata about that table from sys.sysobjects where xtype = 'U' or the sys.tables catalog view because ownership chaining does not apply to metadata views.

For example, suppose you have a stored procedure that inserts data into a table and calls a metadata view, as the following code shows:

CREATE PROCEDURE S1.P

AS

INSERT INTO S1.T VALUES (1,1);

-- Metadata call

SELECT \* FROM sys.tables WHERE OBJECT\_ID = OBJECT\_ID('S1.T');

RETURN;

User1 owns schema S1 and creates table T in schema S1. User1 grants EXECUTE permission on the procedure S1.P to User2. When User2 executes P, the INSERT will succeed, but the SELECT statement will not return any rows because User2 by default does not have visibility to User1's metadata. Ownership chaining does not apply to metadata views.

To work around this restriction, you can grant callers of the stored procedure VIEW DEFINITION permission to the table or use a certificate-signed procedure and grant VIEW DEFINITION permission to the certificate user. For example, suppose that User1 next executes the following code:

GRANT VIEW DEFINITION ON OBJECT::S1.T TO User2;

Now when User2 executes S1.P, the SELECT operation from sys.tables will return the expected row because User2 has been granted visibility to the metadata for table S1.T.

**By default, the VIEW ANY DATABASE permission is granted to the PUBLIC role.** Therefore, any user can see a list of all databases from the sys.sysdatabases compatibility view or the sys.databases catalog view. To make even the existence of a database invisible to users, revoke the VIEW ANY DATABASE permission from PUBLIC and selectively grant it to users requiring it. A user can always see in sys.databases the databases owned by that user, and members of the sysadmin server role can always see all databases. In addition, the master and tempdb databases are always visible in sys.databases to all users.

**User-schema separation and deprecated system stored procedures can cause different metadata results.** After you upgrade to SQL Server 2008, users and schemas are no longer implicitly connected. This can cause differences in metadata results if you start using the new user-schema features such as the CREATE USER and CREATE SCHEMA DDL statements. Your upgraded applications will continue to function correctly only as long as you continue using the old SQL Server 2000 APIs such as sp\_adduser. In SQL Server 2005 and SQL Server 2008, a given user can own two tables with the same name that exist in separate schemas belonging to that same user—something not possible in SQL Server 2000 and earlier. The catalog view sys.sysobjects will reflect that there are two tables for that user, whereas in earlier versions of SQL Server, only one row would be returned. For example, suppose user Test owns two schemas: S1 and S2. User Test can create a table in each schema called T. User Test inspects sysobjects for that table, using the following query:

SELECT \* FROM sysobjects WHERE name = 'T' and uid = USER\_ID('Test');

In SQL Server 2000, this query could return only one row. Because user and schema were implicitly connected, a user could own only one table named T. However, in SQL Server 2005 and SQL Server 2008, the query returns two rows: one for each schema (S1 and S2).

#### Reversing Metadata Visibility Limitations

If your application requires it, you can reverse the limitations on metadata visibility and restore behavior similar to that of SQL Server 2000. You can increase the permissions for the PUBLIC role by executing the following statements:

GRANT VIEW ANY DEFINITION TO PUBLIC;

GRANT VIEW SERVER STATE TO PUBLIC;

These statements will let any user see the compatibility views as well as all the new metadata available through DMVs and dynamic management functions (DMFs).

#### Reversing Application Role Restrictions

You can also expand the permissions available for an application role, if required, by using the trace flag 4616 or certificates and signed modules. For information about reversing application role restrictions, see [Troubleshooting Metadata Visibility](http://go.microsoft.com/fwlink/?LinkId=133105) in SQL Server 2008 Books Online and the Microsoft Knowledge Base article, [You may receive a "Permission denied" error message when an application role-based application tries to select records from any one of the system tables in a SQL Server 2005 master database](http://go.microsoft.com/fwlink/?LinkId=133106).

### Pre-Upgrade Security Tasks

The following is a suggested list of security-related tasks you should accomplish before upgrading the relational engine to SQL Server 2008. Make sure these tasks are incorporated into your upgrade plan.

1. **Assess the current database security.** Start by assessing the security level of your current system. Make an inventory of the security features for each legacy server, including:

* Type of authentication (Windows only or Windows and SQL Server).
* Vulnerability of passwords (for SQL Server Authentication).
* Results of Microsoft Baseline Security Analyzer (MBSA) or equivalent privileges of service accounts.
* Use of xp\_cmdshell.
* Required auditing.

SQL Server 2008 has enhanced SQL Server login passwords. Database administrators creating and maintaining SQL Server logins now have the ability to apply the local Windows password policy to their SQL Server logins. If possible, you should plan to incorporate stronger passwords and Windows-level password policies in your SQL Server logins.

In addition, you should review applications and scripts that create new SQL Server logins to determine whether the logic of those applications and scripts needs to be modified to account for the new password security enhancements.

1. **Back up databases.** No matter what upgrade method you use, you must make an initial backup of your databases from the server you plan to upgrade. You should also verify the backup file or tape.

Apply the same security considerations to the backup you take before upgrading as you do for any backup: Apply a password to the backup, and store the backup media securely. If you need to transport the media to a secure location, ensure that your method of transfer is also secure and trusted.

1. **Review and resolve issues identified by SQL Server 2008 Upgrade Advisor.** Be sure to run Upgrade Advisor, address all blocking issues, and find solutions for all other issues. Many of the issues might be security-related. You can find some of these security-related issues listed in "Other Database Engine Upgrade Issues" in the Upgrade Advisor Help topic. Chapter 1, "Upgrade Planning and Deployment," covers using Upgrade Advisor as well as SQL Server 2008 Upgrade Assistant and the Best Practices Analyzer for SQL Server 2000 and SQL Server 2005.
2. **Choose an authentication mode.** Whenever possible, use Windows Authentication for user and application connections to SQL Server. This might not be possible for third-party applications, but for users and middle-tier servers, it just makes sense: Why not let Windows manage the passwords and password aging rather than SQL Server?
3. **Determine service account security.** As mentioned earlier in "Service Account Security," you do not need to grant local administrator rights to SQL Server 2008 service accounts. Before you upgrade, you need to determine the rights those accounts will have or let SQL Server Setup do that automatically for you. In either case, you must have the accounts already created.
4. **Choose an upgrade method.** Make sure that the upgrade method you choose will not compromise your security. For example, you might determine that some data is so sensitive that you do not want it to leave the server, but you also do not want to perform an in-place upgrade. If you have enough resources on the server, you could perform a side-by-side upgrade on the same server. For other security considerations for an in-place or side-by-side upgrade, see "In-Place Upgrade" and "Side-by-Side Upgrade" below.
5. **Create, document, and test the upgrade plan.** Good planning is the best method for preventing errors. Not only should you document your upgrade plan, you should test the upgrade steps in a test environment. If you must use production data, ensure that the test environment is at least as secure as the production environment. For details about planning for an upgrade, see Chapter 1, "Upgrade Planning and Deployment."

## Upgrading to SQL Server 2008

Here are the key security considerations for an in-place or side-by-side upgrade.

### In-Place Upgrade

In SQL Server 2008, services and configuration settings are off by default unless required. During an in-place upgrade, the SQL Server 2008 Setup program will maintain the SQL Server 2008 services on until the upgrade is finished. Therefore, you are assured that the upgrade process will not fail for those reasons.

Because an in-place upgrade occurs on the same database server, replacing the legacy SQL Server instances with the new ones, your data remains as secure as the server. There might be a brief time when the legacy SQL Server service has stopped before the SQL Server 2008 Setup program can start the new SQL Server 2008 instance, and for that duration, your data files are not being exclusively used by a SQL Server service. Ensure that your database server's files are secured from outside users attempting to access those files during the upgrade process.

### Side-by-Side Upgrade

During a side-by-side upgrade, you install a new instance of SQL Server 2008 on a new server or along the legacy instance on your current server. In this case, you should ensure that services and configuration options are set in such as way as to guarantee your successful upgrade. For example, the following features are off by default in a new SQL Server 2008 installation to help reduce the attack surface of your new installation: ad hoc distributed queries, OLE automation, SQL Mail, Web Assistant stored procedures, named pipes, and xp\_cmdshell.

If your new installation requires any of these, you need to enable them by using Policy-Based Management features or sp\_configure.

When you move your data from the old instance to the new, you must choose a transfer method such as backup and restore, detach and attach, BCP, DTS, or the Copy Database Wizard. In the case of the first two methods, you will be copying files over some distance, and you must ensure the security of the copy process and the media used in the copy.

If you use the Copy Database Wizard, be aware that your SQL Server logins must be a member of the sysadmin fixed server role on both the source legacy SQL Server instance and on the SQL Server 2008 destination instance. For more information about using the Copy Database Wizard, see [Using the Copy Database Wizard](http://go.microsoft.com/fwlink/?LinkId=133107) in SQL Server 2008 Books Online.

**Note**: If you are using a side-by-side upgrade method, you need to configure the SQL Server 2008 services to match the settings of your source SQL Server instance. In an in-place upgrade, SQL Server 2008 Setup will preserve the services settings of the SQL Server instance you are upgrading, except for the new service for SQL Server 2008 full-text search. Even for an in-place upgrade, service setting for SQL Server 2005 full-text will not be preserved. In an upgrade from SQL Server 2000, you also need to configure the new SQL Server Browser service. For more information, see "Configuring Services and Connections" earlier in this chapter.

## Post-Upgrade Security Tasks

You will need to perform a number of security-related tasks after the upgrade. The following steps can help ensure that your resulting upgraded instance is as secure as possible:

1. **Review service account settings.** Ensure that you have enabled only the services that you need on your upgraded SQL Server 2008 instance. Use the SQL Server Configuration Manager tool to verify the services and settings.
2. **Review configuration settings.** Verify that you have the correct configuration settings, including those that are off by default. Enable only those that are necessary. Use the Policy-Based Management features or sp\_configure to set the correct configuration.
3. **Verify service account security.** Review the Windows privileges given to the service accounts on your new SQL Server 2008 instance, and ensure that they are the minimally required.
4. **Review the authentication mode.** If possible, require Windows Authentication for all connections to SQL Server.
5. **Use strong passwords.** For SQL Server Authentication, require strong passwords for all logins. Also require a strong password for the sa account, even if you are using Windows Authentication. Finally, enable password policy checking.

### Post-Upgrade Security Testing

Your upgrade plan should include a post-upgrade test of your security settings for all databases where data must be secure. There are primarily two ways you can test the security of your upgraded instance of SQL Server: manually or with automated tools.

**Manual testing.** A manual test is an inspection of all the various configuration settings while logged into the server. You can use a checklist of hardening strategies and simply check off the options as you verify them. Settings to look for in such a test would include all the options mentioned in this chapter that affect services and configurations.

**Automated testing.** Another option is to run an automated tool or set of tools to help you probe for vulnerabilities and determine fixes. For example, you can run your standard utilities for assessing SQL Server security, such as the MBSA.

You might find that a combination of manual and automated testing will give you the most confidence in the security level of your upgraded SQL Server 2008 system.

## Conclusion

SQL Server 2008 delivers stronger security and new tools for easier security management. With attention to the new, changed, and discontinued security features we looked at in this chapter, you will be on your way to a smooth transition from SQL Server 2000 or SQL Server 2005 to SQL Server 2008. Just be sure to plan well for the upgrade, make sure you have a secure backup strategy, and test extensively before and after the upgrade to make sure your system is protected and able to take full advantage of the new security capabilities.

## Additional References

For an up-to-date collection of additional references for upgrading database security, see the [[Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkID=132510)](http://msdnstage.redmond.corp.microsoft.com/en-us/library/cc936623.aspx).

For the latest updates to SQL Server 2008 Books Online, see the [Microsoft SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkID=132511).

[SQL Server 2008 Security](http://go.microsoft.com/fwlink/?LinkID=129225)

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkId=132512)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132514)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkId=132515)

# Full-Text Search

## Introduction

SQL Server 2008 provides a new full-text architecture, fully integrated into the Database Engine so that the full-text engine is located in the SQL Server process instead of in a different service as in earlier versions. Integrating the full-text engine into the Database Engine component improves full-text manageability, optimization of mixed queries, and overall performance. This architecture also provides the flexibility to manage all full-text components, including other database objects; to work with the DLL syntax; and to avoid modifications in external components.

Despite this change in architecture, the development team tried to maintain as much compatibility with earlier versions as possible. In fact, although the full-text search engine was completely rewritten, there are few changes in the query syntax model.

This chapter covers the different aspects that you should consider before, during, and after the upgrade process from previous SQL Server versions to SQL Server 2008.

For more information about the new full-text architecture, see [Full-Text Search Architecture](http://go.microsoft.com/fwlink/?LinkId=134531) in SQL Server 2008 Books Online.

## Preparing to Upgrade

You can upgrade a relational database that is enabled for full-text search by performing either an in-place upgrade of the Database Engine and all its databases or by performing a side-by-side database upgrade. With a side-by-side upgrade, you use either the backup/restore method or the detach/attach method, which we cover later in this chapter. For information about choosing between an in-place relational database upgrade and one of the side-by-side relational database upgrade methods, see Chapter 1, "Upgrade Planning and Deployment."

Regardless of whether you choose an in-place upgrade or a side-by-side upgrade of a full-text-enabled relational database, there are a range of potential issues that you might face. Here are the most important full-text search upgrade issues as you move from SQL Server 2000 or SQL Server 2005 to SQL Server 2008. In addition, here are special steps that you have to take if you are upgrading from SQL Server 2000.

For a complete list of deprecated features, discontinued functionality, breaking changes, and behavior changes to full-text search in SQL Server 2008, see [Full-Text Search Backward Compatibility](http://go.microsoft.com/fwlink/?LinkId=134532) in SQL Server 2008 Books Online.

### Deprecated Features

Although there are no full-text search features that will be discontinued in the next release of SQL Server, there are some features that will be removed in a later version. Remember that you can use traces or the new System Monitor object SQLServer:Deprecated Features to check which deprecated features you are using in your applications. For more information, see [SQL Server, Deprecated Features Object](http://go.microsoft.com/fwlink/?LinkID=133121) in SQL Server 2008 Books Online.

The most relevant deprecated features are as follows:

* Replacement of the sp\_fulltext\_catalog procedure. Instead, you should use the new DDL statements CREATE FULLTEXT CATALOG, ALTER FULLTEXT CATALOG, and DROP FULLTEXT CATALOG.
* Replacement of the sp\_fulltext\_column, sp\_fulltext\_database, and sp\_fulltext\_table procedures. The new DDL statements to replace these deprecated stored procedures are CREATE FULLLTEXT INDEX, ALTER FULLTEXT INDEX, and DROP FULLTEXT INDEX.

For a completed list of deprecated full-text search features, see [Deprecated Full-Text Search Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=134570) in SQL Server 2008 Books Online.

### Discontinued Functionality

Table 6-1 describes the most important functionality changes in SQL Server 2008 full-text search that might break custom applications, scripts, or reports. It also describes the replacement features or corrective action that you have to take.

Table 6-1: Discontinued Full-Text Search Functionality

| **Discontinued Feature/Functionality** | **Replacement Feature/Corrective Action** |
| --- | --- |
| **Noise files** - In SQL Server 2008, full-text search uses stopwords and stoplists instead of noise files as in earlier versions. | If you customized the noise word files in the earlier version of SQL Server and you want full-text search to continue using the customized files after upgrade, you must create stoplists and add the stopwords manually. |
| **sp\_fulltext\_table and sp\_fulltext\_column** **stored procedures** – These stored procedures were replaced. | Use the following commands instead:  CREATE/ALTER/DROP FULLTEXT CATALOG  CREATE/ALTER/DROP FULLTEXT INDEX |
| **Full-text indexes on master, model, and tempdb databases** – Full-text indexes on these system databases are not permitted in SQL Server 2005 and SQL Server 2008 and are deleted during an upgrade. | No corrective action is available. |
| **Unsigned third-party components** – The Microsoft Full-Text Engine for SQL Server will not load components that are not signed by Microsoft. | To enable an instance to load a third-party filter after upgrade, use the sp\_fulltext\_service stored procedure to set the service property, *load\_os\_resources* to 1, and set *verify\_signature* to 0 on that instance. |
| **Enabling/disabling full-text search in a database using sp\_fulltext\_database** – In SQL Server 2000, you can enable and disable full-text functionality by using the sp\_fulltext\_database stored procedure. | In SQL Server 2000 and SQL Server 2005, running sp\_fulltext\_database 'enable' on a database that was already full-text enabled caused SQL Server to drop and re-create all the full-text catalogs in the database. By default, all SQL Server 2008 databases have full-text functionality enabled. Using the sp\_fulltext\_database stored procedure with enable or disable parameters will do nothing. |
| **Discontinued full-text search properties** – The following properties were removed from SQL Server 2008:   * DataTimeout * ConnectTimeout * Clean\_up * LogSize | You should remove all references to these properties in your stored procedures and scripts. |

### Breaking Changes

In SQL Server 2000, the MSSearch service was required to run as local system, whereas in SQL Server 2005 the service (known as MSFTSQL) can run under the account of the MSSQLSERVER service or you can specify a different account. When you upgrade to SQL Server 2008, the Microsoft Full-Text Engine now runs under a new low-privilege account that you associate with a new specific security group named SQLServerFDHostUser$<ServerName>$<InstanceName>. You must provide a name and password for this new account at upgrade time.

For more information about breaking changes, see [Breaking Changes to Full-Text Search in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=134571) in SQL Server 2008 Books Online.

### Behavior Changes

There are several behavior changes that might require corrective action after the upgrade is completed. Table 6-2 lists the most important of these changes.

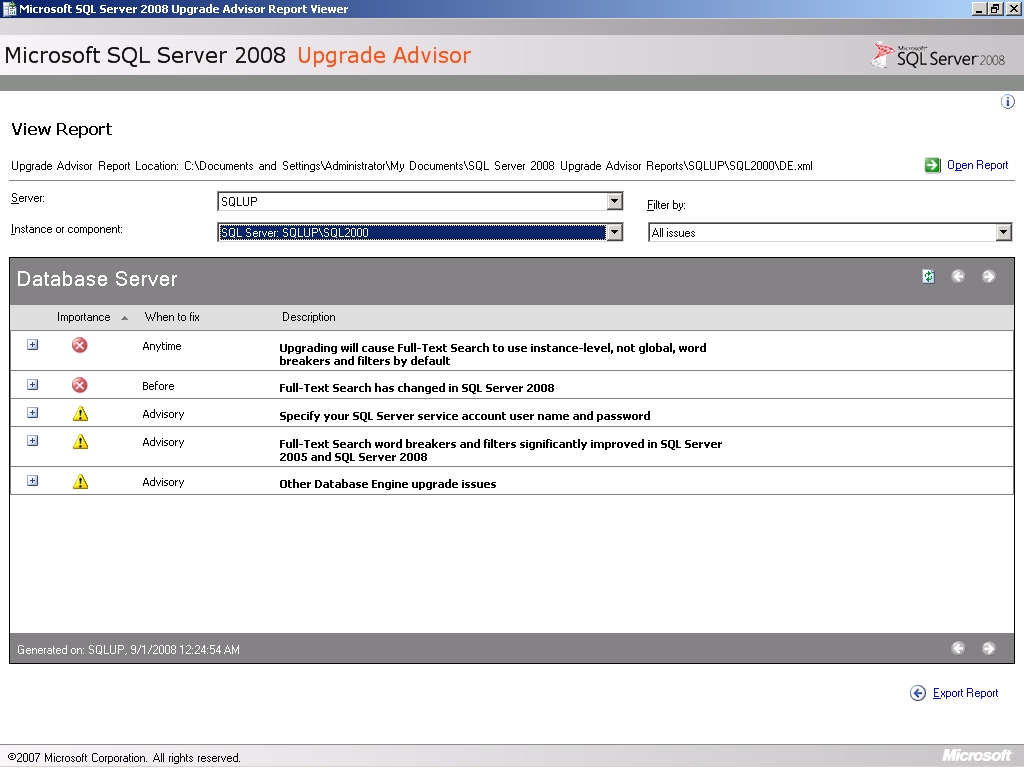
Table 6-2: Behavior Changes That Might Require Corrective Action

| **Behavior Change** | **Description** |
| --- | --- |
| **Word breakers and filters** – SQL Server 2008 has significantly modified word breakers and filters used by full-text search to achieve improvements in functionality and reliability. In certain cases, changes to the word breakers can affect how some data is tokenized. Tokens created in SQL Server 2008might differ from tokens created in SQL Server 2000 or SQL Server 2005. | Changes might require application modification or instance-level word breakers and filters. Semantic inconsistency could arise if full-text indexes are not repopulated, depending on the word breaker used. |
| **Full-text search catalog path** – The path in the sysfulltextcatalogs view and the path returned by the sp\_help\_fulltext\_catalogs and the sp\_help\_fulltext\_catalogs\_cursor system stored procedures has changed. | In SQL Server 2000, a full-text catalog’s path points to its root directory, whereas in SQL Server 2005 its path points to the catalog directory. In SQL Server 2008, because the full-text catalogs are logical objects, there is no file path associated with the full-text search catalog. |
| **Noise word in predicate or function** – In SQL Server 2000, a query that uses a full-text search predicate or function containing a noise word returns an error. In SQL Server 2005 and SQL Server 2008, a similar query returns no rows and a warning is issued. | Modify your application to correctly handle such a warning. |
| **Rows with a rank of 0** – In SQL Server 2000, rows with a rank of 0 are not returned by full-text queries. In SQL Server 2005 and SQL Server 2008, rows with a rank of 0 are returned. This change can affect the results of stored procedures and other queries. | You might have to modify your applications to filter out rows with a rank of 0. |
| **FULLTEXTCATALOGPROPERTY ItemCount returns fewer items** –In SQL Server 2000, ItemCount returns the aggregate of the number of indexed rows in each full-text indexed table in the full-text catalog, plus one for every table in the catalog. In SQL Server 2005, this was changed to accurately reflect the total number of indexed rows in each full-text indexed table in the full-text catalog.  In SQL Server 2008, the ItemCount property was further updated so that it no longer counts NULL rows or BLOB rows with NULL extensions. With this improvement, the final count of items in the full-text catalog includes only the number of rows being successfully aggregated. | Modify stored procedures and scripts to update checks based on the ItemCount property. |

For more information about behavior changes to full-text search, see [Behavior Changes to Full-Text Search in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=134573) in SQL Server 2008 Books Online.

### Running Upgrade Advisor

To obtain a report that identifies many of these potential issues before you start an upgrade, you should run the SQL Server 2008 Upgrade Advisor to analyze the relational database that you want to upgrade. Figure 6-1 shows a sample Upgrade Advisor report. For information about how to install and run this tool, see Chapter 1, "Upgrade Planning and Deployment."



**Figure 6-1:** Sample SQL Server 2008 Upgrade Advisor report

Be aware that there is also a category of issues that either cannot be detected by Upgrade Advisor or whose detection would result in too many false-positive results. For a complete list of full-text search upgrade issues that Upgrade Advisor detects, review the "Full-Text Search Upgrade Issues" topic in the Upgrade Advisor Help file.

Chapter 1 also covers running the SQL Server 2008 Upgrade Assistant and SQL Server 2000 and SQL Server 2005 versions of Best Practices Analyzer to prepare for an upgrade.

### Preparing for a Possible Rollback

Before you start an upgrade of a database that has full-text search enabled, take steps to make sure that you can roll back a failed upgrade if it is necessary and that sufficient space exists in the database for the upgrade to complete successfully. Although the in-place upgrade process was designed and tested to handle most situations, unforeseen problems might occur and result in a failed upgrade. In extreme cases, a failed upgrade can even result in an unusable SQL Server 2000 or SQL Server 2005 installation. Therefore, planning for a failed upgrade process is important.

A side-by-side upgrade of an existing full-text enabled database to SQL Server 2008 should not encounter the same kinds of problems that can affect an in-place upgrade. However, you should follow similar steps to make sure the ability to roll back if it is necessary.

If a failed in-place upgrade occurs, frequently the easiest resolution is to reinstall SQL Server 2000 or SQL Server 2005 and restore the installation to its state before the upgrade process began. To make sure that all the data and configuration files that are needed to restore the existing installation are available, complete the steps outlined in Chapter 3, "Relational Databases," in addition to the following steps before the upgrade process starts:

1. Review requirements to determine whether your hardware and software can support SQL Server 2008. For more information, see [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132172) in SQL Server 2008 Books Online.
2. Use System Configuration Checker (SCC) to scan the server for any conditions that might prevent a successful installation of SQL Server 2008. For more information, see [Check Parameters for the System Configuration Checker](http://msdn.microsoft.com/en-us/library/ms143753.aspx) in SQL Server 2008 Books Online.
3. Review security best practices and guidance for SQL Server. For more information, see [Security Considerations for a SQL Server Installation](http://msdn.microsoft.com/en-us/library/ms144228.aspx) in SQL Server 2008 Books Online.
4. Run Upgrade Advisor on the server to determine any issues that might prevent you from successfully upgrading. For more information, see [Using Upgrade Advisor to Prepare for Upgrades](http://go.microsoft.com/fwlink/?LinkID=132624) in SQL Server 2008 Books Online.
5. Make sure that the file group associated with the base table of a full-text index has sufficient space to accommodate the additional space that is required by SQL Server 2008 full-text indexes.

### Additional Preparation Steps When Upgrading from SQL Server 2000

If you plan to upgrade a SQL Server 2000 full-text-enabled database, you also have to perform the following preparation tasks in addition to those we covered earlier:

1. On a stand-alone computer, stop the Microsoft Search service (but leave it running on a clustered SQL Server configuration).
2. Back up the full-text catalogs, folders, and files in the Windows file system by using any supported method for backing up file system files.
3. Back up the full-text search registry entries. For more information, see [How to Move, Copy, and Backup Full-Text Catalog Folders and Files](http://go.microsoft.com/fwlink/?LinkId=134574) in the Microsoft Knowledge Base.

## Upgrading a Full-Text-Enabled Database

You can choose between two options when you upgrade a full-text enabled database to SQL Server 2008: in-place or side-by-side upgrade.

### In-Place Upgrade

For an in-place upgrade, an instance of SQL Server 2008 is set up side-by-side with the old version of SQL Server 2000 or SQL Server 2005, and data is moved. If the old version of SQL Server had full-text search installed, a new version of full-text search is automatically installed. Side-by-side install means that the following components exist at the instance-level of SQL Server:

* **Word breakers, stemmers, and filters**. Each instance now uses its own set of word breakers, stemmers, and filters, instead of relying on the operating system version of these components. These components are also easier to register and configure at a per-instance level. For more information, see [Word Breakers and Stemmers](http://msdn.microsoft.com/en-us/library/ms142509.aspx) and [Full-Text Search Filters](http://go.microsoft.com/fwlink/?LinkId=134593) in SQL Server 2008 Books Online.
* **Filter daemon host**. The full-text filter daemon hosts are processes that safely load and drive third-party extensible components used for indexing and querying, such as word breakers, stemmers, and filters, without compromising the integrity of the Full-Text Engine. A server instance uses a multithreaded process for all multithreaded filters and a single-threaded process for all single-threaded filters.

**Note**: SQL Server 2008 introduces a service account for the FDHOST Launcher service (MSSQLFDLauncher). This service propagates the service account information to the filter daemon host processes of a specific instance of SQL Server. For information about how to set the service account, see [How to: Set the FDHOST Launcher (MSSQLFDLauncher) Service Account for Full-Text Search (SQL Server Configuration Manager)](http://go.microsoft.com/fwlink/?LinkId=134592) in SQL Server 2008 Books Online.

In SQL Server 2005 and earlier versions, each full-text index is located in a full-text catalog that belongs to a file group, has a physical path, and is treated as a database file. In SQL Server 2008, a full-text catalog is a logical concept—a virtual object—that refers to a group of full-text indexes. Therefore, a new full-text catalog is not treated as a database file that has a physical path. However, during an upgrade of any full-text catalog that contains data files, a new file group is created on the same disk. This maintains the old disk I/O behavior after upgrade. Any full-text index from that catalog is put in the new file group if the root path exists. If the old full-text catalog path is invalid, the upgrade keeps the full-text index in the same file group as the base table or, for a partitioned table, in the primary file group.

#### In-Place Upgrade of Full-Text Search Databases with Third-Party Filters

By default, the full-text engine will not load components that are not signed by Microsoft. To perform an in-place upgrade of full-text search databases with third-party filters, following these steps:

1. Use the sp\_fulltext\_service stored procedure to set the service property, *load\_os\_resources*, for the third-party filter.
2. Turn off the Verify\_signature option.
3. Restart full-text population.

**Note**: For more information about this topic, see the "Discontinued Functionality" section earlier.

### Side-by-Side Upgrade

For a side-by-side upgrade, an instance of SQL Server 2008 is set up side-by-side with the old version of SQL Server on the same or another server, and you move data to the new version.

#### Full-Text Upgrade Option

SQL Server 2008 provides a new instance option, Full-Text Upgrade, which lets you control how SQL Server manages full-text indexes in side-by-side upgrade scenarios. This full-text upgrade option has three possible values:

* **Import.** This is the default option after the setup of a SQL Server 2008 instance and works only for SQL Server 2005 databases. If this option is enabled, SQL Server 2008 tries to import the data in the full-text indexes without resetting or rebuilding them and only copies the data from the old index structures to the new one. Import is the fastest option for an upgrade. But for a set of specific new word breakers, Microsoft cannot guarantee that your queries will return the same results as the SQL Server 2005 version (see the "Semantic Consistency" section in this topic). At import time, this option does not use SQL Server 2008 new and improved word breakers and stemmers. If you try to upgrade a SQL Server 2000 database that has the Import upgrade option selected, SQL Server 2008 will rebuild all full-text indexes.
* **Rebuild.** With this option, SQL Server 2008 will rebuild all full-text indexes, triggering a full population and using the new and improved word breakers. This option can take significant time and could be very CPU- and memory-intensive, depending on the number and size of your full-text indexes. This option guarantees semantic consistency and, in some cases, specific internal optimizations that can improve overall performance later.
* **Reset.** The Reset option gives you more control over the overall total upgrade time because no full-text population or rebuilding will occur. When this option is enabled, SQL Server 2008 deletes the existing full-text catalogs, full-text indexes are disabled for change tracking, and crawls are not started automatically. You can change this option by using the sp\_fulltext\_service stored procedure, as follows:

EXEC master.dbo.sp\_fulltext\_service @action=N'upgrade\_option', @value=1

You can also change this option graphically from SQL Server Management Studio (SSMS) through the instance properties on the Advanced tab.

#### Semantic Consistency

When you import your existing full-text indexes into SQL Server 2008, the data in these indexes is obtained by using the old word breakers and stemmers. But when you query the new SQL Server 2008 full-text index, the full-text engine will use new word breakers so that results of the queries could change. This behavior is known as semantic consistency. But SQL Server 2008 does not improve all word breakers. Table 6-3 shows the word breakers that have not changed from the earlier version so that semantic inconsistency does not apply to them.

Table 6-3: Word Breakers That Haven’t Changed

|  |
| --- |
| Chinese (Hong Kong SAR, PRC) |
| Chinese (Macau SAR) |
| Chinese (Singapore) |
| Danish |
| English |
| English (United Kingdom) |
| Korean |
| Polish |
| Simplified Chinese |
| Thai |
| Traditional Chinese |
| Turkish |

For more information about semantic consistency, review the "Upgrade Option: Semantic Consistency" section of the [SQL Server 2008 Full-Text Search: Internals and Enhancements](http://go.microsoft.com/fwlink/?LinkID=129544) white paper.

#### Considerations for Choosing a Full-Text Upgrade Option

When you choose the appropriate strategy for an upgrade of full-text search, consider the following questions:

* How do you use word breakers?

The SQL Server 2008 full-text search service includes new word breakers and stemmers. These might change the results of full-text queries from previous releases for a specific text pattern or scenario. Therefore, how you use word breakers is important when you choose a suitable upgrade option:

* + If the word breakers of the full-text language that you use did not change in SQL Server 2008, or if recall accuracy is not important to you, using the Import option is suitable. Later, if you experience any recall issues, you can upgrade to the new word breakers by rebuilding your full-text catalogs.
  + If you care about recall accuracy and you use one of the word breakers that were improved in SQL Server 2008, the Rebuild option is suitable.
* Were any full-text indexes built on integer full-text key columns?

Rebuilding performs internal optimizations that improve the query performance of the upgraded full-text index in some cases. Specifically, if you have full-text catalogs that contain full-text indexes for which the full-text key column of the base table is an integer data type, rebuilding achieves ideal performance of full-text queries after upgrade. In this case, we highly recommend that you use the Rebuild option.

* What is the priority for bringing the server instance online?

Importing or rebuilding during upgrade takes a lot of CPU resources, which delays getting the rest of the server instance upgraded and online. If bringing the server instance online as soon as possible is important and if you are willing to run a manual population after the upgrade, the Reset option is suitable.

#### Backup and Restore Method

The first option for doing a side-by-side upgrade is to perform a restore of a SQL Server 2000 or SQL Server 2005 full-text-enabled database in a SQL Server 2008 instance. The main difference between SQL Server 2000 and SQL Server 2005 backups is the content of the backups. A SQL Server 2000 database backup does not contain the full-text index data, but this is not an issue for a restore in SQL Server 2008. Remember that the Import upgrade option is not supported in SQL Server 2000 so that you cannot import or copy the contents of a SQL Server 2000 full-text index into a SQL Server 2008 full-text index.

When you restore the database on SQL Server 2008, a new database file will be created for the full-text catalog. The default name of this file is ftrow\_catalog-name.ndf. For example, if your catalog-name is cat1, the default name of the SQL Server 2008 database file would be ftrow\_cat1.ndf. If the default name is already being used in the target directory, the new database file would be named ftrow\_catalog-name{GUID}.ndf, where GUID is the globally unique identifier of the new file.

After the catalogs are imported, sys.database\_files and sys.master\_files are updated to remove the catalog entries, and the path column in sys.fulltext\_catalogs is set to NULL.

#### Detach and Attach Method

When you attach a full-text-enabled database to SQL Server 2008, catalog files are attached from their previous locations together with the other database files. If SQL Server 2008 cannot find a full-text catalog file or if the full-text file was moved during the attach operation without someone specifying a new location, the behavior depends on the selected full-text upgrade option. If the full-text upgrade option is Import or Rebuild, the attached full-text catalog is rebuilt. If the full-text upgrade option is Reset, the attached full-text catalog is reset.

The state of each attached full-text catalog on SQL Server 2008 is the same as when the database was detached from SQL Server 2005. If any full-text index population was suspended by the detach operation, the population is resumed on SQL Server 2008 even if the full-text upgrade option is configured as Import.

**Note:** If you attach SQL Server 2000 database files to a SQL Server 2008 instance, there is no full-text catalog data in these files. As described in the "Backup and Restore Method" section earlier, you must populate full-text indexes in SQL Server 2008.

#### Side-by-Side Upgrade with Third-Party Filters

By default, the full-text engine will not load components that are not signed by Microsoft. If you are performing a side-by-side upgrade of a full-text enabled database that has third-party filters, perform the following additional steps:

1. Install the third-party filter in the SQL Server 2008 server.
2. Use the sp\_fulltext\_service stored procedure to set the service property, *load\_os\_resources*, for the third-party filter.
3. Turn off the Verify\_signature option.
4. Start the upgrade by using the selected side-by-side upgrade method.

## Post-Upgrade Tasks

It is a good practice to check the crawl log right after you upgrade to make sure that the crawl is running without a problem and that the full-text population is complete. Here are some post-upgrade tasks related to custom noise words that you might have to perform.

### Using Customized Noise Word Files from a Previous SQL Server Version

SQL Server noise words were replaced by SQL Server 2008 stopwords. When you upgrade a database from a previous release of SQL Server to SQL Server 2008, the noise-word files are no longer used in SQL Server 2008. However, the old noise-word files are stored in the FTDATA\FTNoiseThresaurusBak folder, and you can use them later when you update or building corresponding SQL Server 2008 stoplists.

After the upgrade:

* If you never added, modified, or deleted any noise-word files in your installation of SQL Server 2000 or SQL Server 2005, the system stoplist should meet your needs.
* If you modified your noise-word files in the previous SQL Server version, those modifications are lost during upgrade. To re-create those updates, you must manually re-create those modifications in the corresponding SQL Server 2008 stoplist.
* If you do not want to apply any stopwords to your full-text indexes (for example, if you deleted or erased your noise-word files in the earlier version installation), you must turn off the stoplist for each upgraded full-text index.

Be aware that SQL Server 2008 does not create stoplists to implement noise-word files in any upgrade scenario so that you have to create them manually. For more information, see [Stopwords and Stoplists](http://go.microsoft.com/fwlink/?LinkId=134576) in SQL Server 2008 Books Online.

## Conclusion

Full-text search is now fully integrated in the SQL Server 2008 Database Engine, and you upgrade full-text indexes by using the same process as for the base database. SQL Server 2008 also provides a new Full-Text Upgrade option that lets you control how the full-text engine should work with the indexes. By using this option, you can choose the best approach for each scenario, maximizing the performance of the upgrade process in addition to uptime.

## Additional Resources

For an up-to-date collection of additional references for upgrading SQL Server 2008, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkId=132510).

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkId=132512)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132514)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkId=132515)

# Service Broker

## Introduction

SQL Server 2005 introduced Service Broker, a technology that helps database developers build secure, reliable, and scalable applications. Because Service Broker is part of the Database Engine, administration of these applications is part of the routine administration of the database.

Service Broker provides queuing and reliable messaging for SQL Server by implementing a transaction-oriented queue directly within the database. You can use Service Broker for applications that use a single instance of SQL Server, and for those that distribute work across multiple instances. Within a single instance of SQL Server, Service Broker provides a robust, asynchronous programming model. Database applications typically use asynchronous programming to shorten interactive response time and increase overall application throughput. And for applications that work across multiple instances of SQL Server, Service Broker provides reliable messaging between the instances, and helps developers compose applications from independent, self-contained components ("services"). Applications that require the functionality that is available in these services use messages to interact with the services. Service Broker uses TCP/IP to exchange messages between instances and includes features to help prevent unauthorized access from the network and to encrypt messages sent across the network.

Service Broker in SQL Server 2008 retains all the features and functionality from SQL Server 2005, and there are no behavior changes that would affect an upgrade. However, SQL Server 2008 adds some new features that you should know about, and there are a few things to keep in mind for a smooth upgrade. This chapter describes the key Service Broker considerations for upgrading from SQL Server 2005 to SQL Server 2008.

## Feature Changes

The major new features for Service Broker in SQL Server 2008 are as follows:

* The introduction of conversation priority
* A new diagnostic tool
* More detailed support in SQL Server Management Studio (SSMS)
* New Windows System Monitor objects and counters
* New tutorials to help you get started with Service Broker
* More flexible activation options

This guide is focused on upgrade. Therefore, exploring these new features is not discussed here. In the "Post-Upgrade Tasks" section later in this chapter, we discuss some conversation priority changes you might want to make after your upgrade.

For information about architecting effective SQL Server 2008 Service Broker solutions, see [Planning and Architecture (Service Broker)](http://go.microsoft.com/fwlink/?LinkId=132585) in SQL Server 2008 Books Online.

## Preparing to Upgrade

Because Service Broker is part of the SQL Server Database Engine component, you have the same upgrade options available: in-place or side-by-side. For more information, see Chapter 1, "Upgrade Planning and Deployment," We look at each of these upgrade methods as they relate to Service Broker later in this chapter, but if you decide to perform an in-place upgrade, consider preparing for the move by processing all the data in existing queues before the upgrade. You do not have to process the data in existing queues, but doing so might reduce the resource requirements during the upgrade.

### Disk Space Requirements

As part of its new Service Broker conversation priority feature, SQL Server 2008 Setup must upgrade the service queue to rebuild the clustered index on the queue internal table. Rebuilding the index on each queue requires space that depends on the volume of data (messages) in the queue. Because Setup upgrades all the queues in a single transaction, Setup needs as much free disk space as the sum of the clustered index sizes on all the queues in the database. If the transaction fails because of a lack of disk space, the database might become unusable.

Before you perform an in-place upgrade, back up the database, and run the following queries to obtain the required space to upgrade Service Broker:

select \* from sys.service\_queues -- Returns list of Service Broker queues

sp\_spaceused <Name from above query>

### Upgrade Tools

You can use several tools to help you prepare for a successful upgrade. The first helps you find and fix issues that could prevent an upgrade, and identifies items to modify after your move to SQL Server 2008. The second, an external tool, helps you identify compatibility issues between SQL Server versions. The third helps ensure that you are using best practices on your current system so that you have fewer changes to make when you upgrade to SQL Server 2008.

#### Running Upgrade Advisor

SQL Server 2008 Upgrade Advisor helps you prepare for upgrades to SQL Server 2008. Upgrade Advisor analyzes installed components from earlier versions of SQL Server, and then generates a report that identifies issues to fix either before or after you upgrade. For more information, see Chapter 1, "Upgrade Planning and Deployment."

From the Upgrade Advisor Home screen, you can run the following tools:

* Upgrade Advisor Analysis Wizard
* Upgrade Advisor Report Viewer
* Upgrade Advisor Help

The first time you use Upgrade Advisor, run the Upgrade Advisor Analysis Wizard to analyze SQL Server components. When the wizard finishes its analysis, view the resulting reports in the Upgrade Advisor Report Viewer. Each report provides links to information in Upgrade Advisor Help that will help you fix or reduce the effect of the known issues.

Upgrade Advisor analyzes the following SQL Server components:

* Database Engine
* Analysis Services
* Reporting Services
* Integration Services
* Data Transformation Services

You can download Upgrade Advisor at [Microsoft SQL Server 2008 Feature Pack, August 2008](http://go.microsoft.com/fwlink/?LinkID=131220).

#### Running SQL Server 2008 Upgrade Assistant

The SQL Server 2008 Upgrade Assistant is an external tool that lets you determine in a test environment how an application that is currently running on SQL Server 2000 or SQL Server 2005 will run on SQL Server 2008. This tool uses Upgrade Advisor, along with baseline and trace replay, to help identify compatibility issues. For more information, see [SQL Server 2008 Upgrade Assistant](http://go.microsoft.com/fwlink/?LinkID=133150).

#### Running Best Practices Analyzer

Before upgrading your system, we recommend that you use best practices for your existing system by running SQL Server Best Practices Analyzer (BPA). BPA is available for SQL Server 2000 and for SQL Server 2005. SQL Server 2005 BPA gathers data from Windows and SQL Server configuration settings, using a predefined list of SQL Server 2005 recommendations and best practices to determine if there are potential issues in the database environment. Running BPA before upgrading gives you the opportunity to fix any problems and helps ensure that you are using best practices before you go to the new system.

You can download the [SQL Server 2005 BPA](http://go.microsoft.com/fwlink/?LinkId=132957) at the Microsoft Download Center.

### 64-bit Considerations

Table 7-1 shows the supported architectures for SQL Server 2008 Service Broker.

Table 7-1: Architectures Supported in SQL Server 2008 Service Broker

|  |  |
| --- | --- |
| **Architecture** | **Supported** |
| X86 (32 bit) | Yes |
| X64 | Yes |
| IA64 | Yes |

## Upgrading from SQL Server 2005

There are a few Service Broker-related items to keep in mind during your upgrade from SQL Server 2005 to SQL Server 2008.

### In-Place Upgrade

Service Broker operations do not change when a database or an instance of the Database Engine is upgraded from SQL Server 2005 to SQL Server 2008. The Service Broker features available in SQL Server 2005 have the same behavior in SQL Server 2008.

SQL Server 2005 databases are upgraded to SQL Server 2008 when the following are true:

* They are attached to an instance of the SQL Server 2008 Database Engine after they are detached from an instance of the SQL Server 2005 Database Engine.
* The instance of the Database Engine they are in is upgraded from SQL Server 2005 to SQL Server 2008.

You do not have to process existing data/messages in the queues before upgrade. However, doing so might reduce the disk space required for the upgrade, as noted in the earlier "Preparing to Upgrade" section. You can upgrade servers in any order.

### Side-by-Side Upgrade

#### Routing

Although there are typically no special requirements related to Service Broker in addition to those required for SQL Server itself, if you are performing a side-by-side upgrade and have server instance name changes or service name changes, you must plan to resolve any service routing issues as part of the upgrade. If name changes will relate to messages or conversations already in progress, you must consider how and when those messages will be processed. For more information, see [Service Broker Routing and Networking](http://go.microsoft.com/fwlink/?LinkId=133000) in SQL Server 2008 Books Online.

#### Object Names and Collation

Service Broker is designed to let services and applications in instances with different collation configurations communicate easily and efficiently. The database that hosts a service which sends a message might not use the same collation as the database that hosts the service which receives the message. Therefore, Service Broker uses a consistent collation for names, regardless of the collation of the database that hosts the service. To remove collation information from the communication process, Service Broker uses a byte-by-byte comparison to match service names, contract names, and message type names. By matching names as sequences of bytes, Service Broker makes it simple for services to exchange messages correctly without the extra overhead of exchanging collation information.

Therefore, it is important that when you perform an upgrade of Service Broker that requires objects to be recreated via scripts, you should maintain the exact same object names in a byte-by-byte comparison. For more information, see [Understanding Collation and Service Broker](http://go.microsoft.com/fwlink/?LinkId=133001) in SQL Server 2008 Books Online.

#### Preserving Settings

In a side-by-side upgrade, you detach databases from SQL Server 2005 and attach them to SQL Server 2008. The result is that Service Broker is disabled, meaning that the following bits are set to 0 and you must note which ones are 1 before detaching:

* is\_broker\_enabled
* is\_honor\_broker\_priority\_on
* is\_trustworthy\_on

You can see these setting for all databases by running the following query:

SELECT \* FROM sys.databases

## Post-Upgrade Tasks

After the upgrade to SQL Server 2008, consider performing the following Service Broker-related tasks.

### Restoring Settings

You will have to restore the following settings after the upgrade:

* is\_broker\_enabled
* is\_honor\_broker\_priority\_on
* is\_trustworthy\_on

You can restore the settings by using the ALTER DATABASE command.

### Routing Changes

If name changes are part of the upgrade process, you will probably have to change Service Broker routing configurations after the upgrade.

### Implementing Conversation Priorities

SQL Server 2008 introduces conversation priorities, a set of user-defined rules that specifies priority levels and the criteria for determining which Service Broker conversations to assign to each priority level. Typically, messages from conversations that have high priority levels are sent or received before messages from conversations with low priority levels.

When a SQL Server 2005 database is upgraded to SQL Server 2008, conversations continue to operate as they did in SQL Server 2005, but the system objects are built to support conversation priorities, as follows:

* The upgrade process builds the new system objects that are required to support conversation priorities. It adds conversation priority columns to existing system tables, views, trace events, and performance counters.
* By default, the HONOR\_BROKER\_PRIORITY database option is OFF.
* All existing messages in service queues have their priority level set to 10. This means they will be the first messages retrieved by RECEIVE statements.
* By default, all conversation endpoints in the upgraded database are assigned the conversation priority of 5.

You can start to use conversation priorities in an upgraded database by doing the following:

1. Use the ALTER DATABASE statement to set the HONOR\_BROKER\_PRIORITY database option to ON.
2. Use the CREATE BROKER PRIORITY statement to define a set of conversation priorities in the database.

For more information, see [Conversation Priorities](http://go.microsoft.com/fwlink/?LinkId=133002) in SQL Server 2008 Books Online.

## Conclusion

Upgrading SQL Server 2005 Service Broker to SQL Server 2008 Service Broker can be a straightforward process. But first, you must ensure that sufficient disk space is available, and consider the routing of existing messages or conversations. You can implement conversation priorities after the upgrade is complete.

## Additional References

For an up-to-date collection of additional references for upgrading Service Broker to SQL Server 2008, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkID=132510).

For the latest updates to SQL Server 2008 Books Online, see the [Microsoft SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkID=132511).

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkID=132515)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=133003)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkID=116426)

# Transact-SQL Queries

## Introduction

A significant but often overlooked component of every upgrade process is upgrading queries and scripts. And although most database administrators expect the normal upgrade process to upgrade their stored procedures, they fail to realize that new releases of SQL Server contain both subtle and dramatic changes that will affect most query-intensive environments. This chapter covers changes in SQL Server 2008 that might affect your stored procedures, queries, scripts, and applications.

Whether you choose to perform an in-place upgrade or a side-by-side upgrade to the new SQL Server release, the stored procedures in your database will remain in the database after you upgrade it. However, unlike other components in SQL Server, stored procedures will not automatically be upgraded to new functionality when you execute them or when the database is upgraded. You will have to rewrite them to take advantage of new Transact-SQL features in SQL Server 2008. The same is true for upgrading queries embedded in your applications.

In this chapter, we look at the key query-related issues that could prevent an upgrade as well as other changes that might affect how your stored procedures, queries, scripts, and applications behave after an upgrade. You must manually review stored procedures, ad hoc queries, and scripts used against the database for any upgrade issues before you begin the upgrade process. However, you can significantly reduce the amount of manual review you need to perform by executing the Microsoft SQL Server 2008 Upgrade Advisor against SQL Server 2000 and SQL Server 2005 databases and resolving any issues Upgrade Advisor reports.

## Preparing to Upgrade

Preparation is the key to a successful upgrade of your stored procedures, ad hoc queries, and administrative scripts. Having a backup and rollback plan is critical, and you need to know which features have been deprecated or discontinued in SQL Server 2008 as well as which features could derail an upgrade or cause problematic changes in behavior after your upgrade. In this section, we discuss the key Transact-SQL changes you need to understand for a smooth transition to SQL Server 2008, and the tools that can help you identify and resolve possible problems.

### Backup and Rollback Plan

Before any upgrade, it is essential that you back up your existing databases to allow for a quick and easy rollback in the event of complications. Furthermore, before upgrading a production system, make sure to perform all upgrade steps in a development-type environment to help identify and address upgrade issues before attempting a production upgrade.

In addition to performing a backup, you should also run DBCC CHECKDB on the database before upgrading to ensure that the database you are upgrading is not damaged.

### Deprecated Features

Deprecated features are those that SQL Server 2008 still supports but that will be removed in the next version of SQL Server. Although you do not have to remove these features from your implementation to complete an upgrade, you need to address them to make sure you avoid problems in the future. Microsoft has deprecated several Transact-SQL features in SQL Server 2008, including system tables and the UPDATETEXT function. The version also includes a new System Monitor counter to help you identify deprecated features on your upgraded system.

#### System Tables

SQL Server 2000 system tables have been deprecated and replaced by system views and dynamic management views (DMVs) in SQL Server 2005 and SQL Server 2008. For a complete mapping of legacy system tables to system views and DMVs, see [Mapping System Tables to System Views (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkID=133091).

#### UPDATETEXT

The UPDATETEXT function is deprecated in SQL Server 2008. Future versions of SQL Server will no longer support UPDATETEXT. You should modify all UPDATETEXT references to use the .WRITE clause of the UPDATE statement to prevent any future problems. For detailed information about UPDATETEXT, see [UPDATE (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=133119) in SQL Server 2008 Books Online.

#### SQLServer:Deprecated Features Object

After you have upgraded to SQL Server 2008, you might want to identify deprecated features still in use so that you can address those issues and replace them with the newer SQL Server 2008 counterparts. SQL Server 2008 introduces a new System Monitor object called SQLServer:Deprecated Features. This object enumerates a number of deprecated features and how frequently they are used on your SQL Server system. You can view this information either via System Monitor or by using the DMV sys.dm\_os\_performance\_counters. For more information about this object, see [SQL Server, Deprecated Features Object](http://go.microsoft.com/fwlink/?LinkId=133121) in SQL Server 2008 Books Online.

Table 8-1 lists deprecated features, which will not be available after SQL Server 2008.

Table 8-1: Features Not Supported in the Next Version of SQL Server

| Category | Deprecated Feature | Replacement |
| --- | --- | --- |
| Backup and restore | BACKUP { DATABASE | LOG } WITH PASSWORD | None |
| Backup and restore | BACKUP { DATABASE | LOG } WITH MEDIAPASSWORD | None |
| Backup and Restore | RESTORE { DATABASE | LOG } … WITH DBO\_ONLY | RESTORE { DATABASE | LOG } … … WITH RESTRICTED\_USER |
| Backup and restore | RESTORE { DATABASE | LOG } WITH PASSWORD | None |
| Backup and restore | RESTORE { DATABASE | LOG } WITH MEDIAPASSWORD | None |
| Compatibility levels | 80 compatibility level and upgrade from version 80 | Compatibility levels are available only for the last two versions. For more information about compatibility levels, see [ALTER DATABASE Compatibility Level (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=133122) in SQL Server 2008 Books Online. |
| Metadata | DATABASEPROPERTY | DATABASEPROPERTYEX |
| Database objects | WITH APPEND clause on triggers | Recreate the whole trigger. |
| Instance options | Default setting of disallow results from triggers option = 0 | Default setting of disallow results from triggers option = 1 |
| Database options | sp\_dboption | ALTER DATABASE |
| Query hints | FASTFIRSTROW hint | OPTION (FAST n) |
| Remote servers | sp\_addremotelogin  sp\_addserver  sp\_dropremotelogin  sp\_helpremotelogin  sp\_remoteoption | Replace remote servers by using linked servers. |
| Remote servers | @@remserver | Replace remote servers by using linked servers. |
| Remote servers | SET REMOTE\_PROC\_TRANSACTIONS | Replace remote servers by using linked servers. |
| Security | sp\_dropalias | Replace aliases with a combination of user accounts and database roles. Use sp\_dropalias to remove aliases in upgraded databases. |
| SET options | SET DISABLE\_DEF\_CNST\_CHK | None; option has no effect. |
| SET options | SET ROWCOUNT for INSERT, UPDATE, and DELETE statements | TOP keyword |
| Transact-SQL syntax | Use of \*= and =\* | Use ANSI join syntax. For more information, see [FROM (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=133123) in SQL Server 2008 Books Online. |
| Transact-SQL syntax | COMPUTE / COMPUTE BY | Use ROLLUP. |
| System tables | sys.database\_principal\_aliases | Use roles instead of aliases. |
| Transact-SQL | The RAISERROR (Format: RAISERROR integer string) syntax | Rewrite the statement using the current RAISERROR syntax. |
| Other | DB-Library  Embedded SQL for C | Although the Database Engine still supports connections from existing applications that use the DB-Library and Embedded SQL APIs, it does not include the files or documentation required to do programming work on applications that use these APIs. A future version of the SQL Server Database Engine will drop support for connections from DB-Library or Embedded SQL applications. Do not use DB-Library or Embedded SQL to develop new applications. Remove any dependencies on either DB-Library or Embedded SQL when you are modifying existing applications. Instead of these APIs, use the SQLClient namespace or an API such as OLE DB or ODBC. SQL Server 2008 does not include the DB-Library DLL required to run these applications. To run DB-Library or Embedded SQL applications, you must have available the DB-Library DLL from SQL Server 6.5, SQL Server 7.0, or SQL Server 2000. |

For a comprehensive list of deprecated functionality in SQL Server 2005 and SQL Server 2008, see the following topics in SQL Server Books Online:

* [Deprecated Database Engine Features in SQL Server 2005](http://go.microsoft.com/fwlink/?LinkId=133124)
* [Deprecated Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=133092)

### Discontinued Functionality

SQL Server 2008 has discontinued some stored procedure and Transact-SQL command functionality from previous versions of SQL Server, including the syslocks virtual table.

#### syslocks Virtual Table

The syslocks virtual table has been discontinued as of SQL Server 2005. All references to this object should be modified to use the new sys.dm\_tran\_locks DMV. For a complete list of deprecated or discontinued system views and their replacements, see [Mapping System Tables to System Views (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkID=133091) in SQL Server 2008 Books Online.

Table 8-2 lists the commands that have been discontinued in SQL Server 2008. You should change references to these commands in all stored procedures, ad hoc queries, and scripts.

Table 8-2: Discontinued Functionality

| **Category** | **Discontinued Feature** | **Replacement** |
| --- | --- | --- |
| Aliases | sp\_addalias | Replace aliases with a combination of user accounts and database roles. For details, see [CREATE USER (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=133125) and [CREATE ROLE (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=133126) in SQL Server 2008 Books Online. Remove aliases in upgraded databases by using sp\_dropalias. |
| APIs | Registered Servers API | Replaced by a new registered servers API that supports new SQL Server 2008 features. |
| APIs | SQL Namespace API (SQL-NS) | None. |
| APIs | SQL-DMO based WMI provider | For details, see [Managed code: Microsoft.SqlServer.Management.Smo.Wmi](http://go.microsoft.com/fwlink/?LinkId=133127). |
| Backup and restore | BACKUP LOG WITH NO\_LOG | None. The transaction log is automatically truncated when the database is using the Simple recovery model. If you must remove the log backup chain from a database, switch to the Simple recovery model. |
| Backup and restore | BACKUP LOG WITH TRUNCATE\_ONLY | None. The transaction log is automatically truncated when the database is using the Simple recovery model. If you must remove the log backup chain from a database, switch to the Simple recovery model. |
| Backup and restore | BACKUP TRANSACTION | Use BACKUP LOG. |
| Backup and restore | DUMP statement | Use BACKUP. |
| Backup and restore | LOAD statement | Use RESTORE. |
| Backup and restore | Named pipe backup devices | Disk or tape devices. |
| Command prompt utilities | isql utility | Use the sqlcmd utility. |
| Compatibility level | 60, 65, and 70 compatibility levels | Databases must be set to at least compatibility level 80. |
| Configuration options | 'allow updates' option of sp\_configure | Option is present, but direct updates to system tables are not supported. |
| Configuration options | 'open objects' option of sp\_configure | Option is present, but its functionality has been deactivated. In SQL Server 2005, the number of open database objects is managed dynamically and is limited only by available memory. The 'open objects' option has been left in sp\_configure to ensure backward compatibility with existing scripts. |
| Configuration options | 'set working set size' option of sp\_configure | Option is present, but its functionality has been deactivated. |
| Database creation | DISK INIT DISK RESIZE | Legacy behavior from SQL Server 6.x. |
| Database creation | FOR LOAD option of CREATE DATABASE | RESTORE operations can create a database. |
| DBCC | DBCC CONCURRENCYVIOLATION | None. |
| DBCC | DBCC DBREPAIR | Use DROP DATABASE to remove a damaged database. |
| DBCC | DBCC NEWALLOC | DBCC CHECKALLOC |
| DBCC | DBCC PINTABLE, DBCC UNPINTABLE | None. |
| DBCC | DBCC ROWLOCK | Row-level locking is automatic. |
| DBCC | DBCC TEXTALL | Use DBCC CHECKDB. |
| DBCC | DBCC TEXTALLOC | Use DBCC CHECKTABLE. |
| Extended store procedure programming | Use of SRV\_PWD field in the SRV\_PFIELD structure when there has been an impersonation context switch from the original login | None. |
| Groups | sp\_addgroup | Use roles. |
| Groups | sp\_changegroup | Use roles. |
| Groups | sp\_dropgroup | Use roles. |
| Groups | sp\_helpgroup | Use roles. |
| Network protocols | The following protocols: NWLink IPX/SPX, AppleTalk, Banyan Vines, Multiprotocol | Configure your application and the instance of the Database Engine to use one of the supported protocols: TCP/IP sockets, named pipes, VIA, or shared memory. |
| Rebuild master | Rebuildm.exe | Use the REBUILDDATABASE option in Setup.exe. |
| Sample databases | Northwind and pubs | Use AdventureWorks; however, Northwind and pubs are available as downloads, or can be copied from a previous installation of SQL Server. |
| Setup.exe | Remote Setup: The TARGETCOMPUTER parameter | Use a remote connection to run the SQL Server Setup program in user interface mode or from the command prompt. |
| Tools | Surface Area Configuration Tool | The Surface Area Configuration Tool is discontinued for SQL Server 2008. For more information, see [Backward Compatibility](http://go.microsoft.com/fwlink/?LinkID=13217) in SQL Server 2008 Books Online. |
| Transact-SQL | \*= and =\* outer join operators | Use the JOIN syntax of the FROM clause. |
| Virtual tables | syslocks | Use sys.dm\_tran\_locks. |
| Web Assistant | sp\_makewebtask  sp\_dropwebtask  sp\_runwebtask  sp\_enumcodepages | We recommend that you use SQL Server Reporting Services instead. |

For a comprehensive list of discontinued functionality in SQL Server 2005 and SQL Server 2008, see the following topics in SQL Server Books Online:

* [Discontinued Database Engine Functionality in SQL Server 2005](http://go.microsoft.com/fwlink/?LinkId=133134)
* [Discontinued Database Engine Functionality in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133135)

### Breaking Changes

Although Microsoft has worked hard to minimize the impact of upgrading, there are a few breaking changes in SQL Server 2008 that could cause your upgrade to fail.

#### Collations

SQL Server 2008 introduces 80 new collations to bring SQL Server inline with the collations that Windows Server 2008 offers. If you use one of these new collations, clients with older drivers might throw exceptions.

#### .NET Framework

SQL Server 2008 installs .NET Framework 3.5 Service Pack 1 (SP1). If there is a new type in this version of the .NET Framework that has the same name as a user-defined type, SQL Server will mark your database "suspect." In addition, assemblies in the Global Assembly Cache (GAC) will be updated to version 3.5 SP1. If you have assemblies registered in your database that are no longer supported, your application will fail. To check for the existence of any unsupported assemblies in your database, you can run the following query:

SELECT

name

FROM

sys.assemblies

WHERE

clr\_name LIKE '%publickeytoken=b03f5f7f11d50a3a,%'

#### Dynamic Management Views

A few DMVs have been updated in SQL Server 2008. Table 8-3 lists the modifications, which might cause upgrade problems if you are not prepared for them.

Table 8-3: Changes to Dynamic Management Views

| **View** | **Description** |
| --- | --- |
| sys.dm\_os\_sys\_info | Removed the cpu\_ticks\_in\_ms and sqlserver\_start\_time\_cpu\_ticks columns. |
| sys.dm\_exec\_query\_resource\_semaphores sys.dm\_exec\_query\_memory\_grants | The resource\_semaphore\_id column is not a unique ID in SQL Server 2008. This change can affect troubleshooting query execution. For more information, see [sys.dm\_exec\_query\_resource\_semaphores (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=133138) in SQL Server 2008 Books Online. |

#### Transact-SQL

There are a number of Transact-SQL breaking changes in SQL Server 2008. For a complete list, review Table 8-4.

Table 8-4: Breaking Changes to Transact-SQL

| **Feature** | **Description** |
| --- | --- |
| ALTER\_AUTHORIZATION\_DATABASE data definition language (DDL) event | In SQL Server 2005, when the DDL event ALTER\_AUTHORIZATION\_DATABASE fires, the value 'object' is returned in the ObjectType element of the EVENTDATA XML for this event when the entity type of the securable in the DDL operation is an object. In SQL Server 2008, the actual type (for example, 'table', or 'function') is returned. |
| CONVERT | If an invalid style is passed to the CONVERT function, an error is returned when the type of conversion is binary to character or character to binary. In earlier versions of SQL Server, the invalid style is set to the default style for binary-to-character and character-to-binary conversions. |
| GRANT/DENY/REVOKE EXECUTE on assemblies | EXECUTE permission cannot be granted, denied, or revoked to assemblies. This permission has no affect and now causes an error. Grant, deny, or revoke EXECUTE permission on the stored procedures or functions that reference the assembly method instead. |
| GRANT/DENY/REVOKE permissions on system types | Permissions cannot be granted, denied, or revoked to system types. In earlier versions of SQL Server, these statements succeed but have no effect. In SQL Server 2008, an error is returned. |
| OUTPUT clause | To prevent nondeterministic behavior, the OUTPUT clause cannot reference a column from a view or inline table-valued function when that column is defined by one of the following methods:   * A subquery. * A user-defined function that performs user- or system-data access, or is assumed to perform such access. * A computed column that contains in its definition a user-defined function that performs user- or system-data access.   When SQL Server detects such a column in the OUTPUT clause, error 4186 is raised. For more information, see [MSSQLSERVER\_4186](http://go.microsoft.com/fwlink/?LinkId=133139). |
| OUTPUT INTO clause | The target table of the OUTPUT INTO clause cannot have any enabled triggers. |
| READPAST table hint | The READPAST table hint cannot be specified when the READ\_COMMITTED\_SNAPSHOT database option is set to ON and either of the following conditions are true:   * The transaction isolation level of the session is READ COMMITTED. * The READCOMMITTED table hint is also specified in the query,   To specify the READPAST hint in these cases, remove the READCOMMITTED table hint if present and include the READCOMMITTEDLOCK table hint in the query. |
| sp\_helpuser | The following column names that are returned in the result set of the sp\_helpuser stored procedure have changed.   |  |  | | --- | --- | | Previous Column Name | New Column Name | | Group\_name | Role\_name | | Group\_id | Role\_id | | Users\_in\_group | Users\_in\_role | |
| Transparent data encryption (TDE) | TDE is performed at the I/O level: The page structure is unencrypted in memory and is encrypted only when the page is written to disk. Both the database files and log files are encrypted. Third-party applications that bypass the regular SQL Server mechanism for accessing pages (for example, by scanning the data or log files directly) will fail when a database uses TDE because the data is encrypted in the files. Such applications can leverage the Window Cryptographic API to develop a solution for decrypting the data outside of SQL Server. |

For a complete list of breaking changes in SQL Server 2005 and SQL Server 2008, see the following SQL Server Books Online topics:

* [Breaking Changes to Database Engine Features in SQL Server 2005](http://go.microsoft.com/fwlink/?LinkId=133140)
* [Breaking Changes to Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133141)

### Behavior Changes

SQL Server 2008 changes the behavior of a number of features that your stored procedures, queries, and scripts might use. These changes likely will not prevent an upgrade to SQL Server 2008, but they might affect how your query-intensive system works after the upgrade. Be sure to review your code for the changes covered in this section to make sure that it works correctly after your upgrade.

For a complete list of behavior changes in SQL Server 2005 and SQL Server 2008, see the following SQL Server Books Online topics:

* [Behavior Changes to Database Engine Features in SQL Server 2005](http://go.microsoft.com/fwlink/?LinkId=133143)
* [Behavior Changes to Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133142)

#### Query Processor Architecture

Queries on partitioned tables and indexes are handled differently in SQL Server 2008. Specifically, queries that use the USE PLAN hint might contain an invalid plan. For details about this change, see Chapter 3, "Relational Databases," or [Query Processing Enhancements on Partitioned Tables and Indexes](http://go.microsoft.com/fwlink/?LinkId=133145) in SQL Server 2008 Books Online.

#### REPLACE Function

In prior versions of SQL Server, trailing spaces are removed from the input passed to the REPLACE function. SQL Server 2008 keeps the spaces.

#### @@VERSION

The @@VERSION function returns additional detail in SQL Server 2008. You need to modify any scripts that execute this function to account for the new format: major.minor.build.incremental-build.

#### Database Compatibility

If you find that you cannot support certain behavior changes immediately, you have the option of modifying the database compatibility level to limit or allow certain functionality depending on the level you select.

During the upgrade process to SQL Server 2008, a database will retain its existing compatibility level if the database compatibility level is greater than or equal to 80; otherwise, Setup will automatically set the compatibility level to 80.

Here are the compatibility levels and their corresponding SQL Server version:

* 65 = SQL Server 6.5
* 70 = SQL Server 7.0
* 80 = SQL Server 2000
* 90 = SQL Server 2005
* 100 = SQL Server 2008

Note that compatibility levels 60 and 65 are deprecated and will be removed in a future SQL Server release. SQL Server Management Studio and SQL Server Management Objects (SMO) do not support compatibility level 60 and might produce errors if you try to use them against a database with this compatibility level.

You can determine the compatibility level of your databases either by right-clicking the database in SQL Server Management Studio and selecting Properties and then Options, or by executing the following script:

SELECT name, compatibility\_level FROM sys.databases

The compatibility level of a database governs the ability of database administrators and developers to use some of the new features in SQL Server 2008 as well as their ability to retain some legacy behaviors. If you want to use all the features available in the new release of SQL Server, you should resolve any upgrade issues before changing the compatibility level of a database to 100.

To change the compatibility level of a database, right-click the database in SQL Server Management Studio and select Properties and then Options. In the Compatibility Level drop-down list, select the desired compatibility level. You can also change the compatibility level by issuing the following ALTER DATABASE command:

ALTER DATABASE <Database Name> SET COMPATIBILITY\_LEVEL = 100

**Note:** You can change the compatibility level of the model database so that you can create a new database with a non-default compatibility level. The default compatibility level for new SQL Server 2008 installations is 100.

For more information about changing compatibility levels, see [ALTER DATABASE Compatibility Level (Transact-SQL](http://go.microsoft.com/fwlink/?LinkID=133122)) in SQL Server 2008 Books Online.

#### INFORMATION\_SCHEMA Views

SQL Server 2005 and SQL Server 2008 introduce several changes to INFORMATION\_SCHEMA views. Database administrators and developers should review their stored procedures, ad hoc scripts, and administrative scripts to determine whether the changes that Table 8-5 lists will affect their scripts.

Table 8-5: Changes to INFORMATION\_SCHEMA Views

| **View** | **Change** |
| --- | --- |
| All INFORMATION\_SCHEMA columns that return the schema of an object | Now return the schema and not the user. |
| SCHEMATA | Now returns all schemas in a database instead of all databases in an instance. |
| COLUMNS | This view's ORDINAL\_POSITION column has been changed from smallint to int. |
| PARAMETERS | This view's ORDINAL\_POSITION column has been changed from smallint to int. |
| REFERENTIAL\_CONSTRAINTS | This view's MATCH\_OPTION column has been changed from varchar(4) to varchar(7); the default value has changed from 'NONE' to 'SIMPLE'.  Also, this view's UPDATE\_RULE and DELETE\_RULE columns have been changed from varchar(9) to varchar(11). |
| ROUTINE\_COLUMNS | This view's ORDINAL\_POSITION column has been changed from smallint to int. |

#### System Tables

SQL Server 2008 includes many changes to system tables, so you need to review your stored procedures, ad hoc queries, and administrative scripts to determine whether the changes will affect your code. Table 8-6 lists some of the key changes.

Table 8-6: System Table Changes in SQL Server 2008

| **System Table** | **Change** |
| --- | --- |
| Syslockinfo | Returns a value of 0 for the rsc\_objid and rsc\_indid columns instead of the object id and index id, respectively. |
| Sysaltfiles | The name column has been changed from nchar(128) to sysname, and the filename column has been changed from nchar(260) to nvarchar(260). |
| Sysconfigures | The config column has been changed from smallint to int. |
| Syscursorcolumns | The data\_type\_sql column has been changed from smallint to int. |
| sysfiles | The name column has been changed from nchar(128) to sysname, and the filename column has been changed from nchar(260) to nvarchar(260). |
| Sysmessages | The severity column has been changed from smallint to tinyint. |
| Sysperfinfo | The cntr\_value column has been changed from int to bigint. |
| Sysprocesses | The waittime column has been changed from int to bigint, the hostprocess column has been changed from nchar(8) to nchar(10), and the request\_id column has been added. |
| Sysprotects | The columns column has been changed from varbinary(4000) to varbinary(8000). |
| Sysservers | The srvcollation column has been changed from int to sysname, the nonsqlsub column has been added, and the topologyx column now returns a 0. |
| Sysoledbusers | The rmtpassword column now returns a NULL. |
| Sysindexes | The keys column and the statblob column now return a NULL. |
| Syscomments | The compressed column now returns a 0. |
| Sysdevices | The size column now returns a 0. |
| Sysobjects | The schema\_ver column now returns a 0. |
| Sysremotelogins | The status column now returns a 0. |

For a complete list of the changes to system tables, see [Behavior Changes to Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133142) in SQL Server 2008 Books Online.

#### System Stored Procedures

The new release of SQL Server has also introduced several changes to system stored procedures. Be sure to review your stored procedures, ad hoc queries, and administration scripts to determine whether the changes listed in Table 8-7 will affect your code.

Table 8-7: System Stored Procedure Changes in SQL Server 2008

| **System Stored Procedure** | **Change** |
| --- | --- |
| sp\_lock | Returns a value of 0 for the objid and indid columns instead of the object id and index id, respectively. |
| sp\_bindefault | The @objname parameter has been changed from nvarchar(517) to nvarchar(776). |
| sp\_bindrule | The @objname parameter has been changed from nvarchar(517) to nvarchar(776). |
| sp\_changeobjectowner | The @objname parameter has been changed from nvarchar(517) to nvarchar(776). |
| sp\_detach\_db | The @keepfulltextindexfile parameter has been changed from nvarchar(517) to nvarchar(776). |
| sp\_fulltext\_service | The @action parameter has been changed from archar(20) to nvarchar(100), and the @keepfulltextindexfile parameter has been changed from an int to sql\_variant. |
| sp\_getapplock | Has added the @DbPrincipal parameter. |
| sp\_releaseapplock | Has added the @DbPrincipal parameter. |
| sp\_setapprole | Has added the @fCreateCookie and @cookie parameters. |
| sp\_settriggerorder | Has added the @namespace parameter, and the @stmttype parameter has been changed from varchar(10) to varchar(50). |
| sp\_sproc\_columns | Has added the @fUsePattern parameter. |
| sp\_stored\_procedures | Has added the @fUsePattern parameter. |
| sp\_table\_privileges | Has added the @fUsePattern parameter. |
| sp\_table\_privileges\_ex | Has added the @fUsePattern parameter. |
| sp\_tables | Has added the @fUsePattern parameter. |
| sp\_tables\_ex | Has added the @fUsePattern parameter. |
| sp\_addtype | The permission has changed from any user being able to execute this stored procedure to users must be members of the db\_ddladmin or db\_owner database role to execute sp\_addtype. |
| sp\_altermessage | No longer specifies whether or not a system message is written to the Windows Application log. |
| sp\_changedbowner | The permission has changed: Members who execute this system stored procedure under the security context of the db\_ddladmin or db\_securityadmin fixed database roles must also be granted CONTROL permission on the securable. |
| sp\_help | Now returns two result sets for functions instead of the previous one result set. |
| xp\_cmdshell | If an error occurs during the execution of this extended stored procedure, it will not raise an error message and will terminate the execution of the extended stored procedure. |

For a full list of system stored procedure changes, see [Behavior Changes to Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133142) in SQL Server 2008 Books Online.

#### Security

Security created and maintained through Transact-SQL statements has undergone some changes in SQL Server 2008. Database administrators and developers should review their stored procedures, ad hoc queries, and administrative scripts to determine whether any of the security behavior changes listed in Table 8-8 will affect their scripts.

Table 8-8: Changes to Security-Related Transact-SQL Statements

| **Security Feature** | **Change** |
| --- | --- |
| GRANT ALL | Granting ALL permissions has been deprecated in SQL Server 2005. GRANT ALL will grant only permissions that were grantable in SQL Server 2000. The system will display an informational message stating the deprecation. |
| Password Comparisons | SQL Server 2005 has changed the behavior of password comparisons. Now, precise case is checked for password comparisons. |
| DROP LOGIN | SQL Server 2005 lets you drop logins even if database users are mapped to the login. |
| System Metadata | System metadata is no longer viewable through the public role. Users will see metadata only for objects they own or have some permissions on, either directly or through a role. |
| Virtual Table Access | Users need the VIEW SERVER STATE and SELECT permissions to access system virtual tables. |

For a full list of security-related Transact-SQL changes, see [Behavior Changes to Database Engine Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133142) in SQL Server 2008 Books Online.

Chapter 5, "Database Security," covers other security issues involved in upgrading to SQL Server 2008.

#### Functions

SQL Server 2008 has changed the behavior of referencing built-in functions. In the new SQL Server release, each reference to a built-in function produces a different result because it is evaluated one time for each outer query reference. Multiple references to such columns in a subquery will not cause the function to be evaluated multiple times, and you can reuse the value produced by these functions in the subquery. Review your code to make sure it accounts for the changes to function behavior that Table 8-9 lists.

Table 8-9: Changes to Built-in Functions

| **Function** | **Change** |
| --- | --- |
| fn\_servershareddrives | The permission to execute this function has been changed; the function now requires the VIEW SERVER STATE permission on the server. |
| fn\_virtualfilestats | The permission to execute this function has been changed; the function now requires the VIEW SERVER STATE permission on the server. |
| fn\_virtualservernodes | The permission to execute this function has been changed; this function now requires the VIEW SERVER STATE permission on the server. |
| HOST\_ID | Now returns a char(10) instead of a char(8). |
| SERVERPROPERTY | The ProductVersion property has been changed from varchar to nvarchar. |
| UPDATE() | Now detects changes to timestamp columns, and returns TRUE if checked inside a trigger. |

**Note:** In SQL Server 2005 and SQL Server 2008, user-defined functions can include most nondeterministic built-in system functions.

#### Reserved Keywords

One possible issue you might face when upgrading a database and changing the compatibility level of that database involves keywords marked as reserved. SQL Server uses reserved keywords for defining, manipulating, and accessing databases. Reserved keywords are part of the grammar of the Transact-SQL language that SQL Server uses to parse and understand Transact-SQL statements and batches.

Although you can use Transact-SQL reserved keywords as identifiers or names of databases or database objects (such as tables, columns, views, and so on), you can do this only by using either quoted identifiers or delimited identifiers. Using reserved keywords as the names of variables and stored procedure parameters is not restricted. For more information, see [Using Identifiers as Object Names](http://go.microsoft.com/fwlink/?LinkId=133147) in SQL Server 2008 Books Online.

SQL Server 2008 Upgrade Advisor (which we discuss in a moment) will flag stored procedures for usage of new reserved keywords. But make sure to perform a manual review of Transact-SQL code embedded in application code and other external sources.Table 8-10 lists SQL Server 2008 reserved keywords.

Table 8-10: SQL Server 2008 Reserved Keywords

|  |  |  |
| --- | --- | --- |
| ADD | EXISTS | PRECISION |
| ALL | EXIT | PRIMARY |
| ALTER | EXTERNAL | PRINT |
| AND | FETCH | PROC |
| ANY | FILE | PROCEDURE |
| AS | FILLFACTOR | PUBLIC |
| ASC | FOR | RAISERROR |
| AUTHORIZATION | FOREIGN | READ |
| BACKUP | FREETEXT | READTEXT |
| BEGIN | FREETEXTTABLE | RECONFIGURE |
| BETWEEN | FROM | REFERENCES |
| BREAK | FULL | REPLICATION |
| BROWSE | FUNCTION | RESTORE |
| BULK | GOTO | RESTRICT |
| BY | GRANT | RETURN |
| CASCADE | GROUP | REVERT |
| CASE | HAVING | REVOKE |
| CHECK | HOLDLOCK | RIGHT |
| CHECKPOINT | IDENTITY | ROLLBACK |
| CLOSE | IDENTITY\_INSERT | ROWCOUNT |
| CLUSTERED | IDENTITYCOL | ROWGUIDCOL |
| COALESCE | IF | RULE |
| COLLATE | IN | SAVE |
| COLUMN | INDEX | SCHEMA |
| COMMIT | INNER | SECURITYAUDIT |
| COMPUTE | INSERT | SELECT |
| CONSTRAINT | INTERSECT | SESSION\_USER |
| CONTAINS | INTO | SET |
| CONTAINSTABLE | IS | SETUSER |
| CONTINUE | JOIN | SHUTDOWN |
| CONVERT | KEY | SOME |
| CREATE | KILL | STATISTICS |
| CROSS | LEFT | SYSTEM\_USER |
| CURRENT | LIKE | TABLE |
| CURRENT\_DATE | LINENO | TABLESAMPLE |
| CURRENT\_TIME | LOAD | TEXTSIZE |
| CURRENT\_TIMESTAMP | MERGE | THEN |
| CURRENT\_USER | NATIONAL | TO |
| CURSOR | NOCHECK | TOP |
| DATABASE | NONCLUSTERED | TRAN |
| DBCC | NOT | TRANSACTION |
| DEALLOCATE | NULL | TRIGGER |
| DECLARE | NULLIF | TRUNCATE |
| DEFAULT | OF | TSEQUAL |
| DELETE | OFF | UNION |
| DENY | OFFSETS | UNIQUE |
| DESC | ON | UNPIVOT |
| DISK | OPEN | UPDATE |
| DISTINCT | OPENDATASOURCE | UPDATETEXT |
| DISTRIBUTED | OPENQUERY | USE |
| DOUBLE | OPENROWSET | USER |
| DROP | OPENXML | VALUES |
| DUMP | OPTION | VARYING |
| ELSE | OR | VIEW |
| END | ORDER | WAITFOR |
| ERRLVL | OUTER | WHEN |
| ESCAPE | OVER | WHERE |
| EXCEPT | PERCENT | WHILE |
| EXEC | PIVOT | WITH |
| EXECUTE | PLAN | WRITETEXT |

In addition, the ISO standard defines a list of reserved keywords. Avoid using ISO reserved keywords for object names and identifiers. The ODBC reserved keyword list, shown in the next table, is the same as the ISO reserved keyword list.

**Note:** The ISO standard reserved keywords list sometimes can be more restrictive than SQL Server and at other times less restrictive. For example, the ISO reserved keywords list contains INT. SQL Server does not have to distinguish this as a reserved keyword.

#### ODBC Reserved Keywords

ODBC features keywords reserved for use in ODBC function calls. These words do not constrain the minimum SQL grammar; however, to ensure compatibility with drivers that support the core SQL grammar, applications should avoid using these keywords. Table 8-11 lists the SQL Server 2008 ODBC reserved keywords.

Table 8-11: ODBC Reserved Keywords

|  |  |  |
| --- | --- | --- |
| ABSOLUTE | EXEC | OVERLAPS |
| ACTION | EXECUTE | PAD |
| ADA | EXISTS | PARTIAL |
| ADD | EXTERNAL | PASCAL |
| ALL | EXTRACT | POSITION |
| ALLOCATE | FALSE | PRECISION |
| ALTER | FETCH | PREPARE |
| AND | FIRST | PRESERVE |
| ANY | FLOAT | PRIMARY |
| ARE | FOR | PRIOR |
| AS | FOREIGN | PRIVILEGES |
| ASC | FORTRAN | PROCEDURE |
| ASSERTION | FOUND | PUBLIC |
| AT | FROM | READ |
| AUTHORIZATION | FULL | REAL |
| AVG | GET | REFERENCES |
| BEGIN | GLOBAL | RELATIVE |
| BETWEEN | GO | RESTRICT |
| BIT | GOTO | REVOKE |
| BIT\_LENGTH | GRANT | RIGHT |
| BOTH | GROUP | ROLLBACK |
| BY | HAVING | ROWS |
| CASCADE | HOUR | SCHEMA |
| CASCADED | IDENTITY | SCROLL |
| CASE | IMMEDIATE | SECOND |
| CAST | IN | SECTION |
| CATALOG | INCLUDE | SELECT |
| CHAR | INDEX | SESSION |
| CHAR\_LENGTH | INDICATOR | SESSION\_USER |
| CHARACTER | INITIALLY | SET |
| CHARACTER\_LENGTH | INNER | SIZE |
| CHECK | INPUT | SMALLINT |
| CLOSE | INSENSITIVE | SOME |
| COALESCE | INSERT | SPACE |
| COLLATE | INT | SQL |
| COLLATION | INTEGER | SQLCA |
| COLUMN | INTERSECT | SQLCODE |
| COMMIT | INTERVAL | SQLERROR |
| CONNECT | INTO | SQLSTATE |
| CONNECTION | IS | SQLWARNING |
| CONSTRAINT | ISOLATION | SUBSTRING |
| CONSTRAINTS | JOIN | SUM |
| CONTINUE | KEY | SYSTEM\_USER |
| CONVERT | LANGUAGE | TABLE |
| CORRESPONDING | LAST | TEMPORARY |
| COUNT | LEADING | THEN |
| CREATE | LEFT | TIME |
| CROSS | LEVEL | TIMESTAMP |
| CURRENT | LIKE | TIMEZONE\_HOUR |
| CURRENT\_DATE | LOCAL | TIMEZONE\_MINUTE |
| CURRENT\_TIME | LOWER | TO |
| CURRENT\_TIMESTAMP | MATCH | TRAILING |
| CURRENT\_USER | MAX | TRANSACTION |
| CURSOR | MIN | TRANSLATE |
| DATE | MINUTE | TRANSLATION |
| DAY | MODULE | TRIM |
| DEALLOCATE | MONTH | TRUE |
| DEC | NAMES | UNION |
| DECIMAL | NATIONAL | UNIQUE |
| DECLARE | NATURAL | UNKNOWN |
| DEFAULT | NCHAR | UPDATE |
| DEFERRABLE | NEXT | UPPER |
| DEFERRED | NO | USAGE |
| DELETE | NONE | USER |
| DESC | NOT | USING |
| DESCRIBE | NULL | VALUE |
| DESCRIPTOR | NULLIF | VALUES |
| DIAGNOSTICS | NUMERIC | VARCHAR |
| DISCONNECT | OCTET\_LENGTH | VARYING |
| DISTINCT | OF | VIEW |
| DOMAIN | ON | WHEN |
| DOUBLE | ONLY | WHENEVER |
| DROP | OPEN | WHERE |
| ELSE | OPTION | WITH |
| END | OR | WORK |
| END-EXEC | ORDER | WRITE |
| ESCAPE | OUTER | YEAR |
| EXCEPT | OUTPUT | ZONE |
| EXCEPTION |  |  |

#### Future Keywords

Microsoft has announced keywords that could be reserved in future releases of SQL Server as new features are implemented. Consider avoiding the use of these words as identifiers. Database administrators and developers who determine that their stored procedures, queries, or scripts use these new keywords need to either modify the keyword or enclose the keyword in quotation marks or brackets. Table 8-12 lists keywords that could be reserved in future SQL Server releases.

Table 8-12: Future Keywords

|  |  |  |
| --- | --- | --- |
| ABSOLUTE | FREE | PRESERVE |
| ACTION | FULLTEXTTABLE | PRIOR |
| ADMIN | GENERAL | PRIVILEGES |
| AFTER | GET | READS |
| AGGREGATE | GLOBAL | REAL |
| ALIAS | GO | RECURSIVE |
| ALLOCATE | GROUPING | REF |
| ARE | HOST | REFERENCING |
| ARRAY | HOUR | RELATIVE |
| ASSERTION | IGNORE | RESULT |
| AT | IMMEDIATE | RETURNS |
| BEFORE | INDICATOR | ROLE |
| BINARY | INITIALIZE | ROLLUP |
| BIT | INITIALLY | ROUTINE |
| BLOB | INOUT | ROW |
| BOOLEAN | INPUT | ROWS |
| BOTH | INT | SAVEPOINT |
| BREADTH | INTEGER | SCROLL |
| CALL | INTERVAL | SCOPE |
| CASCADED | ISOLATION | SEARCH |
| CAST | ITERATE | SECOND |
| CATALOG | LANGUAGE | SECTION |
| CHAR | LARGE | SEQUENCE |
| CHARACTER | LAST | SESSION |
| CLASS | LATERAL | SETS |
| CLOB | LEADING | SIZE |
| COLLATION | LESS | SMALLINT |
| COMPLETION | LEVEL | SPACE |
| CONNECT | LIMIT | SPECIFIC |
| CONNECTION | LOCAL | SPECIFICTYPE |
| CONSTRAINTS | LOCALTIME | SQL |
| CONSTRUCTOR | LOCALTIMESTAMP | SQLEXCEPTION |
| CORRESPONDING | LOCATOR | SQLSTATE |
| CUBE | MAP | SQLWARNING |
| CURRENT\_PATH | MATCH | START |
| CURRENT\_ROLE | MINUTE | STATE |
| CYCLE | MODIFIES | STATEMENT |
| DATA | MODIFY | STATIC |
| DATE | MODULE | STRUCTURE |
| DAY | MONTH | TEMPORARY |
| DEC | NAMES | TERMINATE |
| DECIMAL | NATURAL | THAN |
| DEFERRABLE | NCHAR | TIME |
| DEFERRED | NCLOB | TIMESTAMP |
| DEPTH | NEW | TIMEZONE\_HOUR |
| DEREF | NEXT | TIMEZONE\_MINUTE |
| DESCRIBE | NO | TRAILING |
| DESCRIPTOR | NONE | TRANSLATION |
| DESTROY | NUMERIC | TREAT |
| DESTRUCTOR | OBJECT | TRUE |
| DETERMINISTIC | OLD | UNDER |
| DICTIONARY | ONLY | UNKNOWN |
| DIAGNOSTICS | OPERATION | UNNEST |
| DISCONNECT | ORDINALITY | USAGE |
| DOMAIN | OUT | USING |
| DYNAMIC | OUTPUT | VALUE |
| EACH | PAD | VARCHAR |
| END-EXEC | PARAMETER | VARIABLE |
| EQUALS | PARAMETERS | WHENEVER |
| EVERY | PARTIAL | WITHOUT |
| EXCEPTION | PATH | WORK |
| FALSE | POSTFIX | WRITE |
| FIRST | PREFIX | YEAR |
| FLOAT | PREORDER | ZONE |
| FOUND | PREPARE |  |

For the most recent reserved keyword list, see [Reserved Keywords (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=133149) in SQL Server 2008 Books Online.

### Upgrade Tools

There are three main tools available to help identify potential problems before you upgrade to SQL Server 2008: the SQL Server 2008 Upgrade Advisor, the SQL Server 2008 Upgrade Assistant, and the Best Practices Analyzer (BPA), which comes in versions for SQL Server 2000 and SQL Server 2005. For details about using these upgrade tools, see Chapter 1, "Upgrade Planning and Deployment."

#### SQL Server 2008 Upgrade Advisor

Perhaps the most useful upgrade tool is Upgrade Advisor, available for download as part of the [Microsoft SQL Server 2008 Feature Pack August 2008](http://go.microsoft.com/fwlink/?LinkId=133088). This tool quickly identifies blocking issues and many other known potential problems so that you can address them before or during the upgrade process. You should always run this tool on your SQL Server 2000 or SQL Server 2005 instances, including Transact-SQL database code and external scripts, before upgrading.

#### SQL Server 2008 Upgrade Assistant

Upgrade Assistant is an external tool that lets you determine in a test environment how an application currently running on SQL Server 2000 or SQL Server 2005 will run on SQL Server 2008. This tool uses Upgrade Advisor, along with baseline and trace replay, in a test environment to help identify compatibility issues. For more information about Upgrade Assistant and download instructions, see [SQL Server 2008 Upgrade Assistant](http://go.microsoft.com/fwlink/?LinkId=133150) on the Scalability Experts Web site.

#### Best Practices Analyzer

In addition, you can use BPA to help identify bad practices in your Transact-SQL database code and Transact-SQL scripts. You should always run BPA on your Transact-SQL code before an upgrade because you might identify practices that you need to change. The upgrade process could be the opportunity you need to implement these changes. There is a version of BPA for SQL Server 2000 and another for SQL Server 2005.

You can download SQL Server 2000 BPA at [the SQL Server 2000 BPA download page](http://go.microsoft.com/fwlink/?LinkId=133151) and SQL Server 2005 BPA at [SQL Server 2008 Best Practices Analyzer download page](http://go.microsoft.com/fwlink/?LinkId=133152).

### 64-Bit Considerations

Transact-SQL queries are completely compatible between 32-bit and 64-bit editions of SQL Server 2008.

### Known Issues and Workarounds

This section helps you focus on some key Transact-SQL-related known issues regarding upgrading from SQL Server 2000 or SQL Server 2005 to SQL Server 2008 and their solutions. As part of your upgrade preparation, review these issues and make sure you resolve any of them in your implementation before upgrading.

#### AUTO\_UPDATE\_STATISTICS

To make sure that database statistics are updated as part of your upgrade—and that your queries get optimal query plans—set the AUTO\_UPDATE\_STATISTICS configuration option to ON before you upgrade any databases to SQL Server 2008. When you set AUTO\_UPDATE\_STATISTICS to ON, SQL Server updates all statistics when they are first referenced, ensuring that the system is using the most up-to-date information to try to create the best plans for your queries.

You can use the following statement to enable AUTO\_UPDATE\_STATISTICS:

ALTER DATABASE <Database Name> SET AUTO\_UPDATE\_STATISTICS ON

You can then use either the [sp\_updatestats stored procedure](http://go.microsoft.com/fwlink/?LinkId=133154) or the [UPDATE STATISTICS command](http://go.microsoft.com/fwlink/?LinkId=133155) to update statistics immediately to ensure optimal performance.

#### System Object Name Collation Matching

Earlier releases of SQL Server match system object names against the collation of the master database. SQL Server 2008 matches system object names against the collation of the current database. You need to review your stored procedures, ad hoc queries, and scripts to determine whether matching against the collation of the current database will cause a system object name search to fail.

#### Query Hints

With a few exceptions, SQL Server 2008 requires the WITH keyword when you use a table hint in the FROM clause of a query. You need to review your stored procedures, ad hoc queries, and administrative scripts for table hint usage and modify the table hint syntax to include the WITH keyword. For information about which hints do not require the WITH keyword, see [FROM (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkID=133123) in SQL Server 2008 Books Online. Other FROM clause changes include the following:

* The FROM clause supports the SQL-92-SQL syntax for joined tables and derived tables. SQL-92 syntax provides the INNER, LEFT OUTER, RIGHT OUTER, FULL OUTER, and CROSS join operators.
* The outer join operators (\*= and =\*) are not supported when the compatibility level of the database is set to 90.

#### Extended Stored Procedures

If your database environment uses extended stored procedures, you might need to reregister them as extended stored procedures. Extended stored procedures that were registered without the full path for the DLL might not work after an upgrade because of changes in the directory structure of SQL Server 2008. These directory structure changes might leave SQL Server unable to locate the DLL for the stored procedure.

You can use the sp\_dropextendedproc and the sp\_addextendedproc system stored procedures to reregister your extended stored procedures.

If your database environment uses custom extended stored procedures, you need to review those extended stored procedures for use of the SRV\_PWD field in the SRV\_PFIELD structure when there has been an impersonation context switch from the original login; this functionality has been discontinued.

**Note:** Extended stored procedures will be removed in a future version of SQL Server. Avoid using this feature in new development work, and plan to modify applications that currently use this feature to use Common Language Runtime (CLR) integration instead.

#### Trace Flags

Many database administrators use trace flags to manage the behavior of their SQL Server instance. Trace flag behavior can change between SQL Server releases, so you need to review any use of trace flags in stored procedures, queries, scripts, and SQL Server startup parameters. Microsoft recommends that you disable these trace flags during the upgrade process and re-enable them only after testing the functionality of the trace flag under the 100 compatibility level.

In SQL Server 2008, one change in trace flag behavior concerns trace flags that you issue in query sessions. In previous releases of SQL Server, trace flags set in one query session did not affect the behavior of already established sessions. However, trace flags set in SQL Server 2008 query sessions will immediately affect other concurrent sessions.

Additionally, DBCC TRACEON syntax has changed in the new release: If you do not specify the second argument of DBCC TRACEON, the trace flag is valid only for the current (local) connection. In SQL Server 2000, the default behavior of this second parameter is to set the trace flag for all connections (global).

#### Triggers

In SQL Server 2008, the ability to use DDL statements on inserted and deleted tables inside of data-manipulation language (DML) triggers has been discontinued. Before you begin your upgrade, be sure to review your DML trigger-creation scripts and modify them to remove any DDL statements.

#### Indexed Views

You need to be aware of a couple of known issues related to indexed views in SQL Server 2008. The first concerns function determinism, and the second involves the IGNORE\_DUP\_KEY option.

**Function determinism.** Database administrators wanting to set the compatibility level of an upgraded database to 100 should review the index-creation scripts for their indexed views because the following function expressions are now considered nondeterministic in SQL Server 2005 and SQL Server 2008. These functions might interfere with indexed view creation as follows:

* Implicit conversion of non-Unicode character data between collations.
* References to string literals that are implicitly converted to datetime and smalldatetime.

You should modify your index-creation scripts by explicitly converting the literal to the preferred date data type, using a deterministic date format style.

**IGNORE\_DUP\_KEY.** The IGNORE\_DUP\_KEY option must be set to OFF in SQL Server 2008 when you create a unique clustered index on a view; this is the default setting. Setting this option to ON could lead to index view corruption, which would require you to rebuild the index on the view with this option removed or set to OFF. Make sure you review index-creation scripts for indexed views to determine whether the IGNORE\_DUP\_KEY has been turned ON during the index creation.

#### Backup and Restore Scripts

Database administrators should review database backup and restore scripts and modify named pipe usage for backup devices.

#### Network Software

Stand-alone named and default instances support the following network protocols:

* Shared memory
* Named pipes
* TCP/IP
* VIA

**Note:** Shared memory is not supported on failover clusters. For more information about clusters, see Chapter 4, "High Availability."

SQL Server does not support the Banyan VINES Sequenced Packet protocol (SPP), Multiprotocol, AppleTalk, or NWLink IPX/SPX network protocols. Clients previously connecting through these protocols must select a different protocol to connect to SQL Server.

Network software requirements for the 64-bit versions of SQL Server are the same as the requirements for the 32-bit versions.

#### Configuration Options

SQL Server 2008 has changed several options of the sp\_configure system stored procedure. The following options remain present, but their functionality is deactivated:

* allow updates
* open objects
* set working set size

In addition, direct system table manipulation is no longer allowed, even if the *allow updates* option is enabled. You should check all stored procedures, scripts, and embedded code to see whether any of these options exist in your code.

To check within a given database for stored procedures or other objects that update system tables, you can use the following query:

SELECT DISTINCT

OBJECT\_NAME(id) AS UserObject,

OBJECT\_NAME(depid) AS SystemObject

FROM

dbo.sysdepends

WHERE

OBJECTPROPERTY(depid, 'IsSystemTable') = 1

AND

resultobj = 1

You still must manually review ad hoc queries, scripts, and embedded code to look for system table updates because they exist outside the scope of the above query.

#### Database Creation Syntax

SQL Server 2008 includes several changes to the syntax for creating databases that could affect your current database-creation scripts. Remember to review your administrative scripts for the discontinued options that Table 8-13 lists and modify the code before executing the scripts against a SQL Server 2008 installation.

Table 8-13: Discontinued Database-Creation Syntax

|  |  |
| --- | --- |
| **Discontinued Syntax** | **Modification** |
| DISK INIT | SQL Server 6.x legacy behavior with no replacement. |
| DISK RESIZE | SQL Server 6.x legacy behavior with no replacement. |
| FOR LOAD option of CREATE DATABASE statement | Modify code to create the database during the RESTORE operation. |

#### Login Creation

Before beginning your upgrade process, be sure to review stored procedures, ad hoc queries, and administrative scripts containing login creation to determine whether logins are being created with the following names; these names are now reserved in SQL Server 2008:

* sysadmin
* serveradmin
* setupadmin
* securityadmin
* processadmin
* dbcreator
* diskadmin
* bulkadmin

For the most recent reserved keyword list, review [Reserved Keywords (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkID=133149) in SQL Server 2008 Books Online.

#### Join Syntax

Under the 90 and 100 database-compatibility levels, SQL Server 2008 has discontinued the older join syntax **\*=** and **=\***. If you are running under this compatibility level, you need to modify any stored procedures, ad hoc queries, and administrative scripts that use the old syntax to join tables so that they use the JOIN syntax of the FROM clause.

#### System Object Removal

Beginning with SQL Server 2005, all system objects are in a read-only Resource database, which means you can no longer drop system objects. Before upgrading from SQL Server 2000, make sure you review stored procedures, ad hoc queries, and administrative scripts to determine whether those scripts try to drop system objects. If they do try to drop system objects, you need to modify the code by removing the DROP statement call.

#### sp\_helptrigger System Stored Procedure

The sp\_helptrigger system stored procedure adds an additional column to the end of its return set in SQL Server 2005 and SQL Server 2008. This additional column (trigger\_schema) needs to be accounted for in upgraded stored procedures, ad hoc queries, and administrative scripts. Be sure to review your code for calls to this system stored procedure and modify the handling of the result set to account for the additional column.

#### Query Governor Cost Limit

SQL Server 2008 changes how the database system does query cost modeling. Thus, if you apply SET GOVERNOR\_QUERY\_COST\_LIMIT or the *query governor cost limit* option of sp\_configure, queries that ran fine in an earlier version of SQL Server might not run in SQL Server 2008. Make sure you set the connection or server instance's query governor cost-limit settings to an appropriate value or to 0, which specifies no limit on how long a query can run. To read more about the query governor cost limit, see [Query Governor Cost Limit Option](http://go.microsoft.com/fwlink/?LinkId=133156) in SQL Server 2008 Books Online.

## Upgrading from SQL Server 2000 or SQL Server 2005

During the upgrade process, Transact-SQL code objects are essentially passive. Whether you choose an in-place upgrade or a side-by-side upgrade, the end result will be the same as far as your Transact-SQL code is concerned.

### In-Place Upgrade

The Transact-SQL code in database objects will remain unchanged by a direct, in-place upgrade. Any changes you must make to the scripts should be applied after the upgrade (see "Post-Upgrade Tasks" below). In addition, external scripts on disk will be unaffected.

### Side-by-Side Upgrade

In a side-by-side upgrade, any Transact-SQL objects stored in the database will be automatically moved to the new server, but their Transact-SQL code will remain unchanged. You will need to ALTER or recreate the objects directly to update them as required.

In addition, you might need to move any Transact-SQL external scripts to a new server or correct references within your database to those scripts.

## Post-Upgrade Tasks

After upgrading to SQL Server 2008, work through the following checklist to ensure optimum performance:

1. Execute DBCC CHECKDB WITH DATA\_PURITY to check the database for column values that are not valid or are out of range. After you have successfully run DBCC CHECKDB WITH DATA\_PURITY against an upgraded database, you do not need to specify the DATA\_PURITY option again because SQL Server will automatically maintain "data purity." This is the only DBCC CHECKDB check that you need to run as a post-upgrade task.
2. Run DBCC UPDATEUSAGE to correct any incorrect page or row counts.
3. Update statistics by using the sp\_updatestats stored procedure to ensure all statistics are up-to-date.
4. Update revised database Transact-SQL objects such as stored procedures and functions.
5. Update external Transact-SQL scripts.
6. Run a set of test scripts against the new database, and validate the results.

After the upgrade, you will want to analyze how your new SQL Server 2008 instance performs compared with your original SQL Server 2000 or SQL Server 2005 instance. See [RML Utilities for SQL Server](http://go.microsoft.com/fwlink/?LinkId=133157) for a suite of tools for load testing, workload replay, and performance analysis.

For additional post-upgrade details, see Chapter 3, "Relational Databases."

## Conclusion

Although SQL Server 2008 introduces a number of great new features, Microsoft has worked hard to minimize the impact on upgrading existing code. With a good understanding of how the changes to Transact-SQL features might affect your stored procedures, ad hoc queries, and administrative scripts, you can make needed fixes before the upgrade. In addition, the Upgrade Advisor, Upgrade Assistant, and BPA can help ease your transition from SQL Server 2000 or SQL Server 2005 to SQL Server 2008.

## Additional References

For an up-to-date collection of additional references for upgrading Transact-SQL queries to SQL Server 2008, see the [[[Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkID=132510)](http://go.microsoft.com/fwlink/?LinkID=132510)](http://msdnstage.redmond.corp.microsoft.com/en-us/library/cc936623.aspx).

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkId=132512)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132514)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkId=132515)

# Notification Services

## Introduction

SQL Server 2008 does not include Notification Services; SQL Server 2005 is the last version of Notification Services. However, you can get the latest SQL Server 2005 Notification Services Components Package, which has been updated to allow Notification Services to work with a SQL Server 2005 or SQL Server 2008 database instance. The "Feature Changes" section that follows describes the Components Package in more detail.

SQL Server Notification Services provides a robust platform for developing and hosting notification applications. A Notification Services application lets subscribers select information that is important to them and then automatically receive notifications when an event occurs that matches their criteria.

Notification Services is based on SQL Server, XML, and the .NET Framework and offers a rapid development platform for creating notification applications. It has built-in capabilities to handle real-time and scheduled notifications; multiple languages and time zones; extensible event, content formatting, and delivery channel options; and flexible deployment scenarios.

This chapter describes the steps you need to take to use Notification Services in SQL Server 2008.

## Feature Changes

The Components Package includes the necessary server and client components to create and host Notification Services applications. It does not provide integration with SQL Server Management Studio (SSMS), so there is no SQL Server 2008 client tools support for Notification Services. Instead, you must use the command-line utility NSControl.exe to create, register, enable, and update Notification Services instances in SQL Server 2008. For more information, see [nscontrol Utility](http://go.microsoft.com/fwlink/?LinkId=133004) in SQL Server 2005 Books Online.

You can download the SQL Server 2005 Notification Services Components Package from the [Microsoft Web Download Center](http://go.microsoft.com/fwlink/?LinkID=51456). At the time of this writing, the redistribution package is at Release Candidate 1 and available for download at the [Microsoft downloads site.Web site](http://go.microsoft.com/fwlink/?LinkId=133093).

## Preparing to Upgrade

Before you upgrade an instance of Notification Services to SQL Server 2008, you should take precautionary steps that will help you recover from an unsuccessful upgrade attempt, if it is necessary. These steps include stopping all Notification Services instances and backing up all relevant data and configuration files.

### Stop the Notification Services Instances

Before you start the upgrade, you should stop and disable each instance of Notification Services that is on the target server. The following commands, which you issue through the Notification Services command prompt, demonstrate this process by using an instance named ABCPress*.* Be sure to run these commands on each server in a scale-out deployment scenario:

NET STOP NS$ABCPress

NSControl disable –name ABCPress

For more information, see [nscontrol disable Command](http://go.microsoft.com/fwlink/?LinkId=133095) in SQL Server 2005 Books Online.

When you use the side-by-side upgrade scenario to upgrade, you should also unregister the instance of Notification Services on the existing server that is running SQL Server 2005. To do this, use the NSControl unregister command as follows:

NSControl unregister –name ABCPress

For more information about the NSControl unregister command, see [nscontrol register Command](http://go.microsoft.com/fwlink/?LinkId=133098) in SQL Server 2005 Books Online.

**Note:** Do not run NSControl delete*.* This command will delete the instance and application databases for the instance of Notification Services.

### Back Up the Instance Data

It is good practice to back up all user and system databases before anupgrade. By default, instances of Notification Services instances and Notification Services applications create databases with the following naming conventions: *<InstanceName>*NSMain and *<InstanceName><ApplicationName>*, respectively. Keep in mind that SQL Server 2005 lets you create Notification Services database objects in a user-defined database. As a precaution, you should also back up those databases before upgrading.

Additionally, you should back up the source code and Instance Configuration File (ICF) and Application Definition File (ADF) files for all instances, applications, and custom components. These backups give you a means for full recovery if the upgrade fails.

### Install the Prerequisites

The SQL Server 2005 Notification Services Components Package installation requires the target computer to meet the same minimum hardware and software requirements as the edition of the SQL Server instance that will host the Notification Services instances. For complete installation requirements for SQL Server 2005, see [Hardware and Software Requirements for Installing SQL Server 2005](http://go.microsoft.com/fwlink/?LinkID=125254) in SQL Server 2005 Books Online. For information about SQL Server 2008 installation requirements, see [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132172) in SQL Server 2008 Books Online.

New installations of the SQL Server 2005 Notification Services Components Package require that the following components be installed on the target computer. You can install these components from the latest release of the Feature Pack for SQL Server 2005, available for download from the [Microsoft Download Center](http://go.microsoft.com/fwlink/?LinkId=133102).

* Microsoft SQL Server 2005 Management Objects (SMO) Collection
* Microsoft SQL Server Native Client
* SQL Server XML

**Note:** SQL Server XML is required only for installations that use the built-in FileSystemWatcher Event Provider to submit XML events to the application.

### Install the Notification Services Components

Before you upgrade to SQL Server 2008, download and install the SQL Server 2005 Notification Services Components Package.

#### Verify the Notification Services Edition

When you install the SQL Server 2005 Notification Services Components Package RC1 on a server that already has an edition SQL Server installed on it (for example, Express), an incorrect edition might be recorded in the server’s registry. This inconsistency will prevent the instance of Notification Services from starting.

After you install the Components Package, ensure that the correct edition appears in the server’s registry. Use the regedit utility to locate the correct registry key:

* For 32-bit versions of Windows:   
  HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\90\NS\Setup
* For 64-bit versions of Windows:  
  HKEY\_LOCAL\_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Microsoft SQL Server\90\NS\Setup

Add a new string value named RedistEdition. Set the new key’s value to the edition that corresponds to your license by using one of the following values:

* Standard Edition
* Developer Edition
* Enterprise Evaluation Edition
* Enterprise Edition

For more information about editing the registry, see the SQL Server 2005 Notification Services Components Package RC1 Readme.

## Upgrading from SQL Server 2000

SQL Server 2005 Notification Services introduced substantial enhancements compared to SQL Server 2000 Notification Services, including better performance, subscriber-defined conditions, a management API, and new views to manage subscribers and subscriptions.

These enhancements require certain modifications to an instance of SQL Server 2000 Notification Services. These modifications are beyond the scope of this document. However, the [SQL Server 2005 Upgrade Technical Reference Guide](http://go.microsoft.com/fwlink/?LinkId=133117) describes the required changes to the ICF, the ADF, and any custom components you created to extend the built-in capabilities of Notification Services.

To upgrade an instance of SQL Server 2000 Notification Services to SQL Server 2008, the instance and application must first adhere to the changes that are required by a SQL Server 2005 upgrade. After those are completed, the upgrade process is similar to an upgrade from SQL Server 2005 to SQL Server 2008.

## Upgrading from SQL Server 2005

Although Microsoft has upgraded the Notification Services 2005 components to work with SQL Server 2008, it has made no additional changes or enhancements to Notification Services. Thus, upgrading a SQL Server 2005 Notification Services instance to SQL Server 2008 is a straightforward process.

### In-Place Upgrade

You can upgrade a SQL Server 2005 Notification Services instance to work with the SQL Server 2008 Database Engine by using an in-place upgrade method. When you upgrade SQL Server 2005 to SQL Server 2008, the Notification Services components, if installed, are left on the server. The Database Engine and other components are upgraded to SQL Server 2008. When you no longer need the Notification Services components, you can remove them by using Add/Remove Programs in Control Panel. For more information about the steps required to ensure a working Notification Services environment, see the "Preparing to Upgrade" section later in this chapter.

After running the SQL Server 2008 upgrade, if you need to continue using SQL Server 2005 Notification Services, install SQL Server 2005 SP3. The changes available in the Components Package are also included in SQL Server 2005 SP3. Applying it will allow existing Notification Services applications to use the SQL Server 2008 Database Engine.

#### Upgrading the Database Engine

Your first step is to upgrade the SQL Server 2005 Database Engine to SQL Server 2008. For detailed descriptions of this process, see Chapter 1, "Upgrade Planning and Deployment," and Chapter 3, "Relational Databases," in this guide.

After you upgrade SQL Server 2005 to SQL Server 2008, you cannot manage your Notification Services instances by using the SSMS graphical interface. Instead, you must use the NSControl command-line utility to manage Notification Services for the upgraded Database Engine.

After upgrade the Database Engine to SQL Server 2008, you must install SQL Server 2005 SP3 or the Components Package on the existing instance of SQL Server 2005 Notification Services.

#### Enabling and Starting the Instance

After you have upgraded the Database Engine to SQL Server 2008 and applied SQL Server 2005 SP3 or the Components Package, you must enable the instance by using the NSControl enable command, as the following command shows:

nscontrol enable -name ABCPress

For more information, see [nscontrol enable Command](http://go.microsoft.com/fwlink/?LinkId=133233) in SQL Server 2005 Books Online.

When you are ready, you can use the following command to start the Notification Services Windows service:

net start N$ABCPress

You can also start the service in Control Panel under Services.

### Side-by-Side Upgrade

Alternatively, you can use a side-by-side method to upgrade SQL Server 2005 Notification Services to use the SQL Server 2008 Database Engine. To use this technique, you install a separate instance of SQL Server 2008. Detach the instance of Notification Services and application databases from the existing instance of SQL Server 2005, and then attach them to the newly installed instance of SQL Server 2008. For more information, see [How to: Upgrade a Database Using Detach and Attach (Transact-SQL)](http://go.microsoft.com/fwlink/?LinkId=133248) in SQL Server 2008 Books Online.

Create and configure the required user accounts and permissions in the SQL Server 2008 Database Engine to allow the Notification Services service to connect to and use the instance of SQL Server instance.

**Note:** If the SQL Server instance name is hard-coded in your configuration files, you might need to update the ICF and ADF files to reflect the new name.

#### Updating the Metadata

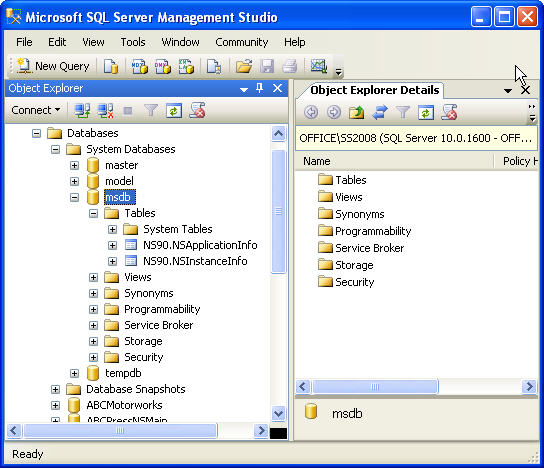
After the instance and application databases are attached to the instance of SQL Server 2008, use the NSControl repair command to create the appropriate metadata entries in the msdb database for the instance of Notification Services.

The following command, which you issue from the SQL Server 2005 Notification Services command prompt, creates the appropriate metadata entries in the msdb database for the ABCPress instance:

NSControl repair -name ABCPress -database ABCPressNSMain \_

-schema dbo -server ServerName

After you run the NSControl repair command, you will see the two new tables that the command created in the NS90 schema of the msdb database, as Figure 9-1 shows.



**Figure 9-1:** Two new tables created by using the NSControl repair command

For more information about NSControl repair, see [nscontrol repair Command](http://go.microsoft.com/fwlink/?LinkId=133269) in SQL Server 2005 Books Online.

#### Registering, Enabling, and Starting the Instance

After you update the metadata in the msdb database, you must register the instance of Notification Services by using the NSControl register command. This creates the appropriate registry entries and adds the Notification Services performance counters to the server. Optionally, registering the instance will create the Windows service for Notification Services.

The following example demonstrates the NSControl register command syntax for an instance of Notification Services named ABCPress:

nscontrol register -name ABCPress -server ss2008 -service \_  
-serviceusername ssns -servicepassword mypassword -sqlusername ssns \_ -sqlpassword mypassword

For more information about the NSControl register command, see [nscontrol register Command](http://go.microsoft.com/fwlink/?LinkId=134575) in SQL Server 2005 Books Online.

After the instance is registered, you can enable the instance by using the NSControl enable command, as the following command shows:

nscontrol enable -name ABCPress

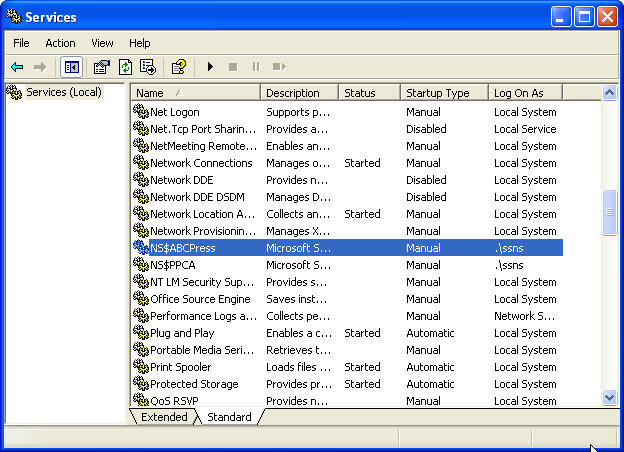
For more information about the NSControl enable command, see [nscontrol enable Command](http://go.microsoft.com/fwlink/?LinkID=133233) in SQL Server 2005 Books Online.

When you are ready, you can use the following command to start the Windows service:

net start N$ABCPress

You can also start the service by using the Services Control Panel applet, as shown in Figure 9-2:

**Figure 9-2:** The Notification Services service in the Services Control Panel applet



## Post-Upgrade Tasks

After the upgrade has completed, you can verify the status of the Notification Services instance by using the NSControl status command, as the following example shows:

nscontrol status -name ABCPress

For details about the NSControl status command, see [nscontrol status Command](http://go.microsoft.com/fwlink/?LinkId=133298) in SQL Server 2005 Books Online.

## Conclusion

Although Notification Services has been deprecated and is not included as part of SQL Server 2008, you can use the SQL Server 2008 Database Engine in conjunction with SQL Server 2005 Notification Services components to develop and host Notification Services applications.

## Additional References

For an up-to-date collection of additional references for running Notification Services with SQL Server 2008, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkID=132510).

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkID=116427)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkID=133003)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkID=132515)

# SQL Server Express

## Introduction

SQL Server 2008 Express is a free version of SQL Server 2008. Because it comes with SQL Server Management Studio Basic, SQL Server Express can be used in the following ways:

* As a local data store
* Embedded with an application
* As a lightweight database server

SQL Server Express is also tightly integrated with Visual Studio 2008 SP1. This facilitates the design and development of database applications.

SQL Server 2008 Express is the ideal upgrade from either the Microsoft SQL Server 2000 Desktop Engine (MSDE) or SQL Server 2005 Express. It includes many new features that make it a compelling upgrade proposition from either of these previous versions.

Although an upgrade from SQL Server 2005 Express is straightforward, MSDE and SQL Server Express have important differences that you must understand before you create an upgrade plan from MSDE.

SQL Server Express removes the workload governor, which limited the number of concurrent operations that MSDE could perform, and has a maximum database size limit of 4 GB as compared to 2 GB for MSDE. Combined with native 64-bit support and new SQL Server 2008 database features, most MSDE applications can be upgraded to SQL Server Express and perform more robustly.

Let’s look at the MSDE and SQL Server Express differences you must consider before you create an MSDE upgrade plan.

## Feature Changes

SQL Server 2008 Express supports all the core database functionality that both MSDE and SQL Server 2005 Express provide. This lets almost all existing database applications work without modifications. This functionality includes support for most of the new SQL Server 2005 and SQL Server 2008 features, including Common Language Runtime (CLR) support, XQuery, dynamic management views, and user-schema separation.

In addition, SQL Server Express has a new set of management tools. SQL Server Express uses the new SQL Server Computer Manager to start and stop database services. You can use the new SQL Server Configuration Manager tool to limit potential security risks by controlling network connections and shutting down unused services. You can also manage SQL Server Express by using SQL Server Management Studio Basic, which is included in SQL Server 2008 Express with Tools and SQL Server 2008 Express with Advanced Services. You can use SQL Server Management Studio Basic to manage all editions of SQL Server starting with SQL Server 2000 to SQL Server 2008.

You can download SQL Server 2008 Express from the [Microsoft SQL Server 2008 Express Web site](http://go.microsoft.com/fwlink/?LinkId=133933).

## Preparing to Upgrade

Table 10-1 shows the upgrade paths that Microsoft supports to SQL Server 2008 Express.

Table 10-1: Upgrade Paths to SQL Server 2008 Express

| **Upgrade from** | **Supported Upgrade Paths** |
| --- | --- |
| SQL Server 2000 (32-bit) MSDE SP4 | SQL Server 2008 Express |
| SQL Server 2005 (32-bit) Express | SQL Server 2008 Express  SQL Server 2008 Express with Tools  SQL Server 2008 Express with Advanced Services |
| SQL Server 2005 (32-bit) Express with Advanced Services | SQL Server 2008 Express with Advanced Services |
| SQL Server 2008 Express | SQL Server 2008 Express  SQL Server 2008 Express with Tools  SQL Server 2008 Express with Advanced Services  SQL Server 2008 Standard  SQL Server 2008 Enterprise |
| SQL Server 2008 Express with Tools | SQL Server 2008 Express with Advanced Services  SQL Server 2008 Standard  SQL Server 2008 Enterprise |
| SQL Server 2008 Express with Advanced Services | SQL Server 2008 Express with Advanced Services  SQL Server 2008 Standard  SQL Server 2008 Enterprise |
| SQL Server 2008 Express x64 (64-bit) | SQL Server 2008 Express x64 (64-bit)  SQL Server 2008 Express with Tools (64-bit)  SQL Server 2008 Express with Advanced Services x64 (64-bit) |
| SQL Server 2008 Express with Advanced Service x64 (64-bit) | SQL Server 2008 Express with Advanced Services x64 (64-bit)  SQL Server 2008 Standard x64 (64-bit)  SQL Server 2008 Enterprise x64 (64-bit) |

When you are upgrading an existing 32-bit instance to a 32-bit instance, both in-place and side-by-side upgrades are supported. In all other cases, side-by-side upgrades are required.

English SQL Server can be upgraded to any localized SQL Server. And a localized SQL Server can be upgraded to a localized version of the same language. However, localized-to-English upgrades are not supported, nor are upgrades of a localized SQL Server to different languages.

Table 10-2 shows the three types of packages available for SQL Server Express.

Table 10-2: SQL Server Express Packages

| **Feature** | **SQL Server 2008 Express** | **SQL Server 2008 Express with Tools** | **SQL Server 2008 Express with Advanced Services** |
| --- | --- | --- | --- |
| **Management** |  |  |  |
| PowerShell Integration | Yes (Separate installation)\* | Yes | Yes |
| Policy-Based Management | Yes (manual only)\*\* | Yes (manual only)\* | Yes (manual only)\*\* |
| Management Studio Basic | No | Yes | Yes |
| **SQL Engine** |  |  |  |
| Integrated Full-Text Search | No | No | Yes |
| Merge & Upsert | Yes | Yes | Yes |
| **New data type support** |  |  |  |
| Filestream support | Yes | Yes | Yes |
| New Date & Time data types | Yes | Yes | Yes |
| Geodetic data types | Yes | Yes | Yes |
| Advanced spatial libraries | Yes | Yes | Yes |
| Support for spatial standards | Yes | Yes | Yes |
| **New tools** |  |  |  |
| Import/Export Wizard | Yes | Yes | Yes |
| **Replication** |  |  |  |
| Change tracking | Yes | Yes | Yes |
| Synchronization Services | Yes (Separate installation)\*\*\* | Yes (Separate installation)\*\*\* | Yes |
| **Reporting Services** |  |  |  |
| Increase RS Memory Limit | No | No | Yes |
| RS Word/Rich Text Export | No | No | Yes |
| IIS Agnostic Report Deployment | No | No | Yes |
| Enhanced Gauges & Charting | No | No | Yes |
| Business Intelligence Development Studio | No | No | Yes |

\* The SqlPS command-line tool can be enabled in SQL Server Express by installing Windows PowerShell 1.0 before installing SQL Server Express.

\*\* Policies can be created in SQL Server Express and run manually. There is no support for automated policy-based management.

\*\*\* Synchronization Services support in SQL Server Express requires that you install the component separately from the SQL Server 2008 Feature Pack.

### SQL Server Express and MSDE Limitations

Table 10-3 lists the limits that have been set for MSDE and SQL Server Express databases.

Table 10-3: SQL Server Express and MSDE Limits

|  |  |  |
| --- | --- | --- |
| **Limitation** | **MSDE Limit** | **SQL Server Express Limit** |
| Concurrent workload governor (throttle) | Yes | No |
| Database size limitation | 2 GB | 4 GB |
| RAM support | 2 GB | 1 GB |
| SMP support | 2 (or 1 if MSDE is run on Windows 98 or Windows Millennium Edition) | 1 |

MSDE includes a workload governor that could affect performance. SQL Server Express does not include such a workload governor.

### SQL Server Express and MSDE Feature Support

Table 10-4 lists the changes in feature support between SQL Server Express and MSDE.

Table 10-4: SQL Server Express and MSDE Feature Support

| **Feature** | **Supported in MSDE?** | **Supported in SQL Server Express?** |
| --- | --- | --- |
| Merge Replication | Yes | Yes, as a Subscriber only |
| Transactional Replication | Yes, as a Subscriber only | Yes, as a Subscriber only |
| Snapshot Replication | Yes | Yes, as a Subscriber only |
| SQL Server Profiler | No | SQL Server Profiler is not installed with SQL Server Express. However, if another version of SQL Server is installed on the system, the SQL Server Profiler application that is installed with the other version can be used with SQL Server Express. |
| Database Engine Tuning Wizard | No | No |
| SQL Server Agent | Yes | No |
| Active Directory registration | No | Yes |

### Deprecated Features

All the deprecated features discussed in other chapters that apply to other SQL Server editions also apply to SQL Server 2008 Express. For details about deprecated features, see the following SQL Server 2008 Books Online topics:

* [Deprecated SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133935)
* [Deprecated Features in SQL Server 2008 Reporting Services](http://go.microsoft.com/fwlink/?LinkId=133936)

### Discontinued Functionality

All the discontinued features discussed in other chapters that apply to other SQL Server editions also apply to SQL Server 2008 Express. For details about discontinued functionality, see the following SQL Server 2008 Books Online topics:

* [Discontinued SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133937)
* [Discontinued Functionality in SQL Server 2008 Reporting Services](http://go.microsoft.com/fwlink/?LinkId=133939)

### Breaking Changes

Many of the changes discussed in other chapters that could potentially break applications also apply to SQL Server 2008 Express. For details about breaking changes, see the following SQL Server 2008 Books Online topics:

* [Breaking Changes to SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133938)
* [Breaking Changes in SQL Server 2008 Reporting Services](http://go.microsoft.com/fwlink/?LinkId=133940)

### Behavior Changes

Many of the behavior changes discussed in other chapters that apply to other SQL Server editions also apply to SQL Server 2008 Express. For more details about behavior changes that you need to watch out for, see the following SQL Server 2008 Books Online topics:

* [Behavior Changes to SQL Server Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133944)
* [Behavior Changes in SQL Server 2008 Reporting Services](http://go.microsoft.com/fwlink/?LinkId=133942)

### Upgrade Tools

#### Running Upgrade Advisor

SQL Server 2008 Upgrade Advisor helps you prepare for upgrades to SQL Server 2008. Upgrade Advisor analyzes installed components from earlier versions of SQL Server and then generates a report that identifies issues to fix either before or after you upgrade. Chapter 1, "Upgrade Planning and Deployment," describes how to use Upgrade Advisor.

When you run Upgrade Advisor, the Upgrade Advisor Home page appears. From the Home page, you can run the following tools:

* Upgrade Advisor Analysis Wizard
* Upgrade Advisor Report Viewer
* Upgrade Advisor Help

The first time you use Upgrade Advisor, run the Upgrade Advisor Analysis Wizard to analyze SQL Server components. When the wizard finishes the analysis, view the resulting reports in the Upgrade Advisor Report Viewer. Each report provides links to information in Upgrade Advisor Help that will help you fix or reduce the effect of the known issues.

You can download the Upgrade Advisor as part of the [Microsoft SQL Server 2008 Feature Pack, August 2008](http://go.microsoft.com/fwlink/?LinkID=133088).

#### Running SQL Server 2008 Upgrade Assistant

The SQL Server 2008 Upgrade Assistant is an external tool that lets you determine in a test environment how an application currently running on SQL Server 2000 or SQL Server 2005 will run on SQL Server 2008. This tool uses Upgrade Advisor, along with baseline and trace replay, in a test environment to help identify compatibility issues.

For more information about Upgrade Assistant and download instructions, see [SQL Server 2008 Upgrade Assistant](http://go.microsoft.com/fwlink/?LinkID=133150).

#### Running Best Practices Analyzer

Before upgrading your system, you also should ensure that you are using best practices for your existing system. The SQL Server 2005 Best Practices Analyzer (BPA) gathers data from Microsoft Windows and SQL Server configuration settings. BPA uses a predefined list of SQL Server 2005 recommendations and best practices to determine if there are potential issues in the database environment. There is also a BPA version for SQL Server 2000.

You can download the [SQL Server 2005 Best Practices Analyzer](http://go.microsoft.com/fwlink/?LinkID=132957) at the Microsoft Download Center.

### 64-Bit Considerations

Table 10-5 shows the supported 64-bit architectures for SQL Server 2008 Express.

Table 10-5: 64-Bit Architectures Supported by SQL Server 2008 Express

| **Architecture** | **Supported** |
| --- | --- |
| X86 (32 bit) | Yes |
| X64 | Yes |
| IA64 | No |

Table 10-6 lists the SQL Server Express packages that are available.

Table 10-6: Available SQL Server 2008 Express Packages

|  |  |
| --- | --- |
| **Package Name** | **Purpose** |
| SQLEXPR32\_x86\_ENU.exe | 32-bit only – core installation |
| SQLEXPR\_x86\_ENU.exe | 32-bit native or 32-bit WOW on 64-bit systems – core installation |
| SQLEXPR\_x64\_ENU.exe | 64-bit only – core installation |
| SQLEXPRWT\_x86\_ENU.exe | 32-bit native or 32-bit WOW on 64-bit systems – with Tools |
| SQLEXPRWT\_x64\_ENU.exe | 64-bit only – with Tools |
| SQLEXPRADV\_x86\_ENU.exe | 32-bit native or 32-bit WOW on 64-bit systems – with Advanced Services |
| SQLEXPRADV\_x64\_ENU.exe | 64-bit only – with Advanced Services |

Note that in the table, ENU refers to the English-language version. Other language versions are also available.

### System Requirements for SQL Server 2008 Express

SQL Server 2008 Express will run on the same hardware as MSDE. Table 10-7 shows the system requirements for SQL Server Express (32-bit), taken from [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=133946) in SQL Server 2008 Books Online.

Table 10-7: SQL Server 2008 Express System Requirements

| **Component** | **Requirement** |
| --- | --- |
| Processor1 | Processor type: Pentium III-compatible processor or faster  Processor speed: Minimum: 1.0 GHz  Recommended: 2.0 GHz or faster |
| Framework | SQL Server Setup installs the following software components required by the product:   * .NET Framework 3.5 SP1 * SQL Server Native Client * SQL Server Setup support files |
| Operating System | Windows XP Home Edition SP2  Windows XP Professional Edition SP2  Windows XP Tablet Edition SP2  Windows XP Media Center 2002 Edition SP2  Windows XP Media Center 2004 Edition SP2  Windows XP Media Center 2005 Edition  Windows XP Professional Reduced Media Edition  Windows XP Home Edition Reduced Media Edition  Windows Vista Ultimate Edition  Windows Vista Home Premium Edition  Windows Vista Home Basic Edition  Windows Vista Enterprise Edition  Windows Vista Business Edition  Windows Server 2008 Standard Edition RC0 or higher(note 4)  Windows Server 2008 Data Center Edition RC0 or higher (note 4)  Windows Server 2008 Enterprise Edition RC0 or higher (note 4)  Windows Server 2008 Standard Edition 64-Bit x64 RC0 or a later version (note 4)  Windows Server 2008 Data Center Edition 64-Bit x64 RC0 or a later version (note 4)  Windows Server 2008 Enterprise Edition 64-Bit x64 RC0 or a later version (note 4)  Windows Server 2003 Small Business Server Standard Edition SP2  Windows Server 2003 Small Business Server Premium Edition SP2  Windows Server 2003 Small Business Server Standard Edition R2 (note 5)  Windows Server 2003 Small Business Server Premium Edition R2 (note 6)  Windows Server 2003 SP2  Windows Server 2003 Enterprise Edition SP2  Windows Server 2003 Data Center Edition SP2  Windows Server 2003 Enterprise Edition SP2  Windows Server 2003 64-Bit x64 Standard Edition SP2 (note 2)  Windows Server 2003 64-Bit x64 Data Center Edition SP2 (note 2)  Windows Server 2003 64-Bit x64 Enterprise Edition SP2 (note 2)  Windows XP Professional Embedded Edition Feature Pack 2007 SP2  Windows XP Professional Embedded Edition for Point of Service SP2 |
| Software | SQL Server Setup requires Windows Installer 4.5 or later and Microsoft Data Access Components (MDAC) 2.8 SP1 or later. After installing required components, SQL Server Setup will verify that the computer SQL Server will be installed on also meets all other requirements for a successful installation. For more information, see [Check Parameters for the System Configuration Checker](http://go.microsoft.com/fwlink/?LinkId=133947). |
| Network Software | Network software requirements for the 64-bit versions of SQL Server are the same as the requirements for the 32-bit versions.  Supported operating systems have built-in network software.  **Note:** SQL Server does not support the Banyan VINES Sequenced Packet protocol (SPP), Multiprotocol, AppleTalk, or NWLink IPX/SPX network protocols. Clients previously connecting with these protocols must select a different protocol to connect to SQL Server.  Standalone named and default instances support the following network protocols:   * Shared memory * Named pipes * TCP/IP * VIA   **Note:** Shared memory is not supported on failover clusters. |
| Internet Software | Microsoft Internet Explorer (IE) 6 SP1 or later is required for all installations of SQL Server. IE 6 SP1 or later is required for Microsoft Management Console (MMC), SQL Server Management Studio, Business Intelligence Development Studio, the Report Designer component of Reporting Services, and HTML Help. |
| Memory2 | RAM: Minimum: 512 MB  Recommended: More than 1 GB  Maximum: SQL Server 2008 Express will use at most 1 GB of memory, regardless of how much is installed on the operating system. For best operating system performance, more than 1 GB of memory installed in the system is recommended. |
| Hard Disk | Disk space requirements will vary with the SQL Server components you install. |
| Drive | A CD or DVD drive, as appropriate, is required for installation from disk. |
| Display | SQL Server graphical tools require VGA or higher resolution: at least 1,024x768 pixel resolution. |
| Other Devices | Pointing device: A Microsoft mouse or compatible pointing device is required. |

1The System Configuration Checker (SCC) will block Setup if the requirement for processor type is not met. The SCC will warn the user but will not block Setup if the minimum or recommended processor speed check is not met. No warning will appear on multiprocessor computers.

2The SCC will warn the user but will not block Setup if the minimum or recommended RAM check is not met. Memory requirements are for this release only and do not reflect additional memory requirements of the operating system. The SCC verifies the memory available when Setup starts.

### Known Issues and Workarounds

#### Installation Messages

You might receive a message when you first install SQL Server 2008 Express that says .NET Framework 2.0 SP2 must be installed first. Although technically correct, the message is misleading because there is no installation package for this version. The correct version is .NET Framework 3.5 SP1.

For the Windows Installer, the SQL Server Express installation looks for the minimum version of 4.5.6001.22159. You can see the minimum version on your computer by checking the product version of msiexec.exe.

When you install SQL Server 2008 Express with Tools, you might be confused by some of the installation screens that say Express with Advanced Services. You can safely ignore those labels.

#### Edition Upgrades

If you upgrade SQL Server 2008 Express to SQL Server 2008 Express with Tools or SQL Server 2008 Express with Advanced Services, you will need to run the installation program twice. The feature selections provided when you run the installation program are those that relate to the version currently installed. After you update the version by running the installation program once, you will then have the option to choose additional features the next time you run the installation program.

## Upgrading from SQL Server 2000 (MSDE)

Although database administrators must contend with just a small number of issues when upgrading MSDE to SQL Server 2008 Express, you must make sure to review and understand the following upgrade issues and account for them in any MSDE upgrade plan:

* Number of instances of MSDE
* MSDE install method
* MSDE language

### Number of Instances of MSDE

The computer on which you intend to perform a database upgrade from MSDE to SQL Server 2008 Express might have multiple instances of MSDE running and also might have instances from other editions of SQL Server 2000. Each instance must be upgraded separately, and each instance might have a different set of upgrade requirements. For example, the different instances might use different languages or collation orders or be installed using a different installation method.

The first step in upgrading from MSDE to SQL Server 2008 Express is determining the instances of MSDE that are installed. Up to 50 instances of MSDE can be installed on a single system. To determine the number of installed MSDE instances on your system, review the following registry key: HKEY\_LOCAL\_MACHINE\Software\Microsoft\Microsoft SQL Server\InstalledInstances.

### MSDE Installation Method

The methodology used to install an MSDE instance affects the available upgrade method for the instance of MSDE that is being upgraded.

**Setup.** If an MSDE instance was installed by using the MSDE setup program (such as an MSI setup or the MSDE2000A.exe setup program), the SQL Server Express installation program will detect the instance, and you will be able to perform the upgrade using the in-place upgrade method.

**Merge Modules.** If an MSDE instance was installed by using an MSDE legacy installation technology known as Merge Modules, the SQL Server Express installation program will not detect the instance, and you will not be able to perform the upgrade using the in-place upgrade method. When MSDE is installed using Merge Modules, MSDE is actually installed as part of another application that uses MSDE and not installed using the MSDE standalone installer.

To determine if an MSDE instance has been installed with an MSI setup, go to Add or Remove Programs in Control Panel. If the instance appears in the listing, it was installed with an MSI setup. If it does not appear in the listing, it was installed as part of an application and should be removed by that application’s installer program.

**Important:** Although an instance of MSDE installed using Merge Modules cannot be upgraded in-place, its user databases can be upgraded individually.

### MSDE Language

The language of the MSDE installation determines the language of the SQL Server 2008 edition you are upgrading to. If you are performing an in-place upgrade by installing SQL Server Express on top of an existing MSDE installation, the language of SQL Server Express must be the same as the language of the MSDE installation or it must be set to English. SQL Server Express supports the same 12 languages that MSDE does.

**Note:** You can detach a user database from an MSDE instance and attach it to a SQL Server Express instance installed with any language.

### In-Place Upgrade

For MSDE installations that were installed by using the MSDE setup program (which creates an MSDE entry in the Add/Remove Programs list), performing an in-place upgrade is the recommended upgrade procedure. The in-place upgrade will automatically replace all the MSDE components and will also automatically upgrade all user databases.

#### In-Place Upgrade Steps

To perform an in-place upgrade from MSDE to SQL Server 2008 Express, use the following steps:

1. **Download and install Windows Installer 4.5.** Windows Installer 4.5 is required by SQL Server 2008 Express. You can download Windows Installer 4.5 from the [Microsoft Download Center](http://go.microsoft.com/fwlink/?LinkID=133102). After you have downloaded and installed it, you will likely need to perform a system reboot.
2. **Download and install .NET Framework 3.5 SP1.** The .NET Framework 3.5 SP1, available for download from the Microsoft Download Center, is a prerequisite for SQL Server 2008 Express. After downloading .NET Framework 3.5 SP1, install it by running the dotnetfx.exe program.
3. **Start the SQL Server 2008 Setup program, and install the prerequisite software.** SQL Server Express is installed by running the executable (i.e., .exe) program for the package type you are installing. The prerequisite Microsoft SQL Native Client and Microsoft SQL Server 2008 Setup Support files are installed, and the Setup program copies and installs all supporting files on the target system.
4. **Perform the system configuration checks.** The Setup program runs the system configuration checks before the actual setup begins to verify that the system meets the minimum criteria for installation and detects any pending reboot requirements. If your system fails the configuration tests, click the failed link for more information and then take the required corrective action.
5. **Determine features to install on the Feature Selection page.** By default, several features are turned off, so you must explicitly choose the components you want to install.
6. **Select the appropriate instance to upgrade on the Instance Name page.** The Setup program will detect all instances of MSDE that were installed by using the MSI installation method and. By default, Setup will select the default instance. If you want to upgrade an instance of MSDE that is not the default instance, click Installed Instance and select the MSDE instance you want to upgrade.
7. **Specify the logon information that the Setup program should use to connect to the instance being upgraded.** Generally, you will leave the default option of Windows Authentication.
8. **Specify the remaining configuration options** (generally accept all defaults), and then click Install on the Ready to Install dialog box. This will upgrade the specified instance of MSDE to SQL Server 2008 Express.

### Side-by-Side Upgrade

Although an in-place upgrade is the easiest method for upgrading from MSDE to SQL Server 2008 Express, there are situations where performing an in-place upgrade is not possible. You cannot perform an in-place upgrade when MSDE has been installed by using Merge Modules or if you want to change languages or collation. In these cases, you must use a side-by-side upgrade method in which SQL Server 2008 Express is installed at the same time as MSDE or in which the MSDE databases are detached, the MSDE instance is uninstalled, a new copy of SQL Server 2008 is installed, and the databases are reattached to the new instance. This guide takes the second approach to show you how to move from MSDE to SQL Server 2008 Express.

Another important reason for performing a side-by-side upgrade is to enable you to fully test the upgraded database without overwriting your current working database instance.

#### Side-by-Side Upgrade Steps

To upgrade from MSDE to SQL Server 2008 Express when you cannot or do not want to perform an in-place upgrade, use the following steps:

1. Log in to the SQL Server 2000 MSDE system as an administrator, and verify that the instance of MSDE or SQL Server Express that you want to upgrade is running.
2. Open a command prompt, and use sqlcmd/osql to connect to the instance you want to upgrade. To connect to the local, default instance of MSDE using Windows Authentication, use the following command:

osql -E or SQLCMD –E

To connect to a named instance, use the –S switch and specify the instance name, as shown in the following command, to connect to the desired named instance:

osql -E -S servername\instancename

1. List all the databases on the instance of MSDE by using the following commands at the osql prompt:

1> SELECT name FROM master.dbo.sysdatabases WHERE DBID > 6

2> GO

This lists all the user databases on the instance of MSDE.

1. Detach each of the user databases on the instance of MSDE by entering the following command at the osql command prompt:

1> EXEC sp\_detach\_db 'database\_name'

2> GO

This takes each of the user databases offline. Replace the value of *database\_name* with the name of the databases that you want to move from MSDE to SQL Server Express. The databases will later be attached to the new instance of SQL Server 2008.

1. Exit the osql utility by entering the following command at the osql command prompt:

1> exit

1. Shut down MSDE by opening the SQL Server Service Manager on the System Tray, and then in the Services drop-down list, select the SQL Server service, click Stop, and then click Yes.

**Note:** You can also use the Services application or the NET STOP command to stop the SQL instance.

1. Repeat Step 6 for the Distributed Transaction Coordinator and the SQL Server Agent services (if they are running).
2. Remove MSDE by using the Add/Remove Programs applet from the system’s Control Panel, selecting the entry named Microsoft SQL Server Desktop Engine, and clicking Remove.

**Important:** If MSDE was installed as part of another application, there will be no entry in the Add/Remove programs list. In this case, remove MSDE using that application’s installation program.

**Note:** This step can be skipped until a later time if you are installing SQL Server 2008 to a different instance name than the instance of MSDE being upgraded.

1. Download and install Windows Installer 4.5, which is required by SQL Server Express. You can download Windows Installer 4.5 from the Microsoft Download Center. After you have downloaded Windows Installer 4.5, install it. You will likely require a system reboot at this point.
2. Download and install .NET Framework 3.5 SP1, which is a prerequisite for SQL Server Express. You can download it from the Microsoft Download Center. After downloading .NET Framework 3.5 SP1, install it by running the dotnetfx.exe program.
3. Install SQL Server Express by running the SQL Server Express executable program. Select the appropriate installation options for the new instance you are installing, including the instance name if you want to specify a name other than SQLEXPRESS, although the use of this name is recommended.

**Important:** The Setup program will change the name of a default instance to SQLEXPRESS rather than the MSDE default of the host computer name. If you want the instance name to be the name of the host computer, you must specify that name as the named instance name.

1. After SQL Server Express is installed, start sqlcmd by opening a command prompt, typing the following command, and then pressing Enter:

sqlcmd -E

This connects you to the local, default instance of SQL Server Express using Windows Authentication. If you want to connect to a named instance, use the –S switch and specify the instance name, as shown below, to connect to the desired named instance:

sqlcmd -E -S servername\instancename

1. Attach each of the user databases that were detached from the MSDE instance by entering the following command at the sqlcmd command prompt:

1> EXEC sp\_attach\_db 'database' , 'C:\Program Files\Microsoft SQL Server\MSSQL\Data\database\_data\_filename.mdf', 'C:\Program Files\Microsoft SQL Server\MSSQL\Data\database\_log\_filename.LDF'

2> Go

Replace the values of database\_data\_filename and database\_log\_filename with the names of the database files from the user database that was detached from your previous MSDE installation. This example shows the default installation path that was used by MSDE and likely where your databases will have been located. If your installation used a custom path, you can substitute the correct path value. Repeat for each detached user database.

1. Exit the sqlcmd utility by typing *exit* and pressing Enter.
2. Enable any needed protocols.

The default installation for SQL Server Express enables shared memory, which enables local access only; the named pipes and TCP/IP protocols are disabled. If your database installation requires network access, open SQL Server Configuration Manager, open the SQL Server 2008 Network Configuration node, select Protocols for MSSQLSERVER, and then enable the required protocols by right-clicking the protocol and selecting the Enable option from the context menu.

### Performing Scripted Upgrades

The previous sections in this chapter showed you how to upgrade MSDE to SQL Server 2008 Express by using the interactive setup program. The setup program is fine for upgrading a few systems, but using the interactive upgrade process is not the most effective means of upgrading a large MSDE installation base like you might find in an enterprise environment. To accommodate installing and upgrading large numbers of systems, the SQL Server Express setup process is entirely scriptable. You can run SQLEXPR\_x86\_ENU.EXE (or the version you have downloaded) from the command line, as part of a command shell script, or from another program to perform new installations of SQL Server Express or to upgrade existing MSDE installations.

Table 10-8 shows the common parameters that might be passed to the setup program.

Table 10-8: Common Parameters Passed to the Setup Program

| Parameter | Usage |
| --- | --- |
| /q | Quiet install |
| /ACTION=Upgrade | Install, Upgrade, or Uninstall |
| /FEATURES=SQL,Tools | Select the features that need to be installed. |
| /INSTANCENAME=MSSQLSERVER | Select the instance name to be upgraded. |
| /SECURITYMODE=SQL | Is SQL Authentication to be enabled? |
| /SAPWD="StrongPassword" | If sa is to be enabled, it should have a strong password. |
| /SQLSVCACCOUNT="DomainName\UserName" | Which account will SQL Server run as? |
| /SQLSVCPASSWORD="StrongPassword" | What password is needed for that account? |

For more information about how to set up SQL Server Express from the command line, see [How to: Install SQL Server 2008 from the Command Prompt](http://go.microsoft.com/fwlink/?LinkID=132474) in SQL Server 2008 Books Online.

## Upgrading from SQL Server 2005 Express

### In-Place Upgrade

To perform an in-place upgrade from SQL Server 2005 Express to SQL Server 2008 Express, take the following steps:

1. **Download and install Windows Installer 4.5.** Windows Installer 4.5 is required by SQL Server 2008 Express and can be downloaded from the Microsoft Download Center. After you have downloaded and installed it, you will likely need to perform a system reboot.
2. **Download and install .NET Framework 3.5 SP1**, which is a prerequisite for SQL Server 2008 Express. You can download it from the Microsoft Download Center and then install it by running the dotnetfx.exe program.
3. **Start the SQL Server 2008 Setup program, and install the prerequisite software.** SQL Server 2008 Express is installed by running the executable (i.e., .exe) program for the package type you are installing.
4. **Specify the remaining configuration options** (generally accept all defaults), and then click Install on the Ready to Install dialog box. This will upgrade the specified instance of SQL Server 2005 Express to SQL Server 2008 Express.

### Side-by-Side Upgrade

To upgrade from SQL Server 2005 Express when you cannot or do not want to perform an in-place upgrade, use the following steps if you have the SQL Server Express Management Tools installed; otherwise, perform the detach/attach operations as per the MSDE upgrade instructions described earlier in this chapter.

1. Log in to the SQL Server 2005 Express system as an administrator, and verify that the instance of SQL Server Express that you want to upgrade is running.
2. Connect to the SQL Server 2005 Express system by using SQL Server Management Studio (SSMS), Express, or other edition.
3. Detach each of the user databases by right-clicking the name of the database and selecting the Detach option. (Note that you could also have done this via a backup and restore option instead, but detach/attach is generally easier).
4. Shut down SQL Server 2005 Express by opening the SQL Server Configuration Manager and stopping the SQL Server services.
5. Repeat Step 4 for the Distributed Transaction Coordinator and the SQL Server Agent services (if they are running).
6. Remove SQL Server Express by using the Add/Remove Programs applet from the system’s Control Panel. Note that you can skip this step until later if you are installing SQL Server 2008 to a different instance name than the instance of SQL Server 2005 Express being upgraded.
7. Download and install Windows Installer 4.5, which is required by SQL Server 2008 Express. You can download it from the Microsoft Download Center. After you have downloaded Windows Installer 4.5, install it. You will then likely need to reboot your system.
8. Download and install .NET Framework 3.5 SP1, which is also a prerequisite for SQL Server 2008 Express. You can also download it from the Microsoft Download Center. After downloading .NET Framework 3.5 SP1, install it by running the dotnetfx.exe program.
9. Install SQL Server 2008 Express by running the SQL Server Express executable program. Select the appropriate installation options for the new instance you are installing, including the instance name if you want to specify a name other than SQLEXPRESS, although the use of this name is recommended.

**Important:** The Setup program will change the name of a default instance to SQLEXPRESS rather than the MSDE default of the host computer name. If you want the instance name to be the name of the host computer, you must specify that name as the named instance name.

1. After SQL Server 2008 Express is installed, connect to it using SSMS (Express or other edition).
2. Attach each of the user databases that were detached from the SQL Server 2005 Express instance by right-clicking the Databases node in Object Explorer and choosing the Attach Database option. (As noted earlier, you could alternatively restore the databases at this point if you used the backup/restore option instead of detach/attach).
3. Enable any needed protocols.

The default installation for SQL Server 2008 Express enables shared memory, which enables local access only; the named pipes and TCP/IP protocols are disabled. If your database installation requires network access, open SQL Server Configuration Manager, open the SQL Server 2008 Network Configuration node, select Protocols for MSSQLSERVER, and then enable the required protocols by right-clicking the protocol and selecting the Enable option from the context menu.

## Post-Upgrade Tasks

You should verify the SQL Server 2008 Express installation by performing the following post-upgrade steps:

1. Use Configuration Manager to verify that the upgraded instance is running. To start Configuration Manager, double-click SQL Server Configuration Manager under Configuration Tools in the Microsoft SQL Server 2008 program group.
2. Within SQL Server Configuration Manager, open the SQL Server 2008 Services node and check for an upgraded instance entry to verify that it has a status of running. If the SQL Server service is not running, you can manually attempt to start it by right-clicking the entry and selecting Start from the context menu. If the service will not start, the installation was not successful and will need to be redone.

**Note:** When you are upgrading instances of SQL Server 2005 Express that support connections from networked users, it is important to know that SQL Server 2008 Express, by default, disables all remote connections. If you need to enable remote connections to SQL Server Express, open SQL Server Configuration Manager, expand the SQL Server 2008 Network Configuration node, select Protocols for MSSQLSERVER, and then enable the required protocols by right-clicking the protocol and selecting the Enable option from the context menu.

## Upgrading to Other Editions of SQL Server 2008

Although the core database capabilities of MSDE and SQL Server 2008 Express are similar, the feature sets and limitations are different. These differences or projected requirements for features outside the SQL Server 2008 Express feature set could cause you to select a different edition of SQL Server 2008 to upgrade to.

Table 10-9 compares features between MSDE and the SQL Server 2008 Express, Workgroup, and Standard editions.

Table 10-9: Comparing MSDE and SQL Server 2008 Express, Workgroup, and Standard Editions

| **Feature** | **MSDE** | **SQL Server 2008 Express** | **SQL Server 2008 Workgroup** | **SQL Server 2008 Standard** |
| --- | --- | --- | --- | --- |
| **Maximum Number of Instances** | 16 | 16 | 16 | 16 |
| **Maximum # of Processors** | 2 | 1 | 2 | 4 |
| **Maximum RAM** | 2 GB | 1 GB | 3 GB | No limit |
| **Maximum Database Size** | 2 GB | 4 GB | No lLimit | No limit |
| **Workload Governor** | Yes | No | No | No |
| **XCopy Support** | No | Yes | Yes | Yes |
| **SQL Agent** | Yes | No | Yes | Yes |
| **DTS Runtime** | Yes | Yes (Web download) | Yes (Web download) | Yes |
| **Replication Publishing** | Yes | No | Yes | Yes |
| **High Availability Features**  **(Database Mirroring and Cluster Support)** | No | No | No | Yes |
| **BI Features (Analysis Services, Integration Services)** | No | No | No | Yes |
| **Report Server** | No | Yes | Yes | Yes |
| **Service Broker** | No | Client-only | Yes | Yes |
| **Full-Text Search** | No | Yes | Yes | Yes |

### Upgrading to SQL Server 2008 Workgroup

Upgrading to SQL Server 2008 Workgroup might be a compelling option for the following three scenarios.

#### RAM Requirements Beyond the Level Supported by SQL Server 2008 Express

MSDE supports up to 2 GB of RAM and two processors, while SQL Server Express supports only 1 GB of RAM and a single processor. Although rare, some MSDE applications need more that 1 GB of RAM. If this is your case, you should consider upgrading to SQL Server 2008 Workgroup.

To quickly check MSDE RAM usage, follow these steps:

1. Press Ctl-Alt-Del.
2. Open Task Manager.
3. Check the Mem Usage column for the sqlservr.exe process.

#### Processor Requirements Beyond the Level Supported by SQL Server 2008 Express

Although MSDE supports two processors compared to SQL Server Express’s single processor, it is unlikely that this would necessitate a move to SQL Server 2008 Workgroup. In most cases, it would be more cost-effective to upgrade to a higher performance processor. SQL Server Express supports multicore processors and can be installed on any server, but each installation of SQL Server Express can access only one physical processor.

#### You Require SQL Server Agent or for the Instance to Act as a Replication Publisher

Application requirements for SQL Server Agent or for the instance to act as a replication Publisher might affect your decision about which edition of SQL Server 2008 to upgrade to.

**MSDE** – Supplies SQL Server Agent. An instance of MSDE can act as a replication Publisher.

**SQL Server Express** – Does not supply SQL Server Agent. An instance of SQL Server Express can act only as a replication Subscriber.

If your application requires SQL Server Agent, you can use the Windows Task Scheduler to schedule jobs and database tasks. You might also consider upgrading to SQL Server 2008 Workgroup.

SQL Server 2008 Express does not support using your SQL Server Express instance as a replication Publisher to other SQL Server Express databases. You would need to consider upgrading to SQL Server 2008 Workgroup.

### Upgrading to SQL Server 2008 Standard Edition

The primary reason you would consider upgrading from MSDE to SQL Server 2008 Standard is that you predict your future application requirements will exceed the capabilities or feature set available in SQL Server Express or Workgroup. This upgrade scenario would be based on projections that your future database requirements could exceed 3 GB of RAM, that you will need a database size larger than 4 GB, or that you need the high-availability or business intelligence (BI) features in SQL Server 2008 Standard.

**Note:** If you need enterprise features such as data compression, Resource Governor, or table partitioning, you will need to upgrade to SQL Server 2008 Enterprise.

## Conclusion

SQL Server 2008 Express is the ideal upgrade path for most existing SQL Server 2000 MSDE and SQL Server 2005 Express database management systems. The upgrade from SQL Server 2005 Express is straightforward, and there are only a few issues to review and prepare for before upgrading from MSDE. But make sure you understand these upgrade issues before making your move to ensure a smooth and successful upgrade.

## Additional References

For an up-to-date collection of additional references for upgrading to SQL Server 2008 Express, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkID=132510).

For the latest updates to SQL Server 2008 Books Online, see the [Microsoft SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkID=132511) .

[SQL Server 2008 Express Web site](http://go.microsoft.com/fwlink/?LinkID=133933)

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkID=132512)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkID=132514)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkID=132515)

# Analysis Services

## Introduction

SQL Server Analysis Services (SSAS) provides a powerful multidimensional Database Engine for building sophisticated OLAP and data mining solutions. Its ability to handle various data warehouse designs, efficiently store and index large amounts of data, quickly perform complex calculations, and economically manage data caching structures gives it power beyond most other multidimensional solutions on the market today. Because SSAS is included with SQL Server and is covered by the same license, many organizations have implemented SSAS to resolve their multidimensional and OLAP reporting challenges.

With SSAS 2000, Microsoft introduced many new features and capabilities to increase the functionality of its first release (OLAP Services 7.0). Features such as distinct count measures, parent/child dimensions, improved aggregation design, and data mining capabilities increased the value of SSAS 2000 and made its adoption rate among organizations higher than any other multidimensional Database Engine.

With SSAS 2005, Microsoft worked hard to further increase the value of SSAS by adding breakthrough capabilities to expand the number and kinds of solutions the platform could be used to develop and support. Changes in the underlying architecture of the product provided increased scalability, a unified model for supporting OLAP and traditional reporting needs, significant improvements in the development and administration of a given solution, and new Key Performance Indicator (KPI) and data mining features.

The release of SSAS 2008 expands the value of SSAS again by adding capabilities to cover even more business scenarios, such as improved time series forecasts, and to guide OLAP developers toward producing more efficient and effective solutions. Developer guidance comes in the form of significantly reworked wizards that streamline and simplify how to create objects while enforcing best practices learned from customer implementations of SQL Server 2005. The design tools present real-time feedback to notify the developer of deviations from best practices. New in SSAS 2008 are graphical tools to help developers build attribute relationships and examine, create, modify, and remove aggregations. New capabilities in this release include improved query performance in many cases and additional data mining models.

For information about how to upgrade data mining models, see “Data Mining” in Chapter 12.

## Preparing to Upgrade

### In-Place Upgrade vs. Side-by-Side Upgrade

You can upgrade SSAS 2000 to SSAS 2008 in one of two ways: by an in-place upgrade or a side-by-side upgrade. For SSAS 2005, only an in-place upgrade is supported. Let’s look at each of these in turn.

#### In-Place Upgrade

With an in-place upgrade, the SSAS 2000 engine and associated tools are removed and replaced by SSAS 2008. During the upgrade process, the SSAS 2000 database metadata is moved to SSAS 2008, and the upgraded databases must be fully reprocessed to populate them with data. But for updates from SSAS 2005, no database reprocessing is required. Because of functionality and feature changes with SSAS 2008, databases upgraded from SSAS 2000 might require modification before users can access the business intelligence (BI) data through the upgraded database, although this is not the case for SSAS 2005 databases.

After the in-place upgrade is complete, only SSAS 2008 will remain. An in-place upgrade is an all-or-nothing approach; if an in-place upgrade fails, you must roll back to an earlier version (there is a go/no-go point in the Setup program before which you can cancel the upgrade). To roll back to SSAS 2000 after an upgrade to SSAS 2008 is complete, you can reinstall SSAS 2000 and restore the SSAS databases. To roll back to SSAS 2005 after an upgrade to SSAS, you have to uninstall SSAS 2008, restart, reinstall SSAS 2005, and restore the SSAS databases. Downtime because of upgrade problems can be significant and backups of the SSAS 2000 or SSAS 2005 databases are critical.

#### Side-by-Side Upgrade

With a side-by-side upgrade, you install an instance of SSAS 2008 alongside SSAS 2000, which remains until uninstalled. During the upgrade process, users can continue to access the databases in SSAS 2000, which is unaffected by the upgrade process. After a side-by-side upgrade is complete, both SSAS 2000 and SSAS 2008 are installed. You can move and test database metadata without affecting the SSAS 2000 installation. After SSAS 2008 is fully tested, SSAS 2000 can be uninstalled.

**Note:** With a side-by-side upgrade, you can either use the existing server environment for the new installation or install SSAS 2008 on a new server.

**Important:** The side-by-side upgrade option provides for greater availability during the upgrade process, simplifies rollback (should that be required), and results in simpler testing scenarios because both versions are available at the same time.

Both upgrade options will result in SSAS 2008 versions of the databases from a given instance of SSAS 2000 or SSAS 2005. However, given the new features and advances that SSAS 2008 brings, you should consider redesigning SSAS 2000 databases to take advantage of the new platform.

### Determining and Evaluating Potential Upgrade Issues

Regardless of whether you decide to perform an in-place upgrade or a side-by-side upgrade of SSAS 2000 or SSAS 2005 to SSAS 2008, there are a range of potential issues that you might face during an upgrade. To obtain a report that identifies many of these potential issues before you start an upgrade, you should run the Microsoft SQL Server 2008 Upgrade Advisor to analyze the databases on an existing instance of SSAS 2000 or SSAS 2005. The Upgrade Advisor helps you determine whether you will encounter any of these issues during an upgrade. If the Upgrade Advisor reports any of these issues, follow its recommendations and guidance for possible mitigation options and strategies. For more information about how to install and run this tool, see “Upgrade Planning and Deployment” in Chapter 1. There is also a category of issues that either cannot be detected by the Upgrade Advisor or the detection of the issue would result in too many false-positive results.

The following section discusses the most important upgrade issues, whether detected by the Upgrade Advisor or not. For a complete list of backward-compatibility issues, breaking changes, and behavior changes to SSAS 2008, see the [SQL Server Analysis Services Backward Compatibility](http://go.microsoft.com/fwlink/?LinkId=132476) in SQL Server 2008 Books Online.

### Issues Preventing an Upgrade

Generally, invalid objects in a database will prevent it from being upgraded. Note, however, that the problem of invalid objects applies only to SSAS 2000 because no objects were invalidated between SSAS 2005 and SSAS 2008. Therefore, before you start an upgrade process, ensure that all the objects in each database on a server are valid and can be processed and queried.

### Deprecated Features

Table 11-1 describes the most common SSAS objects and settings that are deprecated in SQL Server 2008, which means that they will not be supported in future releases of SQL Server.

Table 11-1: Deprecated Objects and Settings

| **Deprecated Feature/Functionality** | **Comments** |
| --- | --- |
| InsertInto (Connection string property) | Original connection string syntax for populating local cubes. |
| CreateCube (Connection string property) | Original connection string syntax for populating local cubes. Not supported in ADOMD.NET. |
| SQL Server 2000 Predictive Model Markup Language (PMML) | This feature was replaced with standard PMML. |
| Create Action | Extended syntax is supported, but English statements cannot create new kinds of actions. DDL should be used instead. |
| CalculationPassValue |  |
| CalculationCurrentPass |  |
| NON\_EMPTY\_BEHAVIOR query optimizer hint turned on by default | By default, the NON\_EMPTY\_BEHAVIOR query optimizer hint will be turned off in a future release. It is an MDX optimization hint that can produce incorrect results when it is not used correctly. |
| CELL\_EVALUATION\_LIST intrinsic cell property | Blank in SSAS. |
| COM assemblies | COM assemblies might pose a security risk. |

For more information about deprecated features in SSAS 2008, see [Deprecated Analysis Services Functionality in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132477) in SQL Server 2008 Books Online.

### Discontinued Functionality

Some objects and settings in an SSAS 2000 database cannot be directly upgraded to SSAS 2008, although SSAS 2005 does not have this problem. The SSAS 2000 issues are typically due to architectural changes or feature changes that prevent a direct mapping of objects and settings to the new version of the platform. In some cases, these objects and settings are upgraded to replacement features that accomplish the same result; in other cases, replacement objects and features can be added to the resulting databases after an upgrade is complete. Before you start an upgrade process, develop a strategy for handling these kinds of issues.

Table11-2 describes the most common objects and settings that cannot be upgraded to SSAS 2008 because of discontinued or changed functionality.

Table 11-2: Discontinued Objects and Settings

| **Discontinued Feature/Functionality** | **Explanation/Replacement Feature/Corrective Action** |
| --- | --- |
| Mining Execution Location | Accepted for backward compatibility, but is ignored. Execution is always on the server. |
| Mining Location |  |
| Log File | The Log File feature is replaced by the Trace feature. |
| Execution Location |  |
| Distinct Measures by Key |  |
| Large Level Threshold |  |
| Aggregated Providers | This feature is replaced by plug-in algorithms. |
| Linked Cubes | This feature is replaced by linked measure groups. |
| Custom Level Formulas | This feature is replaced by MDX scripts. |
| Cube and Database Role commands | No longer supported |
| CreateVirtualDimension |  |
| CreatePropertySet |  |
| Ignore | In SSAS 2000, this function was reserved for future use. |
| Active Directory Registration |  |
| Skipped levels in parent/child hierarchies | The SkippedLevelsColumn property, which indicates how many levels to skip when you create a member under a parent in a parent/child hierarchy, is no longer supported. |
| Surface Area Configuration Tool |  |

For more information about discontinued features in SSAS 2008, see [Discontinued Analysis Services Functionality in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132475) in SQL Server 2008 Books Online.

### Breaking Changes

There are many changes between SSAS 2000 and SSAS 2008 that are significant and result in breaking changes. These changes were actually introduced in SSAS 2005, and SSAS 2008 merely behaves in the same manner. Table 11-3 covers these changes.

Table 11-3: Breaking Changes Between SSAS 2000 and SSAS 2008

| **Breaking Change** | **Description/Recommended Action** |
| --- | --- |
| An object that depends on a linked object is not upgraded. | Linked cubes and linked dimensions are not upgraded by the Upgrade Advisor in SSAS 2008. Therefore, objects that refer to a linked cube or a linked dimension cannot be upgraded because the linked object on which the object is based cannot be upgraded. For example, an OLAP mining model that is based on a linked cube cannot be upgraded because the linked cube on which the mining model is based cannot be upgraded. |
| Autoexist can produce different query results when multiple hierarchies are upgraded into the same dimension. | When multiple hierarchies or virtual dimensions are upgraded from SSAS 2000 into the same SSAS 2008 dimension, querying the upgraded hierarchies that the dimension contains might produce different results than querying the same hierarchies in SSAS 2000. This is because autoexist functionality automatically removes tuples that do not exist in the dimension from any cross-join of sets that contains members from the upgraded hierarchies. To resolve this issue, you should review calculations that involve multiple hierarchies in the same dimension. |
| Browsing experience is different when disabled levels are used. | Since SSAS 2005, hidden or disabled levels in hierarchies are no longer supported. Hidden or disabled levels are upgraded as visible levels. Calculations that involve hierarchies that contain such levels might return unexpected results. After you upgrade, review and verify calculations that involve hierarchies that previously contained hidden or disabled levels. |
| Bucketing might be different for grouping levels. | Since SSAS 2005, automatic grouping might return a different set of member groups. Calculations that rely on these member groups might return unexpected results. After you upgrade, review and verify calculations that rely on member groups. |
| Conversion from neutral language to specific language might produce unexpected results. | In SSAS 2000 and earlier versions, SSAS used only neutral language identifiers, also known as primary language identifiers—for example, LANG\_ENGLISH (0x09) for English and LANG\_CHINESE (0x04) for Chinese.  To support translation and collation options, SSAS now uses specific language identifiers, which are a combination of a primary language identifier and a sublanguage identifier for a specific culture. For example, the combination of the primary language identifier LANG\_ENGLISH (0x09) and the sublanguage identifier SUBLANG\_ENGLISH\_AUS (0x03) describes Australian English.  Upgrading from neutral to specific language identifiers can change the expected translation and collation behavior and can produce unexpected results. After you upgrade, review and validate objects such as dimensions, hierarchies, and members for which the language identifier has changed. |
| Cube role commands are not supported. | SSAS 2008 does not support command objects on cube roles and will not upgrade commands from earlier versions. |
| Custom level formulas aggregate differently. | If a cube contains a dimension with custom level formulas, and also contains dimensions both before and after the dimension with custom member formulas or unary operators, the cube might return different results than earlier versions of SSAS. This occurs because calculation precedence rules have changed. |
| Custom member formulas and custom rollup formulas are upgraded into Multidimensional Expressions (MDX) script. | In earlier SSAS versions, the following properties are supported on dimensions and levels: custom rollup formulas, custom member formulas, all member formulas, and custom level formulas. In SSAS 2005, these properties were replaced by functionality supported in MDX scripts and are upgraded to MDX scripts during the upgrade process. |
| Custom aggregations are not upgraded. | Aggregations that were manually generated in earlier versions of SSAS are not upgraded in SSAS 2008. Only aggregations generated by the Storage Design Wizard are upgraded. To resolve this issue, manually create the aggregations by using XML for Analysis (XMLA) scripts. |
| Data members always exist in parent/child dimensions. | Earlier versions of SSAS gave you the option of excluding data members in parent/child dimensions. The DataMembers property of a dimension in earlier versions of SSAS supported three options: None, Hidden, or Visible. The None option is not available in SSAS 2005 or SSAS 2008. Data members are always included in parent attributes. To hide the data members in parent/child dimensions, you can set the DataMembers property of a dimension to Hidden. The MembersWithData property for the parent attribute supports only two options: NonLeafDataHidden or NonLeafDataVisible. |
| Database role commands are not supported. | SSAS 2008 does not support command objects on database roles and will not upgrade commands from earlier versions of SSAS. |
| DefaultMember is upgraded into MDX script. | In earlier versions of SSAS, the default member of a dimension is specified by an MDX expression, which is contained in the DefaultMember property of the dimension. In SSAS 2005, this property was replaced by functionality supported in MDX scripts, and the property is upgraded to a MDX script. |
| Dimension and hierarchy renaming by the upgrade process might cause different query results. | Dimension hierarchies in SSAS 2000 are internally represented as separate dimensions, and a naming convention is used to identify them. Upgrading to SSAS 2008 might create a separate dimension with a new name for each dimension hierarchy instead of combining the dimension hierarchies together under the parent dimension because auto-exist results in different security rules than would apply in earlier versions of SSAS. |
| Drillthrough report settings are not upgraded. | Although drillthrough reports exist in SSAS 2008, drillthrough report settings are not upgraded from earlier versions of SSAS. |
| Linked cubes are not upgraded. | Earlier versions of SSAS supported linked cubes. In SSAS 2005, this feature was replaced by linked dimensions and linked measure groups. |
| Member unique names might change during upgrade. | SSAS tries to preserve the unique names of members during the upgrade, but there are certain circumstances in which the unique name for a member is changed. If member unique names change, client applications, MDX expressions, and other properties that depend on member unique names might produce unexpected results. |
| ODBC data sources are not supported. | Earlier versions of SSAS let you use ODBC data sources, but this functionality is no longer supported. |
| Remote partitions are not upgraded. | Remote partitions are not upgraded from SSAS 2000 to SSAS 2008. Upgrade the server to SSAS 2008, and then manually create the remote partitions. |
| Some mining model algorithm parameters are not supported. | Earlier SSAS versions support using the MINIMUM\_LEAF\_CASES parameter with the Microsoft Decision Trees algorithm, and the MINIMUM\_CLUSTER\_CASES parameter with the Microsoft Clustering algorithm. Since SSAS 2005, both parameters were renamed to MINIMUM\_SUPPORT. If these parameters were used in mining models that were created by using earlier SSAS versions, the parameters are not upgraded. |
| The CREATE KPI command introduces a new keyword. | A new keyword, KPI, was introduced to the CREATE KPI command. If existing objects have the name KPI, the new keyword will conflict with them. |
| Top Level for dimension security is not supported. | In earlier SSAS versions, you could specify dimension security so that a user saw a top level that is different from the top level of the hierarchy. However, members that are secured using the Top Level setting will be visible after the upgrade. |

Microsoft tried to avoid breaking changes between SSAS 2005 and SSAS 2008. However, there are a few issues that cause breaking changes between the two versions. Table 11-4 lists these changes.

Table 11-4: Breaking Changes Between SSAS 2005 and SSAS 2008

| **Breaking Changes Feature/Functionality** | **Description** |
| --- | --- |
| Visual Basic for Applications (VBA) functions handle null values and empty values differently than in SSAS 2005. | In SSAS 2005, VBA functions return 0 or an empty string when either null values or empty values are used as arguments. In SSAS 2008, they return null. |
| The Analysis Services Migration Wizard will fail because Decision Support Objects (DSO) is not installed by default. | By default, SSAS 2008 does not install the DSO backward-compatibility component. The backward-compatibility package is installed by default, but the DSO component of the package will be disabled. Because the SSAS Migration Wizard relies on this component, it will fail unless the component is installed. To install the DSO component, do the following:   * Open Control Panel. * In Windows XP or Windows Server 2003, select Add or Remove Programs. In Windows Vista and Windows Server 2008, select Programs and Features. * Right-click Microsoft SQL Server 2005 Backward Compatibility and select Change. * In the Backward Compatibility Setup wizard, click Next. * On the Program Maintenance page, select Modify, and then click Next. * On the Feature Selection page, if DSO is not available, click the down arrow and select *This feature will be installed on local hard drive*. Click Next. * On the Ready to Modify the Program page, click Install. * When installation is complete, click Finish.   You can remove DSO after the upgrade is complete by following the previous steps and changing the option for DSO to *This feature will not be available*.  If the backward-compatibility package is not installed, you can install it from the SQL Server 2008 distribution media. Be aware that there are versions for each target architecture (that is, x86, x64, IA-64). These versions can be found at the following respective locations:   * x86\Setup\x86\SQLServer2005\_BC.msi * x64\Setup\x64\SQLServer2005\_BC.msi * ia64\Setup\ia64\SQLServer2005\_BC.msi |
| We do not recommend that you put the partition location in the Data folder. | The server manages the Data folder and creates or drops folders as objects are created, deleted, and altered. Therefore, specifying a partition storage location inside the Data folder is strongly discouraged, especially in the subfolders for databases, cubes, and dimensions. Although the server lets you do this with Create or Alter, it will display a warning. When you upgrade databases from SSAS 2005 to SSAS 2008 that have partition storage locations in the Data folder, it will work. |
| You might get unexpected results for queries that use the EXISTING MDX keyword in ProClarity Analytics Server and Microsoft Office PerformancePoint Server 2007 (PPS). | ProClarity Analytics Server and PPS 2007 use the EXISTING keyword in MDX incorrectly in certain scenarios. Because of changes in SSAS 2008, these queries might return unexpected results. |

For more information about breaking changes in SSAS 2008, see [Breaking Changes to Analysis Services Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132482) in SQL Server 2008 Books Online.

### Behavior Changes

In some cases, upgraded databases will behave differently in SSAS 2008 than they did in SSAS 2000. This is typically due to architectural changes or feature changes in the new version of the platform. In some cases, these behavioral changes will not cause problems when querying the databases. If query problems do arise, the behavioral changes can usually be resolved by making changes to the design of the upgraded databases. Before you start an upgrade process, understand what implications, if any, these issues might have on databases involved in the upgrade. Table 11-5 discusses the most important of these changes.

Table 11-5: Behavior Changes Between SSAS 2000 and SSAS 2008

| **SSAS 2000 Behavior** | **SSAS 2008 Behavior** |
| --- | --- |
| Function: CreateVirtualDimension | Expression will return Error. |
| Function: CreatePropertySet | Expression will return Error. |
| Supports the option to specify dimension security so that a user can view a top level that differs from the top level of the hierarchy. | Top Level for dimension security is not supported. Members that are secured by using the Top Level setting will be visible after the upgrade. |
| AutoCommit and AutoRollback in update | Tries to commit a non-existing transaction. In SSAS 2000, the Update statement created its own transaction. In SSAS 2008, it does not. |

For more information about breaking changes in SSAS 2008, see [Behavior Changes to Analysis Services Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132483) in SQL Server 2008 Books Online.

### 64-bit Considerations

SSAS 2005 and SSAS 2008 are available for 64-bit and 32-bit hardware platforms. You should perform an in-place upgrade using the same platform edition you already have installed. Therefore, the 64-bit edition of SSAS 2005 for the Itanium platform should be upgraded to the same edition of SSAS 2008.

SSAS 2000 is available in 64-bit only on the IA-64 platform. When you perform a side-by-side upgrade using two servers, you can upgrade from one hardware platform edition to another. For example, you can upgrade the 64-bit edition of SSAS 2000 for the Itanium platform to the 64-bit edition of SSAS 2008 for the x64 platform. Because there is no version of Decision Support Objects (DSO) on IA-64, you should run the Migration Wizard from a 32-bit computer to upgrade the databases.

If you want to upgrade from the 32-bit edition of SSAS 2000 or SSAS 2005 to the 64-bit edition of SSAS 2008 for the Itanium platform, be aware that the Business Intelligence Development Studio is not available for the Itanium platform. Therefore, all development tasks related to SSAS 2008 for the Itanium platform must be done from another client or server that has Business Intelligence Development Studio loaded.

## Upgrading from SQL Server 2000

It is important to realize that the upgrade from SSAS 2000 to SSAS 2008 is almost identical to the upgrade from SSAS 2000 to SSAS 2005. Therefore, this section covers the basics of the upgrade and highlights any differences between upgrading to SSAS 2005 versus SSAS 2008. You can find more information about this process in the [SQL Server 2005 Upgrade Technical Resource Guide](http://go.microsoft.com/fwlink/?LinkID=71736).

As with any upgrade, one of the most important steps in the process is effective preparation. Preparing for an upgrade should include two important steps: checking for possible upgrade issues and planning for a failed upgrade. The tables in the “Preparing to Upgrade” section earlier in this chapter list the known issues that might affect a given upgrade process. Although most of these issues will not prevent a given upgrade from completing, some might require design changes after the upgrade to provide the same user experience. Before you attempt an upgrade, review these tables, and determine whether any of the listed issues will affect the upgrade results.

As stressed throughout this guide, run the Upgrade Advisor to analyze the instance of SSAS­, and then review the generated report to verify that you have addressed all issues that must be resolved before the upgrade and that you understand the upgrade issues that you must resolve after Setup is complete. “Upgrade Planning and Deployment” in Chapter 1 also covers running other tools to help with an upgrade preparation and post-upgrade tasks, including the SQL Server 2008 Upgrade Assistant and the SQL Server 2000 version of Best Practices Analyzer (BPA).

Before you start an in-place upgrade, ensure that a failed upgrade can be rolled back. Although the in-place upgrade process should handle most situations, unforeseen problems might occur and result in a failed upgrade. In extreme cases, a failed upgrade could even result in an unusable SSAS 2000 installation. Therefore, planning for a failed upgrade process is important.

**Important**: With an in-place upgrade, the upgrade process handles all aspects of the upgrade, automatically upgrading the metadata for each database found in SSAS 2005. However for SQL Server 2000, the upgrade process will not automatically reprocess the upgraded databases. Each database must be fully processed after the upgrade to ensure users can access the data that is contained in each database.

If a failed upgrade occurs, frequently the easiest resolution is to reinstall SSAS 2000 and restore the installation to its state before the upgrade process was started. To ensure that all the data and configuration information that is needed to restore the existing installation is available, follow these steps before the upgrade process starts:

1. Back up the registry information related to SSAS 2000. Using Registry Editor, export the following registry key to a file:

My Computer\HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\OLAP Server

1. Back up all databases by using the archive command in Analysis Manager. Open Analysis Manager, right-click each OLAP database listed, and then click Archive Database. Provide a unique file name for each .cab file that is created.
2. Back up the SSAS 2000 repository. This is either an Access database named msmdrep.mdb and located in \Program Files\OLAP Services\Bin\, or a SQL Server database.

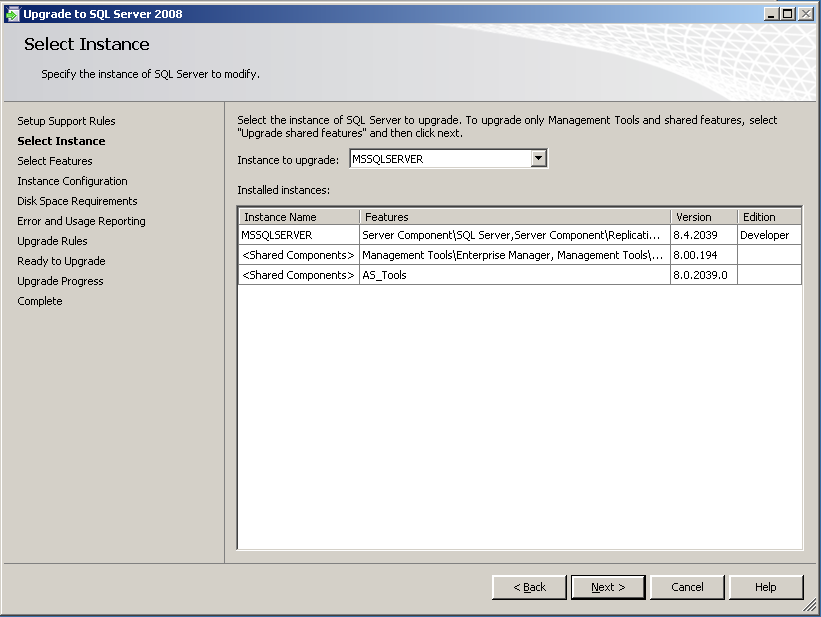
We recommend that you put all files that are generated by the previous steps in a single directory on a network share for safe-keeping during the upgrade process.

### In-Place Upgrade

To start an upgrade, start the Setup application for SQL Server 2008. Select Installation and then *Upgrade from SQL Server 2000 or SQL Server 2005***.** The Setup program will run a system configuration check, collect system information, and prompt for a product key. After it installs some prerequisites, the Setup application will then prompt for the kind of installation to be performed.

**Note:** If you plan to keep the Foodmart 2000 cube after an upgrade, see the “About the Foodmart Sample Database” section in this chapter before you start the upgrade.

After you decide to upgrade from SQL Server 2000 or SQL Server 2005, the Setup program asks for a product key, prompts for the acceptance of the license file, and then installs setup support files. After the setup support files are finished, the setup support rules are checked. The next screen prompts you to select the instance to be upgraded, as shown in Figure 11-1.



**Figure 11-1:** Select the instance of SQL Server 2000 to upgrade

Next is the Feature Selection page, which lets you select the features that you want to install. Frequently when you run Setup and perform an upgrade, you see that all components are selected and no changes can be made.

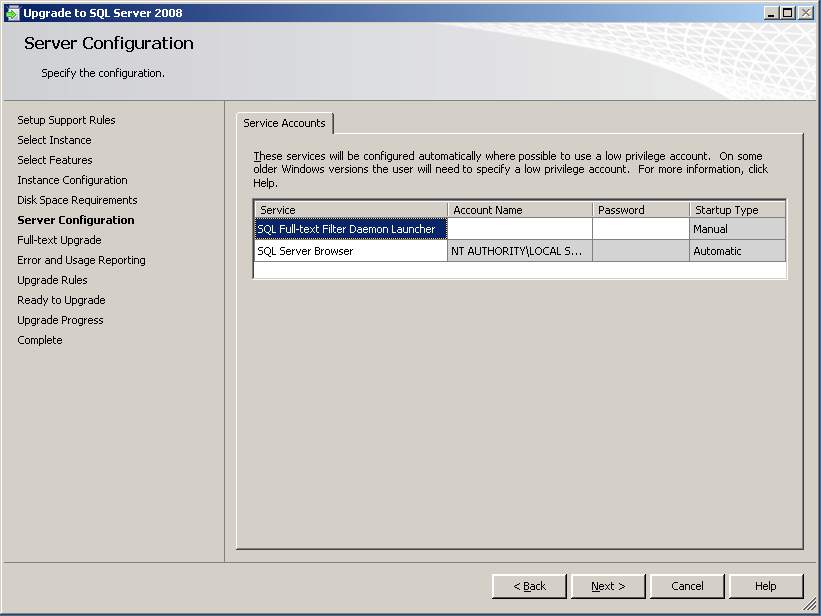
If you cannot select different options, you will have to run Setup again to add features. For example, if Business Intelligence Development Studio was not already installed on the server but was needed in the future, a second run of the Setup application would be required. After you select the components to install, the Setup application prompts for an instance name for the newly installed SQL Server 2008 components. To upgrade an existing installation of SSAS 2000, leave the InstanceID the same and continue. The Setup application should detect any running services based on which components were selected for installation. Ensure that the existing installation of SSAS 2000 is selected and continue.

The next screen, Disk Space Requirement, confirms that sufficient disk space exists on the target drives to install the selected components.

The Setup application will automatically install a new service called the SQL Browser service. The SQL Browser service listens for incoming requests for SQL Server resources and provides information about SQL Server instances installed on a server.

By default, the Setup application will try to use an account that has minimal permissions for the SQL Server Browser service. On some older versions of Windows, the wizard will prompt for what credentials to use for the newly installed service. The credentials can be specified by using a built-in system account or a domain user account. The setting should be based on what security policies are defined for standard Windows services in your organization; if no standard security policies are defined, the service account credentials should be set using either the Local Service built-in account or a domain user account.

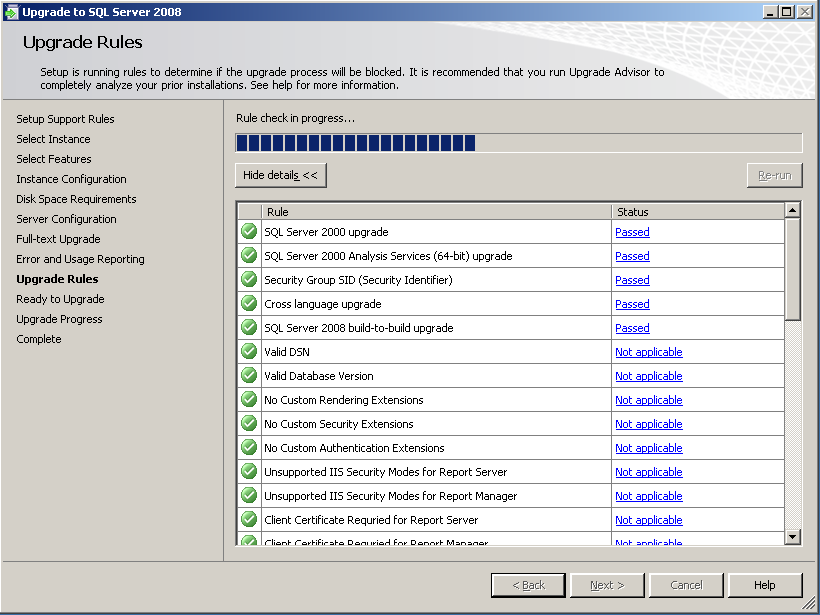
The newly installed SSAS 2008 service will be configured to use the same service account credentials the existing SSAS 2000 service is currently using. If the credentials need to be changed, you can use the SQL Server Configuration Manager to do this after the upgrade process is complete. Figure 11-2 shows the screen that you use during the Setup application to specify the service account credentials for the SQL Server Browser service.



**Figure 11-2:** Service account credentials for new SQL Server Browser service

The Setup application will prompt the user to turn on Error and Usage Reporting. These are optional settings and are off by default. Enabling them means the server might try to send data to Microsoft on an as-needed basis.

The Setup application then checks a series of upgrade rules. You should examine any failures in the rules and address as necessary. Figure 11-3 shows the results of these rules, and because there are no failures, installation can continue.



**Figure 11-3:** Upgrade rules

When the Setup application is ready to continue with the upgrade, it will display a summary of actions that it will take. Ensure that the action to be taken is “upgrade,” and then continue.

Clicking the Upgrade button will let the Setup application start the upgrade. During the upgrade, an Upgrade Progress status screen will show status information about the various steps taken by the Setup application. When the upgrade process is complete, you see a screen that shows the steps together with a success or failure message. A final screen will provide a summary of the installation together with any notes that are relevant to the upgrade process.

After the upgrade has finished, you should follow a short set of post-installation tasks to ensure the upgrade completed successfully. For more information, see “Post-Upgrade Tasks” later in this chapter.

#### Troubleshooting a Failed Upgrade

Should the in-place upgrade fail, the best course of action is to review the setup logs that the Setup application created.

* Review the Summary.txt file that is located in the %Program Files%\Microsoft SQL Server\100\Setup Bootstrap\Log directory. If any error messages are listed, take whatever actions are required to correct the situation, and try the upgrade process again.
* If no error messages are included in the summary, review the Summary\_*[ComputerName]*\_*[date]\_[time]*.log file in the %Program Files%\Microsoft SQL Server\100\Setup Bootstrap\Log\*[date]\_[time]* directory. When you review the file, search for any instances of “Failed” for a Status (which indicates a setup error). If any error messages are listed, take whatever actions are required to correct the situation, and try the upgrade process again.

#### Post-Upgrade Tasks

After you have completed the upgrade of an SSAS 2000 server to SSAS 2008, you must complete a series of post-installation tasks before the upgraded databases will be available to users. In addition, you should perform other post-installation tasks to ensure that each database is working correctly and can be modified in the future if it is necessary.

**Review upgraded databases.** Each database upgraded by the SQL Server 2008 Setup application should be reviewed to ensure the upgrade process completed successfully. Using SQL Server Management Studio, connect to SSAS on the upgraded server. If the workstation components were installed as part of the upgrade, SQL Server Management Studio should be available on the upgraded server; otherwise, SQL Server Management Studio will have to be started on another server or workstation that has had the workstation components for SQL Server 2008 installed.

After a connection to SSAS on the upgraded server is established, expand the Databases folder in the Object Explorer window. If the Object Explorer window is not visible, open the View menu and select Object Explorer. The Auto Hide button, represented by a pushpin in the upper-right corner of the Object Explorer window, can be used to “pin” the window so that it stays open.

In the Databases folder, review the structure of each database that was upgraded. In particular, review the list of dimensions and cubes to see whether the structure of the database is generally the same as it was before the upgrade process was completed. Browsing the dimensions and cubes will not be possible until the next step, processing the databases, is complete.

**Process upgraded databases.** To browse the dimensions and cubes in each upgraded database, each database must be processed. You can do this by using SQL Server Management Studio. Right-click each database listed in the Databases folder and select Process.

After you have selected the Process menu option, SQL Server Management Studio will provide a Process Database dialog box. Ensure that the database selected is listed in the Object Name text box (in the Object List grid) and Process Full is listed in the Process Options text box. Click OK, and SQL Server Management Studio starts to process the selected database. Repeat the processing action for each database listed in the Databases folder.

In some cases, issues with an upgraded database might prevent it from processing. Most of these issues should be reported by Upgrade Advisor. If any such issues exist, you might have to create a Business Intelligence Development Studio project so that you can change the database and redeploy it for processing. See “Create Development Projects for Upgraded Databases” in this chapter for information about how to create Business Intelligence Development Studio projects for upgraded databases.

#### About the Foodmart Sample Database

SSAS 2000 includes a sample OLAP database named Foodmart, which is automatically created when you install SSAS 2000. Therefore, unless it was explicitly deleted, the database will likely exist when a server is upgraded to SSAS 2008. In most cases, the Foodmart database is no longer needed; SSAS 2008 includes two sample databases, AdventureWorks and AdventureWorksDW. These new sample databases are included with the SQL Server samples installed as part of the SQL Server 2008 samples available at [CodePlex](http://go.microsoft.com/fwlink/?LinkID=85384&clcid=0x409). Therefore, the Foodmart database can be safely deleted from SSAS either before or after the upgrade is complete. However, if you must keep the Foodmart database (for testing or training purposes, for example), you have to follow a few steps before and after an in-place upgrade process is complete.

Before you perform an in-place upgrade, you need to save to a backup directory the Microsoft Access database that serves as the source for the Foodmart database because the upgrade process deletes the Microsoft Access database file. Therefore, unless it is saved, the database file will not be available for use by the upgraded Foodmart database. By default, you can find the Microsoft Access database file at %Program Files%\Microsoft Analysis Services\Samples\foodmart 2000.mdb.

After the database file is saved to a backup directory, you can perform an in-place upgrade. After an in-place upgrade is complete, the upgraded Foodmart database must be modified to process cleanly. Take the following steps:

1. Create a Business Intelligence Development Studio project that reflects the design of the upgraded Foodmart database. For information about how to do this, see “Create Development Projects for Upgraded Databases” later in this chapter.
2. After the Business Intelligence Development Studio project is created and open, update the data source for the database. SSAS 2008 does not support ODBC data sources by using native ODBC drivers or OLE DB Provider for ODBC. Therefore, the data source (known as Foodmart.ds in the Business Intelligence Development Studio project) must be updated to use Microsoft Jet 4.0 OLE DB Provider. When you edit the data source, provide the full path and file name of the saved copy of the Microsoft Access database file.
3. After you update the data source, update the primary data source view (DSV), which is named Foodmart.dsv. Specifically, update the three calculated columns defined in the DSV to remove the double quotation marks that are included in the calculation, as Table 11-6 describes.

Table 11-6: Updating Calculated Columns

| **Table** | **Calculated Column** | **Updated Definition** |
| --- | --- | --- |
| Customer | Column1 | Fname+' '+lname |
| sales\_fact\_1997 | Column1 | store\_sales-store\_cost |
| Inventory\_fact\_1997 | Column1 | warehouse\_sales-warehouse\_cost |

1. Finally, update the Warehouse cube (named Warehouse.cube). Specifically, update the two partitions defined in the cube so that their source queries do not contain double quotation marks. Table 11-7 shows updated versions of the source queries used.

Table 11-7: Updated Versions of Source Queries

|  |  |
| --- | --- |
| **Warehouse Cube Parition** | **Source Query** |
| Warehouse | SELECT inventory\_fact\_1997.store\_invoice, inventory\_fact\_1997.supply\_time, inventory\_fact\_1997.warehouse\_cost, inventory\_fact\_1997.warehouse\_sales, inventory\_fact\_1997.units\_shipped, inventory\_fact\_1997.units\_ordered, warehouse\_sales-warehouse\_Cost AS Column1, inventory\_fact\_1997.product\_id, inventory\_fact\_1997.warehouse\_id, inventory\_fact\_1997.store\_id, inventory\_fact\_1997.time\_id FROM inventory\_fact\_1997, time\_by\_day WHERE inventory\_fact\_1997.time\_id=time\_by\_day.time\_id AND time\_by\_day.the\_year=1997 |
| warehouse 98 | SELECT inventory\_fact\_1997.store\_invoice, inventory\_fact\_1997.supply\_time, inventory\_fact\_1997.warehouse\_cost, inventory\_fact\_1997.warehouse\_sales, inventory\_fact\_1997.units\_shipped, inventory\_fact\_1997.units\_ordered, warehouse\_sales-warehouse\_Cost AS Column1, inventory\_fact\_1997.product\_id, inventory\_fact\_1997.warehouse\_id, inventory\_fact\_1997.store\_id, inventory\_fact\_1997.time\_id FROM inventory\_fact\_1997, time\_by\_day WHERE inventory\_fact\_1997.time\_id=time\_by\_day.time\_id AND time\_by\_day.the\_year=1998 |

After you have made these changes, the Business Intelligence Development Studio project can be deployed and processed successfully. The cubes in the resulting Foodmart database can then be browsed and used as expected. They show the same results as the pre-upgraded version of the database.

After a database is processed, you can use SQL Server Management Studio to browse its dimensions and cubes. To start, expand a database in the Object Explorer window in SQL Server Management Studio to display a list of folders under each database. Then, expand the Dimensions and Cubes folders, right-click a given dimension or cube, and select Browse.

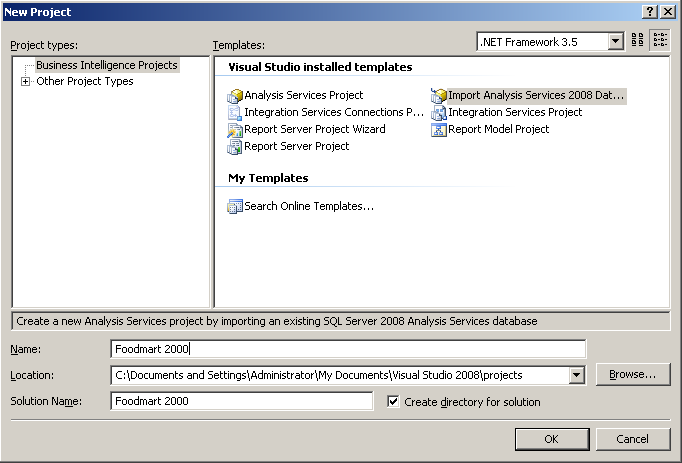
#### Create Development Projects for Upgraded Databases

SSAS 2008 uses a different paradigm for developing and managing databases. Although SQL Server Management Studio is used for management tasks (such as processing a database or objects, changing certain management properties for objects, and handling backup and restore operations), development tasks are now handled through the new Business Intelligence Development Studio application. Business Intelligence Development Studio takes advantage of the Visual Studio 2008 IDE by using specific project and design features for SSAS databases. Therefore, if you want to change the design of a given database, a Business Intelligence Development Studio project must exist with the “source code” for the database (in the form of dimension, cube, and other object definitions).

For new SSAS 2008 databases, you can use Business Intelligence Development Studio to create a new project to house the dimensions, cubes, and other objects in the database. For databases created as the result of an upgrade process, you can use Business Intelligence Development Studio to reverse-engineer a database into a Business Intelligence Development Studio project. To support future development changes to the databases that were involved in the upgrade, you should create a Business Intelligence Development Studio project for each database.

In the new Business Intelligence Development Studio application, create a new project by using the File menu, selecting New, and then selecting Project.

Business Intelligence Development Studio will display a New Project dialog box that shows the different kinds of projects that Visual Studio 2008 can create. Ensure that Business Intelligence Projects is selected as the Project Type on the left side and that Import Analysis Services 2008 Database is selected as the Template on the right side. Give the project to be created a new name and location using the Name and Location options, and then click OK. Figure 11-4 shows the New Project dialog box.



**Figure 11-4:**  New Project dialog box

To create the new project, Business Intelligence Development Studio starts the Import Analysis Services Database Wizard. This wizard handles the import process after the database is selected.

After a given database is imported into a Business Intelligence Development Studio project, close the project by using the File menu and selecting Close Project. Then repeat the process for each upgraded database to ensure a development project is available for any changes an upgraded project might require.

#### Resolve Upgrade Issues

As previously noted, certain features that are available in SSAS 2000 are not upgraded, and other features behave differently in SSAS 2008. For example, drillthrough report settings in a given cube or partition will not be upgraded. Therefore, it is important to thoroughly review each Business Intelligence Development Studio project that is created for upgraded databases to determine whether any features in the original database must be recreated or modified. If any changes are made in a Business Intelligence Development Studio project, the database should be redeployed and reprocessed (by using the Build menu, and selecting Deploy) to make the changes available to users.

### Side-by-Side Upgrade

Frequently, customers will not want to immediately upgrade all databases on a given server to SSAS 2008 in a single upgrade action and completely overwrite the SSAS 2000 instance before the SSAS 2008 installation is tested and ready for production. In these cases, a side-by-side upgrade process can be used to upgrade databases in a more methodical and orderly manner. Side-by-side upgrades can be done on a single server (the existing SSAS 2000 server) or by using two servers (to take advantage of new hardware, for example).

* When the upgrade is performed on a single server, a new instance of SSAS 2008 is installed alongside the existing SSAS 2000 instance and the SSAS 2000 databases are upgraded to the new instance as needed.
* If the upgrade is performed by using two servers, SSAS 2008 is installed on the new server (as the default instance or as a named instance), and the same upgrade of databases is then performed.

Whether upgrading databases using a single server or two servers, the side-by-side upgrade process enables the databases to remain available in SSAS 2000 while they are tested (and possibly updated to resolve issues) in SSAS 2008. When all databases are upgraded, SSAS 2000 can be uninstalled. Optionally, when you use a single server for the upgrade, the new instance of SSAS 2008 can then be renamed as the default instance so that users can connect to it by using only the server name (as is done with SSAS 2000).

#### Preparing for a Side-by-Side Upgrade

A side-by-side upgrade to SSAS 2008 should not adversely affect an existing installation of SSAS 2000, even if a database upgrade effort fails. However, even with a side-by-side upgrade on a single server, we recommend that you take the same preparation steps as for an in-place upgrade, as follows:

1. Back up the registry information related to SSAS 2000. Using Registry Editor, export the following registry key to a file:

My Computer\HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\OLAP Server

1. Back up all existing databases using the archive command in Analysis Manager. Open Analysis Manager, right-click each database listed, and select Archive Database. Provide a unique file name for each .cab file that is created.
2. Back up the SSAS 2000 repository. This is either an Access database named msmdrep.mdb and located in %Program Files%\OLAP Services\Bin\ or a SQL Server database.

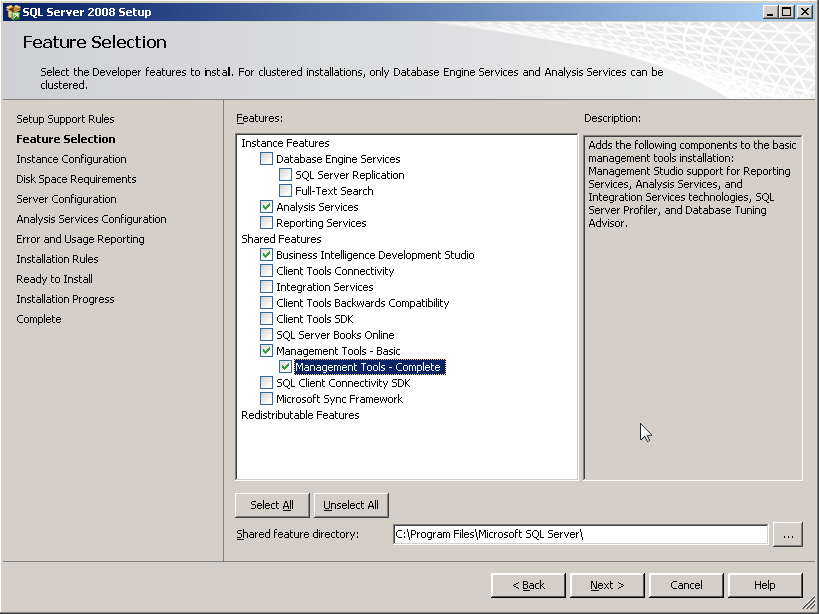
We recommend that you put all files that are generated by these steps in a single directory on a network share for safe-keeping during the upgrade process.

#### Installing the New Instance

After the planning process is complete, the first step in a side-by-side upgrade is to install a new instance of SSAS 2008. This new instance will be created by using the Setup application for SQL Server 2008.

To start the upgrade process, start the Setup application for SQL Server 2008. After it starts, the Setup application will install a set of prerequisites to the installation of SQL Server 2008 components. From the SQL Server Installation Center, select *New SQL Server stand-alone installation or add features to an existing installation*. The Setup program will run a system configuration check, collect system information, and prompt for a product key.

After it installs setup support files, the Setup application will provide options for selecting components to install. Select the Analysis Services option. If any of the SQL Server 2008 workstation components will be needed on the server, include the correct mix of Management Tools – Basic, Management Tools – Complete, and Business Intelligence Development Studio options when you select components. Figure 11-5 shows what the Features Selection screen looks like.

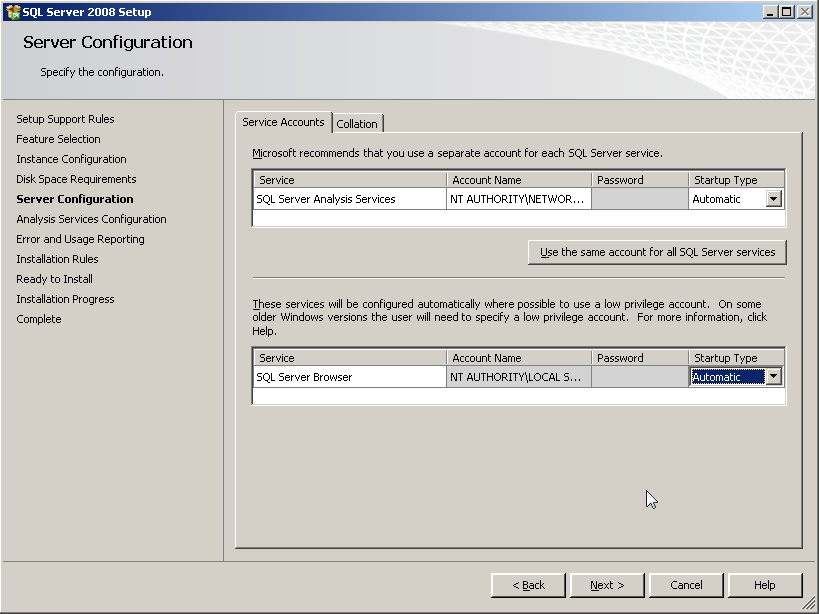


**Figure 11-5:** The Feature Selection screen for a single-server side-by-side upgrade

The Setup application will then prompt for an instance name for the new SQL Server 2008 components:

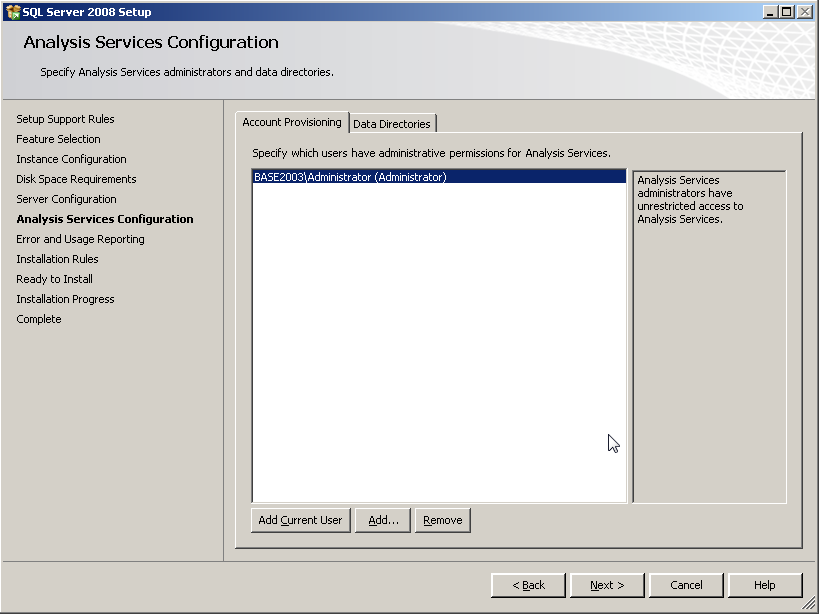
* For a single-server side-by-side upgrade, enter a new instance name for SSAS 2008 and continue. By default, the Setup application selects Named Instance, so a new instance name must be entered in order to continue.
* For a two-server side-by-side upgrade, you can install SSAS 2008 as the default instance or as a named instance.

After checking for disk space, the Setup application will prompt for service account information for the new instance of SSAS and for the new SQL Server Browser service. As mentioned earlier in the “In-Place Upgrade” section, the correct settings should be based on the security policies that are defined for standard Windows services in your organization; if no standard security policies are defined, the service account credentials should likely be set using either the Local Service built-in account or a domain user account. Figure 11-6 shows the screen you use during Setup to specify the service account credentials for the two new services.



**Figure 11-6:** Service account settings for new services

The Setup application will now ask for Analysis Services Configuration information. This screen has two tabs. The first tab asks for account provisioning, which lets users be added as administrators to SSAS. Be careful adding users: any user who you add as an SSAS administrator will have full access to all objects and data, and no security will be applied. Figure 11-7 shows this screen with the local server administrator added as an SSAS administrator.



**Figure 11-7:**  The Analysis Services Configuration screen lets you add SSAS administrators.

The second tab shown in Figure 11-7, Data Directories, lets you change the physical location of the data directories. The next screen gives you the option to turn on error reporting and usage reporting. Finally, installation rules are checked, a summary status screen is shown, and installation starts. A new instance of SSAS 2008, ready for the upgrade, will be installed and then available for use.

#### Moving Databases to the New Instance

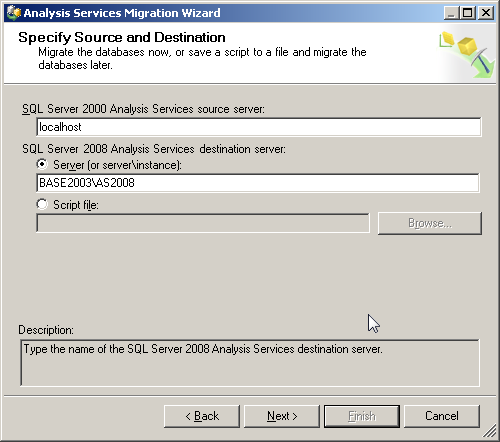
After a new instance of SSAS 2008 is installed, you can move one or more databases from SSAS 2000 to the new instance by using the Analysis Services Migration Wizard, included with SSAS 2008.

**Note:** The Migration Wizard tool will not run unless the steps are followed as outlined in the “Breaking Changes” section of this chapter.

You can start the Migration Wizard in one of two ways:

* Open SQL Server Management Studio and connect to the new instance of SSAS 2008; after you are connected, right-click the instance name in the Object Explorer window, and select Migrate Database.
* Run the MigrationWizard.exe executable.

When the Migration Wizard is started, it requests the name of a source server (running SSAS 2000) and a destination server (using the server\instance format for named instances of SSAS 2008). To move databases in a side-by-side upgrade scenario, enter the name of the source server and the newly installed SSAS 2008 instance as the destination. The wizard will then display a list of databases found on the source server. One or more of the databases listed can be selected for migration. Figure 11-8 shows the source and destination for the Migration Wizard being specified.



**Figure 11-8:** Source and destination for the Analysis Services Migration Wizard

After you have selected one or more databases, the Migration Wizard will validate the structure of each database. The wizard will display the results of the validation effort and provide a log of the results that can be viewed or saved before the migration process is started. After validating each database, the Migration Wizard will then migrate the structure and metadata for each database to the SSAS 2008 instance.

You can run the Migration Wizard as many times as needed to migrate one or more databases at different times. Therefore, a methodical migrate-test-deploy cycle can be used to move databases (and users) to the new instance of SSAS 2008.

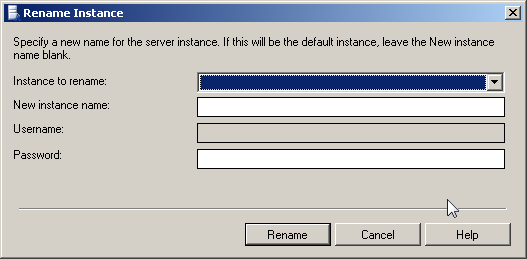
#### Post-Upgrade Tasks

After one or more SSAS 2000 databases have moved to the new instance of SSAS 2008, you should complete the same set of post-installation tasks as described previously in the “Post-Upgrade Tasks” topic of the “In-Place Upgrade” section.

#### Removing SSAS 2000

If all the databases on a given server are moved to a new instance of SSAS 2008, SSAS 2000 might not be needed any longer on the server. In this case, SSAS 2000 can be uninstalled by using Add or Remove Programs. After the earlier version is removed, the new instance of SSAS 2008 can be renamed so that it is recognized as the default instance on the server. You can do this by using the Instance Rename tool that is available for SSAS 2008. You can start the Instance Rename tool by executing ASInstanceRename.exe from the %Program Files%\Microsoft SQL Server\90\Tools\Binn\VSShell\Common7\IDE directory.

When the Instance Rename tool is started, it displays a single Rename Instance dialog box for selecting and renaming a given instance of SSAS 2008. Use the *Instance to rename* drop-down list to select the named instance. If the instance should act as the default instance for the server, leave the *New instance name* text box blank and then click Rename. Figure 11-9 shows the Instance Rename tool.



**Figure 11-9:** Rename Instance dialog box

The Instance Rename tool will update the SSAS 2008 instance and restart its service to complete the process. If an instance is renamed as the default instance, client tools can connect to SSAS 2008 by using the server name without any instance name.

### Redesigning Databases for SSAS 2008

Although upgrading to SSAS 2008 and migrating SSAS 2000 databases to SSAS 2008 provide great options for quickly moving databases to the new platform, many customers might decide to redesign databases to take advantage of specific new features and architecture changes included in the new version. Although developing new databases for SSAS 2008 will not be discussed in detail in this guide, some general ideas and thoughts related to redesigning the Foodmart 2000 database are included here as an example of the available possibilities.

#### Data Source Views

Every SSAS 2008 solution is based on a DSV that encapsulates schema metadata based on the data sources included in the Business Intelligence Development Studio project. When a database is upgraded, the DSV that is generated for the database might include generically named Named Calculations, such as Column1, Column2, and so on. For example, the DSV for an upgraded Foodmart 2000 database contains a series of Named Calculations that are generically named and then used for different dimension and cube designs.

Although this works fine and does not affect users (because client applications do not directly access a DSV), it is not intuitive from a development and support perspective. A better DSV design would include correctly named Named Calculations and could include other DSV options such as Named Queries and Diagrams.

#### Dimensions

SSAS 2008 includes many significant changes to support new features and functionality related to dimensions. Attributes and their relationships (versus levels in a dimension’s hierarchy) are now the primary design element in a dimension. Hierarchies in a dimension define navigation paths through a series of attributes, such as country, state/province, and city. A new feature of dimensions called *attribute hierarchies* exposes individual attributes to query applications. This lets users select any combination of attributes for reporting and analysis, beyond the hierarchies that may be defined as part of the dimension.

When a database is upgraded to SSAS 2008, the resulting dimension designs contain only those dimension attributes that are required to reproduce the hierarchies that are defined in the dimension. In addition, no attribute hierarchies are made visible even for the attributes included in the design of the dimension design. Also, in some cases, dimension attributes are renamed based on rules that are used during an upgrade (typically in an attempt to create attribute names that do not duplicate other objects such as dimensions or hierarchies).

#### Cubes

As with dimensions, SSAS 2008 contains significant changes to the features and functionality related to cubes. The most significant change when you compare the new version to SSAS 2000 is how cubes use measure groups and perspectives. In SSAS 2000, a cube could contain only a single fact table and its related dimensions. If measures from two or more fact tables needed to be combined for reporting or analysis purposes, a virtual cube was used to join two or more cubes based on one or more common dimensions.

In SSAS 2008, a cube can contain measures from multiple fact tables. Measures from a given fact table are grouped together using measure groups, and the relationships between dimensions in the database and each measure group are defined as part of the cube definition. Perspectives are then used to provide subsets of a cube’s dimensions, measure groups, measures, and calculations.

When a database is upgraded to SSAS 2008, the database will contain a separate cube for each basic and virtual cube found in the original database. Given the new features and architecture available in SSAS 2008, separate cubes can frequently be redesigned as a single cube with multiple measure groups and perspectives. Using this design provides better flexibility and performance. For example, the Foodmart 2000 database contains six separate cubes as defined in SSAS 2000. Each of these becomes a separate cube in the upgraded version of the database in SSAS 2008.

In addition to the flexibility improvements provided by new cube designs, SSAS 2008 also provides new features such as Key Performance Indicator (KPI) definitions, SQL Server Reporting Services (SSRS) actions, proactive caching storage mechanisms, and translation definitions. A redesigned Foodmart 2000 cube could take advantage of each of these new features, none of which are used by default in an upgraded database.

As you can see, there are potentially many reasons why a redesigned database might be better than an upgraded database. As stated before, although an upgrade process might provide a quick mechanism for moving to SSAS 2008, redesigning each database to take advantage of the new platform’s architecture, features, and design paradigms could prove helpful in the end.

## Upgrading from SQL Server 2005

Upgrading from SSAS 2005 is much easier, and carries less risk, than an upgrade from SSAS 2000. Whereas one of the goals of the SSAS 2008 team was to avoid any breaking changes for upgrading from SSAS 2005 to SSAS 2008, check the list of changes in this chapter to ensure that no changes will adversely affect an upgraded database. If your SSAS 2005 databases are using some of those features, they might require design changes after the upgrade to provide the same user experience.

Before you try an upgrade, review the behavior changes and determine whether any of the listed issues will affect the query results after an upgrade. In addition, run Upgrade Advisor to analyze the SSAS instance, and then review the generated report to verify that you have addressed all issues that must be resolved before the upgrade and that you understand the upgrade issues that you must resolve after Setup is complete. Also take advantage of such tools as the Upgrade Assistant and Best Practices Analyzer for SQL Server 2005 to aide in upgrade preparation. For more information, see “Upgrade Planning and Deployment” in Chapter 1.

Before you start an in-place upgrade, take steps to ensure that a failed upgrade can be rolled back. Although the in-place upgrade process should handle most situations, unforeseen problems might occur and result in a failed upgrade.

**Important**: With an in-place upgrade, the upgrade process handles all aspects of the upgrade, automatically upgrading the metadata for each database found in SSAS 2005.

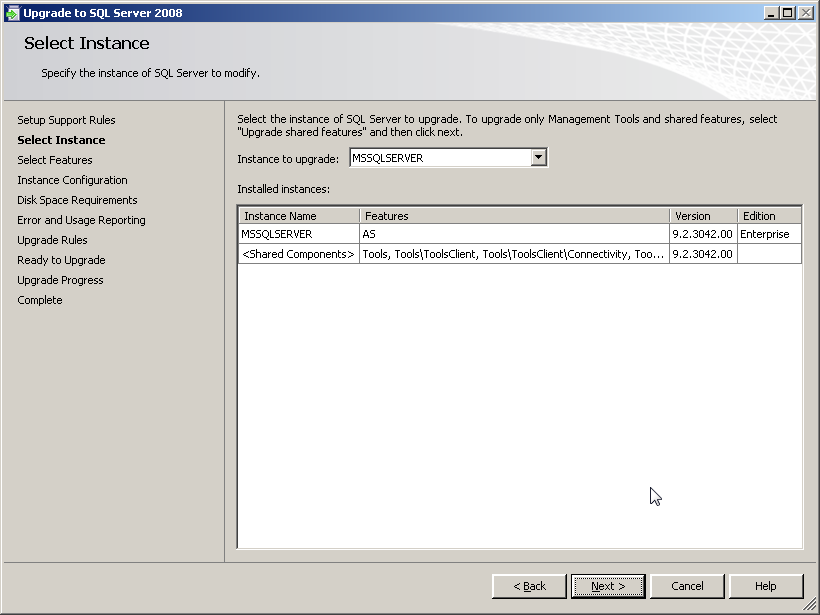
If a failed upgrade occurs, frequently the easiest resolution is to reinstall SSAS 2005 and restore the installation to its state before the upgrade process was started. Back up all databases by using the Back Up command in SQL Server Management Studio. To do this, open SQL Server Management Studio, right-click each database that is listed, and select Back Up. Provide a unique file name for each .abf file that is created, optionally choosing to also encrypt them with a password.

We recommend that you put all the files that are generated by the previous steps in a single directory on a network share for safe-keeping during the upgrade process.

### In-Place Upgrade

To start the in-place upgrade, start the Setup application for SQL Server 2008, selecting Installation and then *Upgrade from SQL Server 2000 or SQL Server 2005***.** The Setup program will run a system configuration check, collect system information, and prompt for a product key. After it installs any prerequisites, the Setup application will then prompt for the kind of installation to be performed.

After you decide to upgrade, the Setup program asks for a product key, prompts for the acceptance of the license file, and then installs setup support files. After the setup support files are finished, the setup support rules are checked. The next screen, which Figure 11-10 shows, asks for the instance to be upgraded.



**Figure 11-10:**  Select the instance of SSAS 2005 to upgrade

The next screen shows the features to be installed. In some cases, all the components are selected and no changes can be made. This is because the server was set up only with SSAS 2005 and the workstation components. If it is impossible to select different options, you will have to run Setup again to add features. For example, if Business Intelligence Development Studio was not already installed on the server but is needed in the future, a second run of the Setup application would be required.

After you have selected the components to install, the Setup application will prompt for an instance name for the newly installed SQL Server 2008 components. To upgrade an existing installation of SSAS 2005, leave the InstanceID the same. The Setup application should detect any running services based on which components were selected for installation. Ensure that the existing installation of SSAS 2005 is selected, and continue.

After you select the instance name, Setup will check for the necessary disk space. The Setup application will then prompt the user to turn on Error and Usage Reporting. These are optional settings and are off by default. Enabling them means the server might try to send data to Microsoft on an as-needed basis.

The Setup application next checks a series of upgrade rules. You should examine any failures in the rules and address them as necessary. Figure 11-11 shows the results of these rules, and because there are no failures, installation can continue.



**Figure 11-11:**  Upgrade rules are checked to ensure installation can continue.

When the Setup application is ready to continue with the upgrade, it will display a summary of actions that will be taken. Ensure that the action to be taken is “upgrade,” and then continue.

Clicking the Upgrade button starts the upgrade process. During the upgrade, an Upgrade Progress status screen will show status information about the various steps taken by the Setup application. When Setup is complete, a screen that displays the upgrade steps together with a success or failure message will be shown. A final screen will provide a summary of the installation together with any notes that are relevant to the upgrade process.

After the upgrade process has finished, you should perform a short set of post-installation tasks to ensure that the upgrade completed successfully. See the “Post-Upgrade Tasks” section in this chapter for information about these tasks.

#### Troubleshooting a Failed Upgrade

Should the in-place upgrade process fail, the best strategy is to review the setup logs that were created by the Setup application.

* Review the Summary.txt file that is located in the %Program Files%\Microsoft SQL Server\100\Setup Bootstrap\Log directory. If any error messages are listed, take whatever actions are required to correct the situation, and try the upgrade process again.
* If no error messages are included in the summary, review the Summary\_*[ComputerName]*\_*[date]\_[time]*.log file in the %Program Files%\Microsoft SQL Server\100\Setup Bootstrap\Log\*[date]\_[time]* directory. When you review the file, search for any instances of “Failed” for a Status (which indicates a setup error). If any error messages are listed, take whatever actions are required to correct the situation, and try the upgrade process again.

#### Post-Upgrade Tasks

After the upgrade of an SSAS 2005 server to SSAS 2008 is complete, you must complete a series of post-installation tasks before the upgraded databases will be available to users. In addition, you should perform other post-installation tasks to ensure that each database is working correctly and can be modified in the future if it is necessary.

**Review upgraded databases.** Each database that was upgraded by the SQL Server 2008 Setup application should be reviewed to ensure that the upgrade process completed successfully. Using SQL Server Management Studio, connect to SSAS on the upgraded server. If the workstation components were installed as part of the upgrade, SQL Server Management Studio should be available on the upgraded server; otherwise, SQL Server Management Studio will have to be started on another server or workstation that has had the workstation components for SQL Server 2008 installed.

After a connection to SSAS on the upgraded server is established, expand the Databases folder in the Object Explorer window. If the Object Explorer window is not visible, open the View menu and select Object Explorer. The Auto Hide button, represented by a pushpin in the upper-right corner of the Object Explorer window, can be used to “pin” the window so that it stays open.

Unlike an upgrade from SSAS 2000, the cubes that are upgraded from SSAS 2005 do not have to be processed to be browsed. Also, projects from Business Intelligence Development Studio 2005 can be opened in Business Intelligence Development Studio 2008 without modifications, so we strongly recommend that you back up the projects of Business Intelligence Development Studio 2005 if they have to be opened in Business Intelligence Development Studio 2005. Projects that were created in Business Intelligence Development Studio 2005 and opened and saved in Business Intelligence Development Studio 2008 cannot be opened in Business Intelligence Development Studio 2005.

## Performing Post-Upgrade Tasks

After you have upgraded one or more databases to the new instance of SSAS 2008, you have to complete a last set of post-upgrade tasks. (If you upgraded from SSAS 2000, you should follow these steps after those listed in the “Post-Upgrade Tasks” topic of the “In-Place Upgrade” section for “Upgrading from SSAS 2000.”) These tasks include the following:

1. Review each updated database by using SQL Server Management Studio to ensure that its contents (specifically, dimensions and cubes) are consistent with its SSAS 2000 or SSAS 2005 counterpart.
2. Process each upgraded database by using the Process command in SQL Server Management Studio.
3. Browse each upgraded database’s dimensions and cubes to ensure a consistent query experience compared to the database’s SSAS 2000 or SSAS 2005 counterpart.
4. Generate a new development project by using Business Intelligence Development Studio for each migrated database.
5. Resolve any migration issues in a given database, updating its dimension and cube designs as needed.
6. Review and possibly combine any upgraded data mining models that are included in each database.
7. Review the details related to each of these tasks in the previous sections to ensure a smooth and complete upgrade process.

## Conclusion

Upgrading to SSAS 2008 provides a wealth of new capabilities and features. Moving from SSAS 2005 provides performance and scalability improvements, together with a better set of developer tools for creating and managing SSAS databases. Although there are some functionality changes, a full redesign is not necessary.

SSAS 2000 databases can be upgraded to SSAS 2008, but the two products are very different. As you have seen in this chapter, there are several breaking changes between SSAS 2000 and SSAS 2008. Therefore, although an upgrade usually works, a partial or full redesign is frequently the best path to ensuring optimal performance and usability.

Upgrading to SSAS 2008 can be accomplished by using either an in-place upgrade or a side-by-side upgrade. The in-place upgrade is a bit more risky because it replaces the earlier version of SSAS. Before you do an in-place upgrade, it is important to make backups of all the SSAS databases. The side-by-side upgrade lets two versions of SSAS run at the same time, with SSAS 2008 being a named instance. When the earlier version of SSAS is removed, the SSAS 2008 version can be changed to the default instance on the server.

Customers upgrading from SSAS 2000 will discover that SSAS 2008 includes a brand new design paradigm that uses Business Intelligence Development Studio. They will also discover a new way to think about cube design that uses attribute hierarchies, multiple fact tables per cube, MDX scripts, KPIs, and much more. The additional features that are provided by SSAS 2008 over SSAS 2000 can seem intimidating, but these new features open up new options for analyzing and mining data.

Organizations upgrading from SSAS 2005 will find a smooth transition. There are improved tools for creating attribute relationships and aggregations, together with improved wizards for creating dimensions and cubes. The engine also contains some performance and scalability improvements. The good news is that the number of breaking changes is very small, and the overall design of cubes, dimensions, and the like has not changed.

## Additional References

For an up-to-date collection of additional references for upgrading to SSAS 2008, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkId=132510).

For the latest updates to SQL Server 2008 Books Online, see the [Microsoft SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132511).

[SQL Server Analysis Services Web site](http://go.microsoft.com/fwlink/?LinkID=85846&clcid=0x409)

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkId=132512)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132514)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkId=132515)

# Data Mining

## Introduction

Data mining is one of the most powerful analytical tools in the SQL Server Business Intelligence (BI) suite. Data mining was introduced as part of SQL Server 2000 Analysis Services (SSAS), the database platform’s OLAP and BI component. Although it was the first foray for SQL Server into advanced data mining analysis, SSAS 2000 supported two of the most popular algorithms: Decision Trees and Clustering. In SQL Server 2005, Microsoft completely rewrote the BI suite. With the debut of the Unified Dimensional Model (UDM) for OLAP, data mining in SSAS 2005 entered the enterprise-level analytical market. In SQL Server 2005, data mining is a mature product, featuring all the popular algorithms. One of the most important advantages of data mining with SSAS 2005 is ease of use and integration with other parts of the BI suite and business applications. With the introduction of the Microsoft Office 2007 Data Mining Add-Ins, the data mining functionality in SQL Server reached from developers, database professionals, and advanced business analysts to end-users.

The data mining success story continues in SSAS 2008. Using the foundation that SSAS 2005 laid, Microsoft has improved data mining in SSAS 2008, adding new features and improving existing functionality. However, there are behavioral and even breaking changes that you have to consider before upgrading SQL Server 2000 or SQL Server 2005 to SQL Server 2008. In addition to covering those changes, this chapter discusses the key steps that you must take to prepare for and perform a successful upgrade as well as important post-upgrade tasks. We have also collected references to the most important data mining upgrade resources, including the following:

* For more information about data mining functionality in SSAS 2008, see [Analysis Services—Data Mining](http://go.microsoft.com/fwlink/?LinkId=132525) in SQL Server 2008 Books Online.
* For additional SQL Server data mining information, see the [SQL Server Data Mining community site](http://go.microsoft.com/fwlink/?LinkId=132527), maintained by the Microsoft SQL Server Data Mining team.
* For other useful data mining content, see the MSDN blogs by [Jamie MacLennan](http://go.microsoft.com/fwlink/?LinkID=81816&clcid=0x409) and [Bogdan Crivat](http://go.microsoft.com/fwlink/?LinkId=132531).

## Data Mining Features

Before you start to upgrade, make it a priority to review the data mining features supported by the different editions of each version of SQL Server. Two tables in this section show this feature information in condensed format. Note the following abbreviations for editions for all three versions:

* EE = Enterprise, available in SQL Server 2000, SQL Server 2005, and SQL Server 2008
* SE = Standard, available in SQL Server 2000, SQL Server 2005, and SQL Server 2008
* PE = Personal, available in SQL Server 2000
* MSDE = Microsoft Desktop Engine, available in SQL Server 2000 and replaced with SQL Server Express (SSE) and SSE with Advanced Services in SQL Server 2005 and SQL Server 2008
* WG = Workgroup, available in SQL Server 2005 and SQL Server 2008
* WE = Web, available in SQL Server 2008
* SSE = SQL Server Express, available in SQL Server 2005 and SQL Server 2008
* SSEA = SQL Server Express with Advanced Services, available in SQL Server 2005 and SQL Server 2008

In addition to the editions that were mentioned previously, Microsoft also offers the Developer and Enterprise Evaluation editions. They have the same functionality as Enterprise but the licensing is different. Table 12-1 shows which SQL Server 2000 editions support data mining features.

Table 12-1: Data Mining Support in Different SQL Server 2000 Editions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature/Edition** | **EE** | **SE** | **PE** | **MSDE** |
| Data Mining | Yes | Yes | Yes | No |

As you can see, in SQL Server 2000, data mining support is all or nothing: If it is supported by an edition, it is supported completely. For complete details about feature support in the editions of SQL Server 2000, see [Features Supported by the Editions of SQL Server 2000](http://go.microsoft.com/fwlink/?LinkId=132532) in SQL Server 2000 Books Online.

Data mining features are supported on a more fine-grained level in SQL Server 2005 and SQL Server 2008, as Table 12-2 shows (cells with a light gray background show editions and features that are available in SQL Server 2008 only).

Table 12-2: Data Mining Features in SQL Server 2005 and SQL Server 2008 Editions

| **Feature** | **EE** | **SE** | **WG** | **WE** | **SSE** | **SSEA** |
| --- | --- | --- | --- | --- | --- | --- |
| Standard data mining algorithms | Yes | Yes | No | No | No | No |
| Data mining tools: wizards, editors, query builders | Yes | Yes | No | No | No | No |
| Algorithm viewers | Yes | Yes | No | No | No | No |
| Improved integrated OLAP and data mining functionality (MDX prediction function, DM dimensions) | Yes | Yes | No | No | No | No |
| Reporting integration with DM prediction queries | Yes | Yes | No | No | No | No |
| Parallelism for model processing | Yes | No | No | No | No | No |
| Parallelism for model prediction | Yes | No | No | No | No | No |
| Text-Mining Term Extraction transformation (SSIS) | Yes | No | No | No | No | No |
| Text-Mining Term Lookup transformation (SSIS) | Yes | No | No | No | No | No |
| Data Mining Query transformation (SSIS) | Yes | No | No | No | No | No |
| Data Mining processing destination (SSIS) | Yes | No | No | No | No | No |
| Algorithm plug-in API | Yes | No | No | No | No | No |
| Advanced configuration and tuning options for data mining algorithms | Yes | No | No | No | No | No |
| Unlimited concurrent data mining queries | Yes | No | No | No | No | No |
| Unlimited attributes for association rules | Yes | No | No | No | No | No |
| Multiple prediction targets for Naïve Bayes, Neural Network, and Logistic Regression | Yes | No | No | No | No | No |
| Cross validation | Yes | No | No | No | No | No |
| Models on filtered subsets of mining structure data | Yes | No | No | No | No | No |
| Time series: custom blending between ARTXP and ARIMA models | Yes | No | No | No | No | No |
| Time series: prediction with new data | Yes | No | No | No | No | No |
| Time series: cross-series prediction | Yes | No | No | No | No | No |
| Sequence prediction | Yes | No | No | No | No | No |

Standard in SQL Server 2005 and SQL Server 2008 supports all the features that are available in SQL Server 2000 plus many more. Enterprise adds even more valuable features. Because of the many improvements in data mining between SQL Server 2000 and SQL Server 2005 or SQL Server 2008, rebuilding models is probably the best strategy.

## Preparing to Upgrade

After you select the SQL Server 2008 edition that suits your needs, you should investigate which features are deprecated in SQL Server 2008. These features will not affect an upgrade, but you will have to update your models to stop using them before your next upgrade. You must also know what functionality cannot be upgraded because it is discontinued or because it has changed in SQL Server 2008. In addition, you should be aware of some behavioral changes in data mining between SQL Server 2000 and SQL Server 2005 and in SQL Server 2008; otherwise, you could get unexpected results. Let’s look at each of these categories of changes. This section also notes potential issues with data mining models. For complete coverage of SSAS data mining changes in SQL Server 2008, see [SQL Server Analysis Services Backward Compatibility](http://go.microsoft.com/fwlink/?LinkID=132476&clcid=0x409) in SQL Server 2008 Books Online.

### Deprecated Features

SSAS 2000 supports the XML markup language called Predictive Model Markup Language (PMML) 1.0. PMML is a standard language to describe data mining models. However, the language is incomplete from the standards point of view, although some specific extensions were added. In contrast, SSAS 2008 supports standard PMML and deprecates SSAS 2000 PMML extensions, which means that you should not use them. This is probably not a big issue because you use PMML directly only if you export your SSAS 2000 mining models to PMML. You can create a mining model in SSAS 2008 from PMML and store it in an SSAS 2008 database. If you export it from SSAS 2008, standard PMML will be generated. Some of the most important SQL Server 2000 extensions to PMML 1.0 include the following:

* Support for nested tables.
* The Discretized, Ordered, and Cyclical model variables in addition to the simple Categorical and Continuous model variables.
* Support for Key columns in nested tables.
* Support for Relation type columns as "hierarchy parents."
* All model variables can have a missing state described, even those with a continuous domain.

For the complete specification of SQL Server 2000 data mining functionality and extensions, see [OLE DB for Data Mining Specification 1.0](http://go.microsoft.com/fwlink/?LinkId=132534).

For a complete list of deprecated features in SSAS 2008, see [Deprecated Analysis Services Functionality in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132477) in SQL Server 2008 Books Online.

### Discontinued Functionality

There is only a short list of discontinued data mining functionality from SSAS 2005 to SSAS 2008:

* Mining Execution Location connection string property
* Mining Location connection string property

In SSAS 2008, the Analysis Services 2008 OLE DB provider does not support the Mining Execution Location and Mining Location properties. Although you can specify the Mining Execution Location property in a connection string, SSAS 2008 ignores the setting. To upgrade local mining models from SQL Server 2000 to SQL Server 2008, connect by using the Analysis Services 2005 OLE DB provider (MSOLAP.3) and set the Mining Location connection string property to the name of the folder that contains your local mining models. The local mining model service will read all the .dmm files in the directory and import them into a .cub file. All access to models in that folder will then be from this file and not the files that are created by SQL Server 2000 data mining. The original files will be left on your computer untouched.

In SSAS 2000, you can create local mining models by using the PivotTable service. However, starting with SSAS 2005, only server models are supported.

For a complete list of SSAS 2008 discontinued functionality, see [Discontinued Analysis Services Functionality in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132475) in SQL Server 2008 Books Online.

### Breaking Changes

If you upgrade your data mining models from SSAS 2005 to SSAS 2008, the following issues could prevent a successful upgrade, force you to update your SSAS databases after the upgrade, or change the results of your mining models:

* ODBC data sources are not supported in SSASS 2008. If you are using ODBC data sources, you must change them to OLE DB providers.
* Decision Support Objects (DSO) are not installed by default when you install SQL Server 2008. Although the backward-compatibility package is installed, the DSO component of the package is disabled. The SQL Server Analysis Services Migration Wizard relies on this component and will fail unless the component is installed.
* You can use Visual Basic for Applications (VBA) functions in your Data Mining Extensions (DMX) statements. However, VBA functions handle NULL values differently in SSAS 2008. In SSAS 2005, VBA functions return 0 or an empty string when NULL or empty values are used as arguments. In SQL Server 2008, VBA functions return NULL.

If you are upgrading from SSAS 2000 to SSAS 2008, you can experience the following additional issue:

* In SSAS 2000, the Decision Trees algorithm uses the MINIMUM\_LEAF\_CASES parameter, and Clustering uses the MINIMUM\_CLUSTER\_CASES parameter. In SSAS 2005, both parameters are renamed to MINIMUM\_SUPPORT. If you use these two parameters in an SSAS 2000 model, they will be upgraded, and SSAS 2008 will use them when it processes a model. However, they are not standard SSAS 2008 parameters, and you should remove them and change to the MINIMUM\_SUPPORT parameter as appropriate.

For information about some of these breaking changes, see [Breaking Changes to Analysis Services Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132482) in SQL Server 2008 Books Online.

### Behavior Changes

There are no specific behavior changes in mining models when you upgrade them from SSAS 2005 to SSAS 2008. And when you upgrade from SSAS 2000, you can expect better results, such as better predictions, because of algorithm improvements. In addition, you should have an improved browsing experience because as of SSAS 2005 because Microsoft added a new set of mining model browsers.

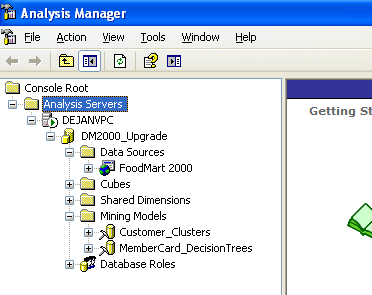
### Running Upgrade Advisor

Fortunately, you do not have to check all the potential upgrade issues manually. The SQL Server 2008 Upgrade Advisor can help you detect potential issues. However, be aware that Upgrade Advisor does not report all possible issues. It reports blocking issues only. You will soon see that some breaking changes are not reported.

“Upgrade Planning and Deployment” in Chapter 1 of this guide covers how to install and use Upgrade Advisor, so we will not repeat that information here. But to illustrate some upgrade issues related to data mining models, we created a simple SSAS database on both SSAS 2000 and SSAS 2005 that contains only a couple of mining models and other necessary objects, without OLAP dimensions and cubes. For our examples, we use the SSAS 2000 FoodMart sample database and the SSAS 2005 AdventureWorksDW sample database. Let’s look at the issues our demonstration found.

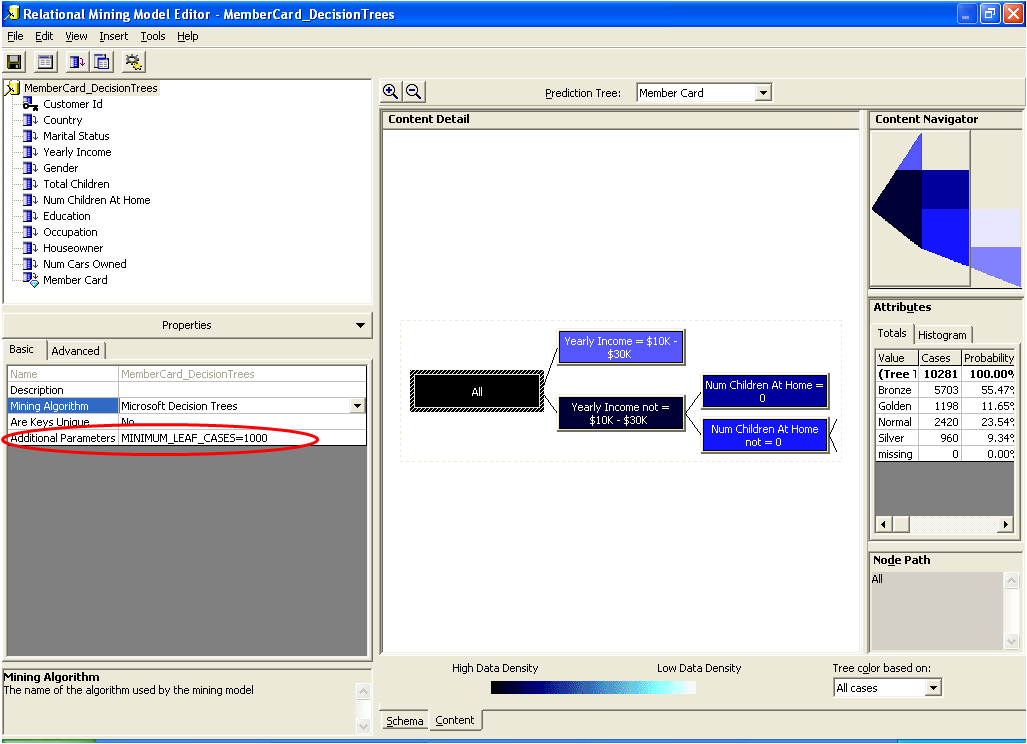
**Note:** If you are performing an in-place upgrade, remember that the Microsoft Access FoodMart sample database will be deleted during the upgrade process. Create a copy before upgrading.

First, let’s look at an example of running Upgrade Advisor against a sample SSAS 2000 database. The sample database contains two mining models that are based on the Customers table in the FoodMart database. One model uses the Decision Trees algorithm to predict member card type based on demographic data, and the other one uses the Clustering algorithm to find clusters of customers based on demographic and card membership data. The Analysis Manager screen in Figure 12-1 shows the data source object and two mining models.



**Figure 12-1:** SSAS 2000 database that contains two mining models

For the Decision Trees model, the MINIMUM\_LEAF\_CASES parameter is set to 1000. Because the model has only 10.281 cases, such a large value for this parameter leads to a shallow tree, with only a few nodes. Figure 12-2 shows the tree and highlighted parameter.



**Figure 12-2:** SSAS 2000 Decision Trees model

Now, let’s start Upgrade Advisor to get an analysis of possible upgrade issues. On the first screen, click the Launch Upgrade Advisor Analysis Wizard link. Select only the Analysis Services component of your SSAS 2000 instance, and run the wizard. After the wizard finishes its analysis, start the Report Viewer. As you can see in Figure 12-3, Upgrade Advisor has correctly detected that the instance uses the ODBC data source. However, it did not detect the MINIMUM\_LEAF\_CASES parameter, which has the same behavior in SSAS 2008 as it does in SSAS 2000 but is renamed in SSAS 2005 and SSAS 2008. Neither of these two issues will prevent the upgrade, but you have to resolve them after the upgrade.



**Figure 12-3:** Upgrade Advisor report on SSAS 2000 mining models

Chapter 1 also details two other upgrade tools that you might find valuable: SQL Server 2008 Upgrade Assistant and the Best Practices Analyzer, which comes in versions for SQL Server 2000 and SQL Server 2005.

## Upgrading from SQL Server 2000

In Chapter 1, you learned how to start the SQL Server 2008 Setup program and perform an in-place and side-by-side upgrade. This section assumes that you have already installed SQL Server 2008, so we will not describe that process here. Instead, we focus only on data mining issues that you might face in an upgrade from SSAS 2000 to SSAS 2008. The in-place upgrade is somewhat simpler, but the side-by-side upgrade strategy gives you more options for upgrading mining models.

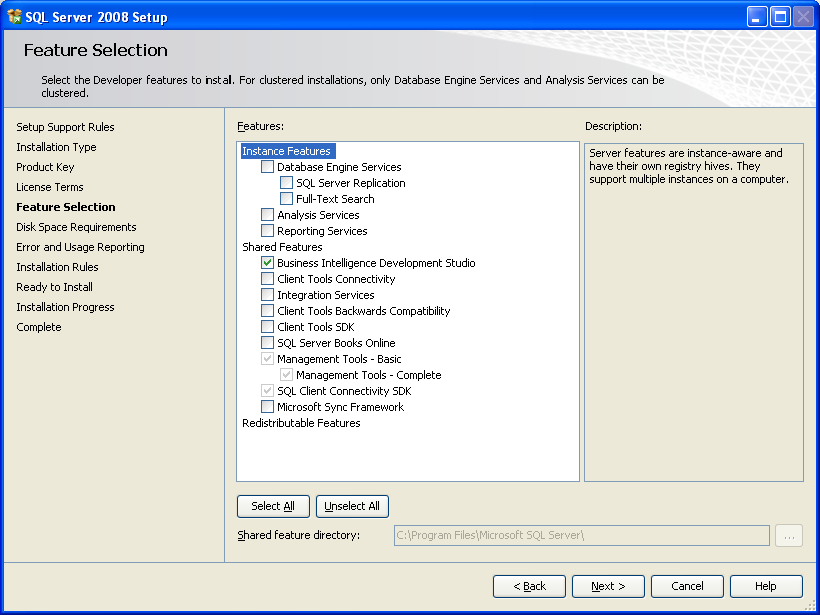
### In-Place Upgrade

After the upgrade, the following objects are created in the SSAS 2008 database, assuming you have mining models and data sources only in SSAS 2000:

* Data sources
* A data source view (DSV) for each mining model
* A mining structure for each mining model
* A mining model inside each structure

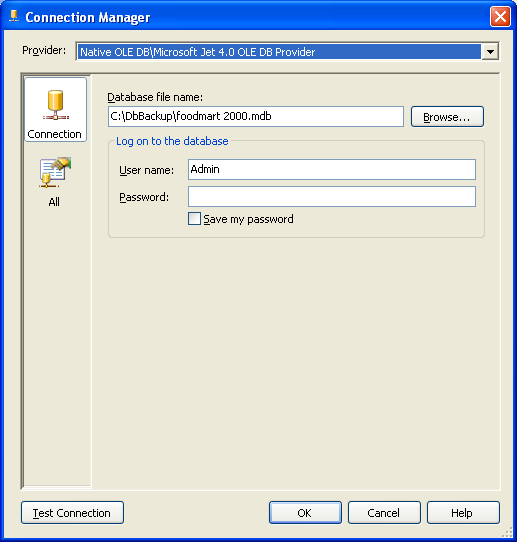
In SSAS 2008, multiple models can share the same structure. The mining structure defines which columns from the source are used for the whole set of models in the same structure. You can easily compare the performance of models that share the same structure.

But before dealing with model performance, you must resolve some other issues. First, in SQL Server 2000, Analysis Manager is both an administrative and development tool. If you are doing an in-place upgrade, by default the Setup program installs only the SQL Server 2008 Management and Development tools (for more information, see “Management and Development Tools” in Chapter 2). You can change the data source connection string property and process mining models in SQL Server Management Studio. However, to continue development, change existing objects, or add new objects to an upgraded SSAS database, you must use the Business Intelligence Development Studio development tool. If you want to continue development on the same computer where your SSAS server is installed, install Business Intelligence Development Studio there, or install it on a client machine. You can also do some development with SQL Server Management Studio by scripting the model, changing the XMLA script manually, and then executing it. However, it is much easier and more intuitive to develop data mining models in Business Intelligence Development Studio. To make sure Business Intelligence Development Studio is installed as part of your in-place upgrade, rerun SQL Server 2008 Setup, and from the Feature Selection screen, select Business Intelligence Development Studio, as Figure 12-4 shows.



**Figure 12-4:** Running Setup again to add Business Intelligence Development Studio as a feature for your in-place upgraded database

As noted, you do not need Business Intelligence Development Studio if you want to continue using existing models without changes. However, if you use ODBC data sources in SSAS 2000, you must change them to OLE DB providers and then process the models. In our example, we are using the ODBC driver in SSAS 2000 to connect to the FoodMart sample database. We copied the database (the .mdb file) to the backup folder. To change the ODBC data sources, in SQL Server Management Studio, right-click the data source that you want to change, and select Properties. Change the Connection String property, and change the provider to the appropriate OLE DB provider, pointing to the database that is the source for your mining models. Figure 12-5 shows the connection string settings for the FoodMart sample database. The provider is now the Microsoft Jet 4.0 OLE DB Provider, and the *Database file name* text box points to the backup of the FoodMart database we created before upgrading.



**Figure 12-5:** Modified connection string properties for the FoodMart demo database

### Side-by-Side Upgrade

Even if you decide to perform a side-by-side upgrade, it is the recommended best practice to first run Upgrade Advisor to check for potential issues before you install SQL Server 2008 and start to upgrade mining models. You can install SQL Server 2008 on the same computer where you are running SQL Server 2000 or on a different computer. SSAS 2008 supports named instances exactly as the Database Engine does. Assuming you have installed SSAS 2008, the management and development tools are already installed, and you can start moving your mining models from SSAS 2000 to SSAS 2008.

You can use the Analysis Services Migration Wizard to migrate SSAS 2000 databases, as the following steps describe:

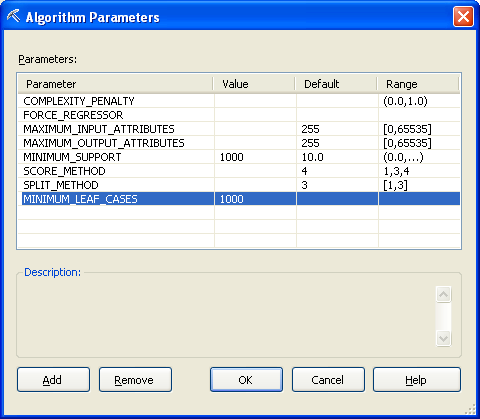
1. Start the Migration Wizard by right-clicking the instance of SSAS 2008 in Object Explorer in SQL Server Management Studio and selecting Migrate Database. Or, you can run the wizard at a command prompt by starting the MigrationWizard.exe application.
2. On the Welcome page, click Next.
3. On the Specify Source and Destination page, select the source (SSAS 2000) and destination (SSAS 2008) servers. Instead of creating a database on the destination server, you can create an XMLA script. You can then execute the script in SQL Server Management Studio or through the Analysis Services Execute DDL task in SQL Server Integration Services (SSIS).
4. On the Select Databases to Migrate page, select the databases that you want to move. You can also rename the destination database on this page.
5. On the Validating Databases page, you can check for any issues that could prevent a successful upgrade. Using the *View log* drop-down list, you can choose to display all messages or errors, warnings, or successes only. If there are no errors, click Next.
6. On the Migrating Databases page, you can follow the upgrade progress. When it is finished, click Next.
7. On the Completing the Wizard page, review the migration report. Click Finish to complete the wizard.

As with an in-place upgrade, you must reprocess moved databases. Of course, if you use ODBC drivers, you must change the data sources to use OLE DB providers. However, with a side-by-side installation, the FoodMart sample database is not deleted during the setup process. Consider using Business Intelligence Development Studio to create a project from your upgraded database. If you want to generate the project, use the Import Analysis Services 2008 Database template when you create a new project in Business Intelligence Development Studio.

### Post-Upgrade Tasks

As we noted earlier, you need to replace some SSAS 2000 mining model parameters with the appropriate SSAS 2008 parameters after an upgrade. As we saw in our earlier example, the SSAS 2000 Decision Trees model used a value of 1000 for the MINIMUM\_LEAF\_CASES parameter to create a shallower tree with fewer nodes. After the upgrade, SSAS 2008 still considers the parameter during processing, and you get the same tree you had in SSAS 2000. Nevertheless, we recommend that you use the standard MINIMUM\_SUPPORT SSAS 2008 parameter. If you used Business Intelligence Development Studio to create a project from the upgraded database, you can use the Data Mining Designer to change the parameters, as these steps describe:

1. Open the Data Mining Designer by double-clicking the mining structure that contains the models that you have to change.
2. On the Mining Models tab, right-click the model that you need to change and select Set Algorithm Parameters.
3. Change the MINIMUM\_SUPPORT parameter to the value of the SSAS 2000 MINIMUM\_LEAF\_CASES parameter, as Figure 12-6 shows, and remove the SSAS 2000 parameter. Click OK.



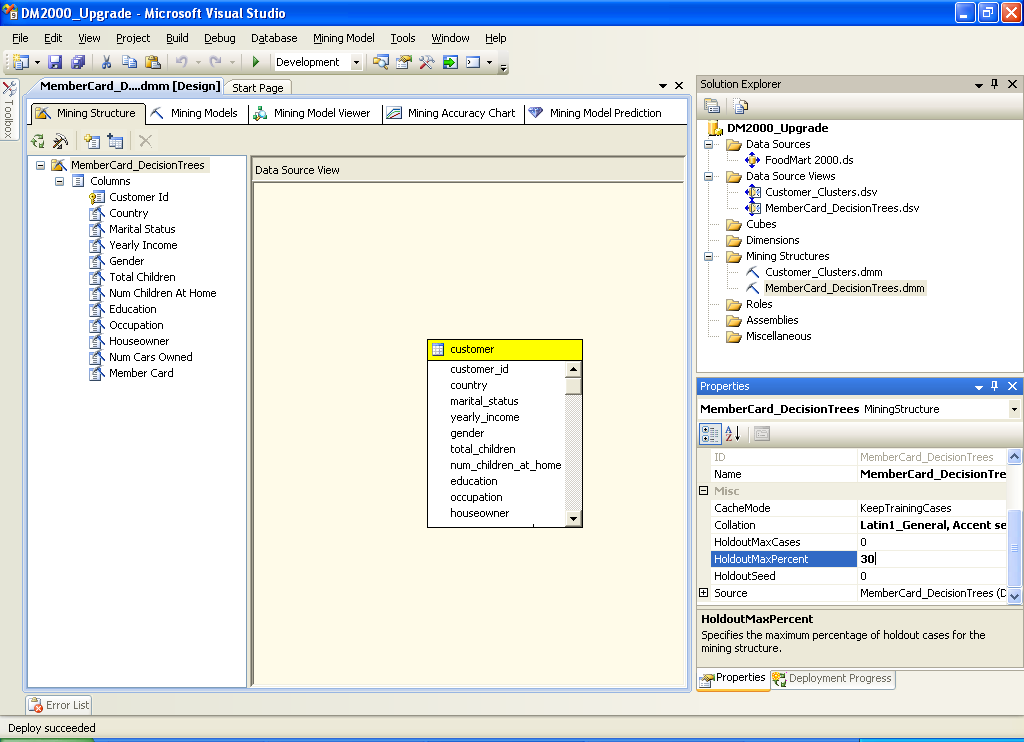
**Figure 12-6:** Replace the SSAS 2000 MINIMUM\_LEAF\_CASES parameter with the SSAS 2008 MINIMUM\_SUPPORT parameter

1. Save and deploy the project, and then view the model. You should get the same tree as in SSAS 2000.

You likely will want to modify more than only the parameters after the upgrade. For example, SSAS 2008 supports many more data mining algorithms than SSAS 2000. So you should consider building additional models on the same structure (that is, on the same data), using different algorithms with different parameters.

In SSAS 2008, you can easily compare the performance of the models, so you might decide to use a completely different model in production. For predictive models, such as Decision Trees, you should put approximately 30 percent of your data aside as a test set and use the other 70 percent of the data as a training set (that is, data that is used to process the models). Then you can try to perform predictions on the test set. Because you already know the outcome of the predicted variables in the test set, you can easily compare which model gives you the most accurate predictions. In SSAS 2008, you can use a Lift Chart, Profit Chart, and Classification Matrix to see the accuracy of the models.

It is important to randomly split your data into training and test sets. You do not want to build a new pattern in your data by using a non-random split. You can split your data by using SSIS Percentage Sampling and Row Sampling transformations. In addition, SSAS 2008 can split the data randomly for you while it processes the mining structure. To do this, set the HoldoutMaxPercent mining structure property to the percentage of data that is used for the test set. Use the Properties window in Business Intelligence Development Studio to specify the mining structure you want to set this property for. Figure 12-7 shows this property set to 30 for our sample MemberCard\_DecisionTrees structure.



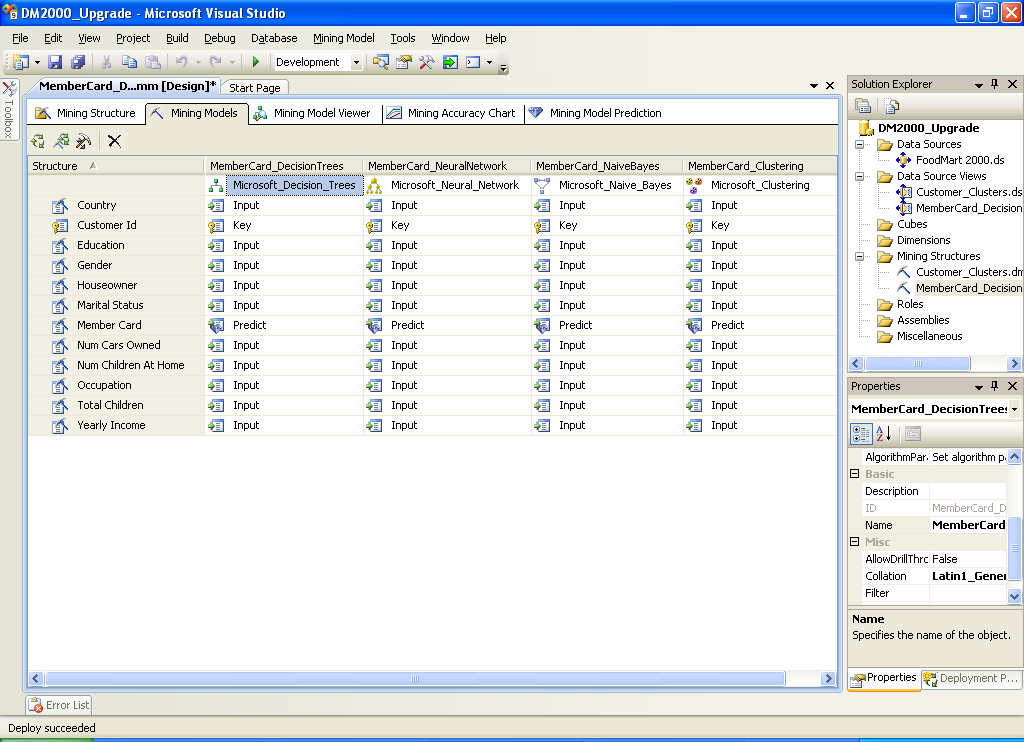
**Figure 12-7:** Setting the percentage of holdout data for the test set

Using the same structure as the two Decision Trees models in our example, let’s create three additional models to show how you would add models to your upgraded structure. One model uses the Neural Network algorithm, one model uses the Naïve Bayes algorithm, and one model uses the Clustering algorithm. In SSAS 2008, you can also use the Clustering algorithm for predictions.

To add a mining model that shares the same structure as an existing model, follow these steps:

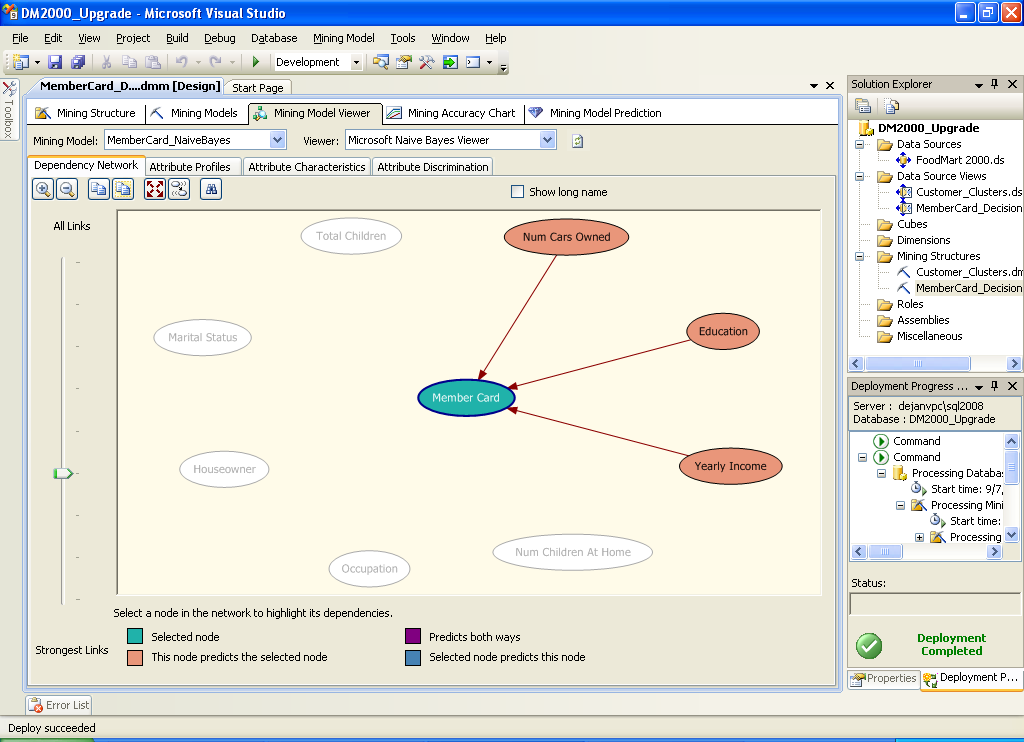
1. In Business Intelligence Development Studio, open the Data Mining Designer.
2. In the Mining Models tab, right-click an existing model and select New Mining Model.
3. Define the model name, and select the algorithm you want to use.
4. Refine the new model by setting the algorithm parameters.
5. Save the project, and then deploy and process it.

Figure 12-8 shows all four models—the upgraded one and the three added after the upgrade—built using the same structure.



**Figure 12-8:** Multiple models using different algorithms but sharing the same mining structure

After your models are deployed and processed, you can use the new data mining viewers to gain lots of additional information not available in SSAS 2000. For example, you can use the Dependency Network view in the Microsoft Tree Viewer and the Microsoft Naïve Bayes Viewer to quickly determine which attributes have the most influence on the predictive attribute. You can use the Attribute Discrimination view in the Microsoft Naïve Bayes Viewer and the Microsoft Neural Network Viewer to find which input attributes favor specific states of a predictable attribute. For more information about data mining viewers in SQL Server 2008, see [Viewing a Data Mining Model](http://go.microsoft.com/fwlink/?LinkId=132569) in SQL Server 2008 Books Online. Figure 12-9 shows the Dependency Network view of the Naïve Bayes Viewer, highlighting three attributes that have the highest influence on the Member Card attribute.



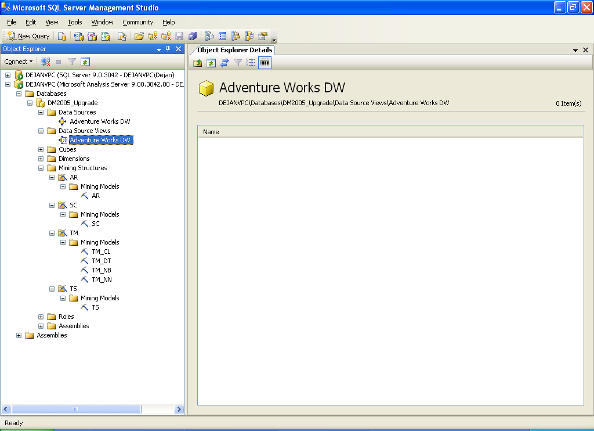
**Figure 12-9:** Dependency Network view showing the three attributes that have the highest influence on a predictable attribute

After you use the SSAS 2008 data mining views to review the original and new models, you might decide to deploy in production a different model than the one that you deployed in SSAS 2000. However, you should not make this decision yet. You should check the accuracy of the models first. Remember in the example, we used 30 percent of the data for a test set. In addition to accuracy, you should also verify the robustness of your models using cross-validation. You will learn more about checking accuracy and robustness in the next part of this chapter, which covers upgrading data mining models from SQL Server 2005 to SQL Server 2008.

## Upgrading from SQL Server 2005

As with an upgrade from SQL Server 2000, you also have two methods for upgrading your SQL Server 2005 data mining models to SQL Server 2008: in-place and side-by-side.

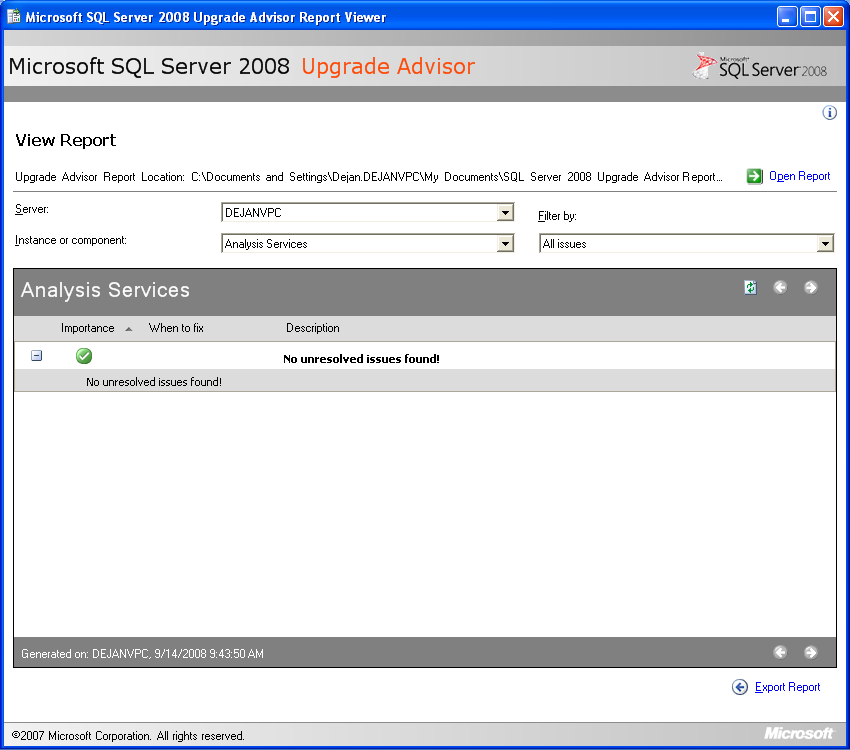
To demonstrate upgrading data mining models from SQL Server 2005 to SQL Server 2008, let’s consider a sample SSAS 2005 database that has a data source from the SQL Server 2005 AdventureWorksDW demo database, a DSV with all the necessary database views included (vTargetMail, vTimeSeries, vAssocSeqOrders, and vAssocSeqLineItems), and seven data mining models in four data mining structures. Four predictive models use the same structure, based on vTargetMail. The models try to predict whether a customer is likely to buy a bike or not based on demographic data and by using the following algorithms: Decision Trees, Naïve Bayes, Neural Network, and Clustering. The Time Series mining model has its own structure, based on the vTimeSeries view, for forecasting the sales quantity and amount of bike models in different regions. And the Association Rules algorithm model is based on vAssocSeqOrders and vAssocSeqLineItems database views, and tries to determine which products are sold together. Although the Sequence Clustering algorithm uses the same source database views, it has its own structure, with the keys defined differently than in the structure for the Association Rules model. Sequence Clustering tries to find not only which products are sold together, but also the order of products in a transaction. Figure 12-10 shows all the objects in the database to be upgraded.



**Figure 12-10:** SSAS 2005 database

### In-Place Upgrade

You start an in-place upgrade from SSAS 2005 to SSAS 2008 similarly to how you start an in-place upgrade from SSAS 2000. The first step is to run the Upgrade Advisor. As you can see in Figure 12-11, Upgrade Advisor found no issues with the SSAS 2005 database.



**Figure 12-11:** Upgrade Advisor found no issues with the SSAS 2005 database

When you upgrade from SSAS 2005 to SSAS 2008, Business Intelligence Development Studio is automatically installed. Therefore, you do not have to run Setup again to install it as you have to when you upgrade from SSAS 2000.

The upgrade process is also painless for the data mining models. Your SSAS databases are automatically upgraded, and you can continue to use your mining models just as you used them in SSAS 2005. In addition, you can use your SSAS 2005 mining projects in Business Intelligence Development Studio 2008 to continue with development. When you open the SSAS 2005 data mining project in Business Intelligence Development Studio 2008 for the first time, the Visual Studio Conversion Wizard starts automatically, and your project is converted to version 2008. You also get a backup of your project automatically. However, if you want more detailed control over versions, consider using a version control system to maintain earlier versions, or back up the project manually before you convert it to Business Intelligence Development Studio 2008. After you have converted a project to Business Intelligence Development Studio 2008, you cannot open it in Business Intelligence Development Studio 2005 any longer.

You also have to perform some important data mining tasks after your in-place upgrade from SSAS 2005 to SSAS 2008. There are many new features and improvements in SSAS 2008 data mining that can lead you to deploy a different predictive model in production, combine mining structures, or refine forecasting models. We discuss these important considerations in the “Post-Upgrade Tasks” section later in this chapter.

### Side-by-Side Upgrade

Although the Analysis Services Migration Wizard works only with upgrades from SQL Server 2000 to SQL Server 2008, you have plenty of options for moving your mining models from SSAS 2005 to SSAS 2008:

* You can back up the SSAS 2005 database and restore it on SSAS 2008.
* With SQL Server Management Studio, you can create an XMLA script for creating the complete database or any object in the database and then execute the script on SSAS 2008.
* You can open the SSAS 2005 project in Business Intelligence Development Studio 2008 and deploy it on SSAS 2008.
* You can reverse-engineer an SSAS 2005 database in Business Intelligence Development Studio 2008 to create a 2008 project and then deploy the project on SSAS 2008.

You can also quickly import SSAS 2005 data mining models to an SSAS 2008 database by using the EXPORT and IMPORT DMX commands.

“Analysis Services” in Chapter 11 covers the options for upgrading a complete SSAS database. So in this section, we focus on the data-mining-specific upgrade options.

With the EXPORT DMX command, you can export a complete mining structure, one or more mining models, or a model or a structure with dependencies. Exporting with dependencies means that all objects that are needed to process the structure, such as the data source and the DSV, are included in the backup (.abf) file. Here are some examples of EXPORT commands executed on our sample SSAS 2005 database:

-- Exporting complete structure

EXPORT MINING STRUCTURE [TM]

TO 'C:\DM2005\_Upgrade\TM\_Structure.abf';

-- Exporting a single model

EXPORT MINING MODEL [TM\_DT]

TO 'C:\DM2005\_Upgrade\TM\_Model.abf';

-- Exporting a model with dependencies

EXPORT MINING MODEL [AR]

TO 'C:\DM2005\_Upgrade\AR\_Model\_Dependencies.abf'

WITH DEPENDENCIES;

In SSAS 2008, you can use SQL Server Management Studio to create an empty database. You can then use the IMPORT DMX command, as follows, to import a mining model or a structure with all the objects required for processing as long as you exported the model or the structure with dependencies, as in the third EXPORT statement in the previous example.

-- Importing a model with dependencies

IMPORT

FROM 'C:\DM2005\_Upgrade\AR\_Model\_Dependencies.abf';

If you already have the destination SSAS 2008 database and you need to import only a mining structure, import it from the backup file that has the complete structure, as follows:

-- Importing complete structure

IMPORT

FROM 'C:\DM2005\_Upgrade\TM\_Structure.abf';

If you import from a file that only has the mining model, the associated structure is also created. Therefore, you cannot have a structure that has the same name in the destination SSAS database. The following command shows an example of importing a mining model:

-- Importing a single model

IMPORT

FROM 'C:\DM2005\_Upgrade\TM\_Model.abf';

This command imports from the file to which you exported the TM\_DT model. And as we noted, the TM structure cannot exist in the destination database because it is recreated there during the import. After the import, the TM structure contains only one model: TM\_DT.

Finally, for backward compatibility with SSAS 2000, the Decision Trees and Clustering algorithms support the PMML presentation of a model. You can access this presentation format by using the DMX SELECT FROM PMML command:

-- PMML presentation

SELECT \*

FROM TM\_CL.PMML;

In the results from this query, you find the PMML presentation in the MODEL\_PMML column. You can use the CREATE MINING MODEL DMX command, copying the XML string from the MODEL\_PMML column into this command, to create a mining model and structure in SSAS 2008 (for brevity, the actual PMML is shortened in the following example):

CREATE MINING MODEL TM\_CL FROM PMML

'<PMML version="2.1" xmlns="http://www.dmg.org/PMML-2\_1"   
…

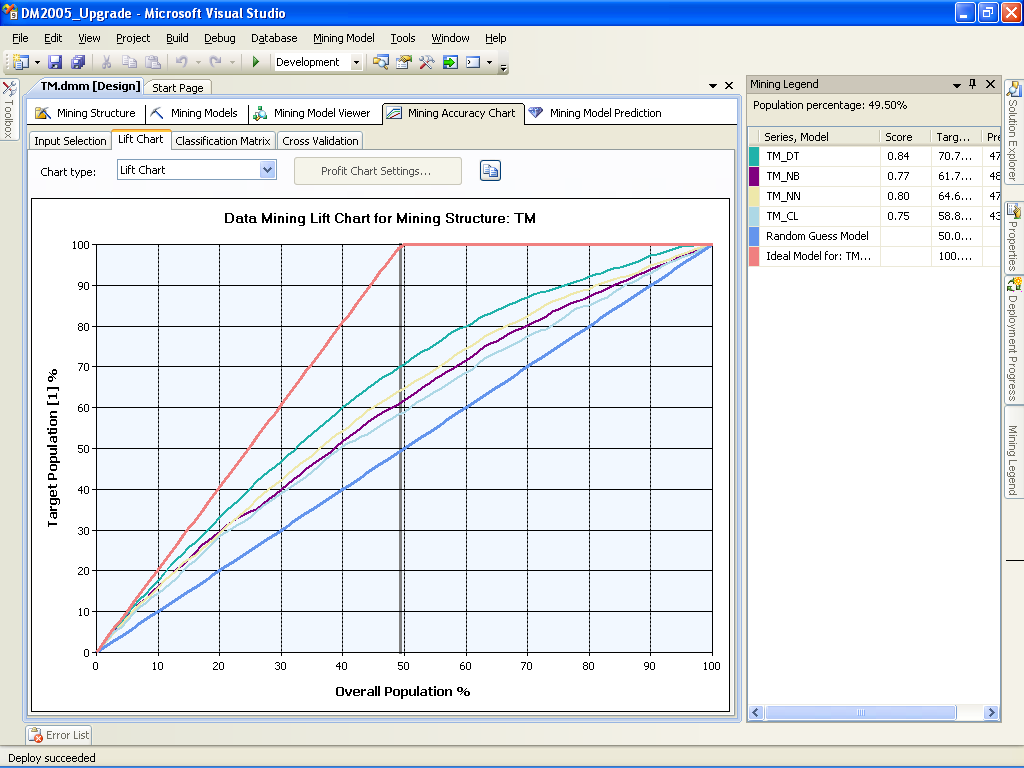
</PMML>';

### Post-Upgrade Tasks

As noted as part of the post-upgrade tasks when you upgrade from SSAS 2000, after an upgrade from SSAS 2005 to SSAS 2008, you should check the accuracy and the robustness of your predictive models before you decide which one to deploy in production.

#### Lift Chart

A Lift Chart is the most popular way to view the accuracy of predictive models. For a Lift Chart, you should split your data into training and test sets. You use the training set to train the models and then try to predict the target variable in the test set. Because you know the real value of the target variable in your test set, you can measure how many times the predictions were accurate and compare the accuracy of different models. The Lift Chart provides a standard way to graphically present this comparison. Figure 12-12 shows a Lift Chart for the predictive models that we created in the sample SSAS 2005 database for the value 1 (buyers) of the predicted variable (Bike Buyer).



**Figure 12-12:** Lift Chart for predicting a single value

From this chart, you can easily see the performance of different models. As you can see in Figure 12-12, the chart shows six curves and lines. The four curves show the predictive models, and the two lines represent the Ideal Model and the Random Guess. The x-axis represents the percentage of the overall population (all cases), and the y-axis represents the percentage of the target population (bike buyers). From the Ideal Model line (the topmost line), you can see that approximately 50 percent of AdventureWorks customers buy bikes. If you could predict with 100 percent probability which customer will buy a bike and which will not, you would have to target only 50 percent of the population to get all bike buyers. The lower line is the Random Guess line. If you would select cases of the population randomly, you would need 100 percent of the cases for 100 percent of bike buyers. Similarly, you would need 80 percent of the population for 80 percent of bike buyers, 60 percent of the population for 60 percent of bike buyers, and so on.

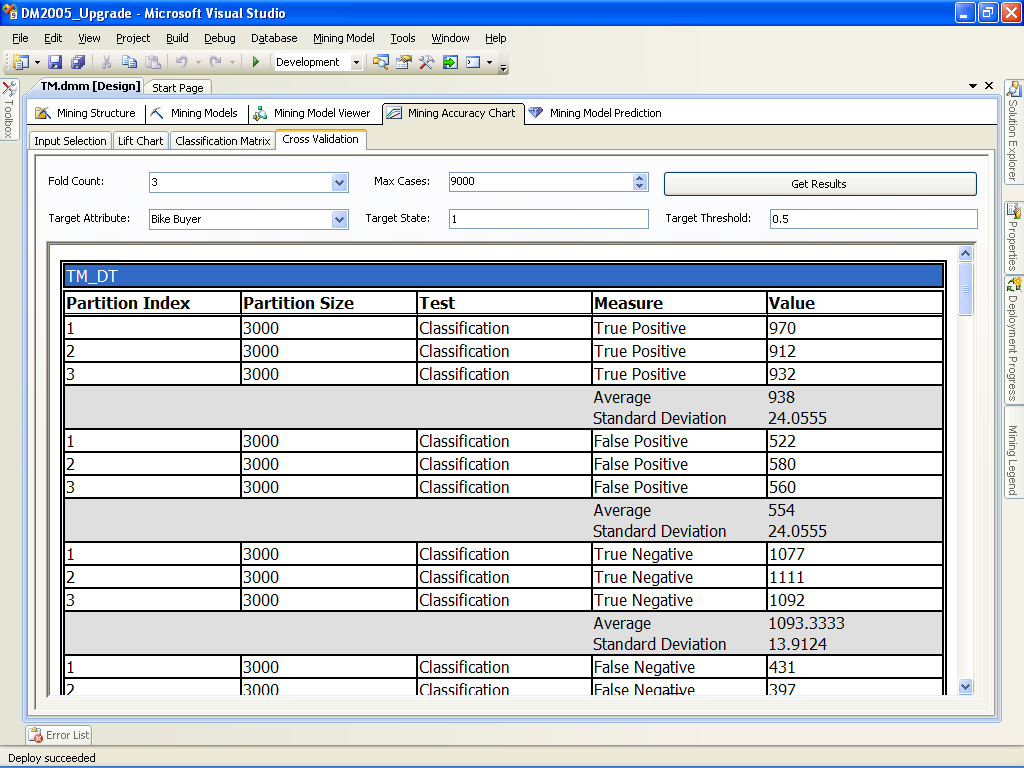
Data mining models give better results in terms of percentage of bike buyers than the Random Guess line but worse results than the Ideal Model line. From the Lift Chart, you can measure the *lift* of the mining models from the Random Guess line. Of course, a model predicts the outcome with less than 100 percent probability in all ranges of the population. Therefore, to get 100 percent of bike buyers, you still need 100 percent of the population.

Data mining models give you interesting results somewhere between 0 and 100 percent of the population. For example, if you take the highest curve, the one right below the Ideal Model line, you can see that if you select 70 percent of the population based on this model, you would get nearly 90 percent of bike buyers. From the Mining Legend window, you can see that this is the Decision Trees curve. For accuracy of predictions from the sample data that is used for analysis, the Decision Trees algorithm generates the best predictions, the Neural Network algorithm generates the second best, the Naïve Bayes algorithm generates the third best, and the Clustering algorithm generates the fourth best. In this example, if you checked the Lift Chart in SSAS 2005, you probably decided to deploy the Decision Trees model into production.

#### Cross-Validation

But what you cannot see from the Lift Chart is how reliable your predictive models are. You do not know whether they behave the same using different data—that is, how robust the predictions are with different data sets. In SSAS 2008, you can test the reliability of predictive models by using cross-validations. With cross-validation, you partition your training dataset into many smaller sections. SSAS creates multiple models on the cross-sections, using one section at a time as test data and other sections as training data, and then trains the models and creates many accuracy measures across partitions. If the measures across several partitions differ a lot, the model is not robust on different training/test set combinations.

Figure 12-13 shows the cross-validation settings that you can specify and the cross-validation results of predictive models.



**Figure 12-13:** Cross-validation of predictive models

You can define the following cross-validation settings:

* **Fold Count.** With this setting, you define how many partitions that you want to create in your training data. In Figure 12-13, three partitions are created. When partition 1 is used as the test data, the model is trained on partitions 2 and 3; when partition 2 is used as the test data, the model is trained on partitions 1 and 3; and when partition 3 is used as the test data, the model is trained on partitions 1 and 2.
* **Max Cases.** You can define the maximum number of cases to use for cross-validation. Cases are taken randomly from each partition. Our example uses 9,000 cases, which means that each partition will hold 3,000 cases.
* **Target Attribute.** This is the variable that you are predicting.
* **Target State.** You can check overall predictions by leaving this field empty, or you can check predictions for a single state that you are interested in. In our example, we are interested in bike buyers (state 1).
* **Target Threshold.** You use this parameter to set the accuracy bar for the predictions. If the predict probability exceeds your accuracy bar, the prediction is considered correct. If not, the prediction is considered incorrect.

The cross-validation report below the settings shows many measures to help you check the reliability of your models. For example, the classifications True Positive, False Positive, True Negative, and False Negative count cases in a partition where predict probability is greater than your accuracy threshold and predicted state matches target state.

You can see in Figure 12-13 that the True Positive classification of Decision Trees gives you very consistent results across partitions. The True Positive classification counts cases predicted as positive (bike buyers, in the example) that are actually positive. In addition, the standard deviation of this measure is not too high. However, when you check the Neural Network model, you can see that it is even more consistent for the True Positive classification, which means that this model is more robust on different data sets than the Decision Trees one. From the cross-validation results, it seems that you should deploy the Neural Network model in production: Although the accuracy of the Neural Network model is slightly lower than that of the Decision Trees model, the reliability is higher. Of course, in production, you should perform many additional accuracy and reliability tests before you decide which model to deploy. But testing the reliability of predictive models is one of the most important post-upgrade tasks when you upgrade to SSAS 2008. For more information, see [Cross-Validation (Analysis Services—Data Mining)](http://go.microsoft.com/fwlink/?LinkId=132570) in SQL Server 2008 Books Online.

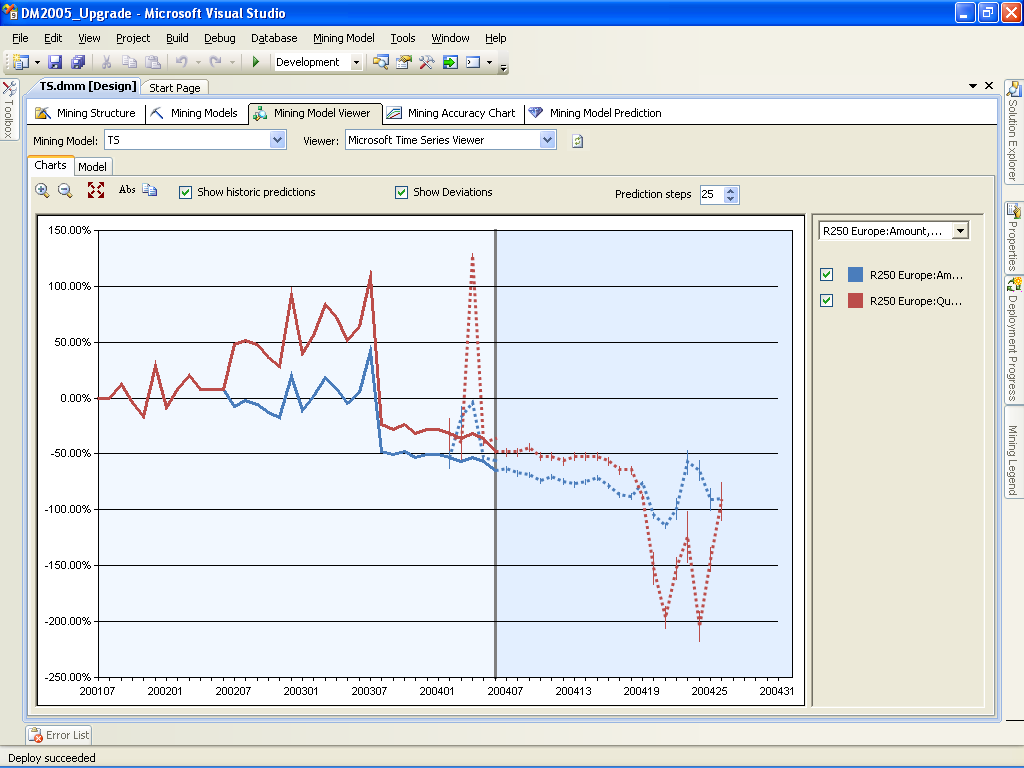
#### Model Filtering

In SSAS 2000 and SSAS 2005, you have to create a different mining structure if you want to use only a subset of data for an additional mining model. In SSAS 2008, you can filter a specific model to use only a subset of data for training. For example, by using the same structure, you can create a model that is trained on the complete training set, another model that is trained only on the female population subset, and the third model that is trained only on the male population subset. You can then compare the performance of the models that are trained on the complete population with those that are trained on the different subsets. If you used different structures for subsets of training data in SSAS 2000 and SSAS 2005, you should consider combining those structures into one structure in SSAS 2008 so that you can compare the performance of the models in a single Lift Chart or with a single cross-validation. For more information, see [Creating Filters for Mining Models (Analysis Services—Data Mining)](http://go.microsoft.com/fwlink/?LinkId=132571) in SQL Server 2008 Books Online.

#### Measuring Quality of Time Series Algorithm

How can you measure the quality of forecasted values with the Time Series algorithm when you do not have the actual data yet? Waiting until the data is available is likely not practical because by that time you might already have made wrong decisions based on your forecasting model. There is a better way to measure the performance of the Time Series model. Using a specific number of periods from the past, you can try to forecast present values. If the model performs well for forecasting present values, the probability is good that it will perform well for forecasting future values.

You control how historical models are created by using two algorithm parameters: HISTORICAL\_MODEL\_COUNT and HISTORICAL\_MODEL\_GAP. The first parameter controls the number of historical models that will be built, and the second parameter controls the number of time slices between historical models. Figure 12-14 uses SSAS 2005 to show historical forecasts (the dotted lines before the current point in time) for the R-250 model for sales quantity and amount in Europe. What you can see is that the forecasts are very unstable and, therefore, not very reliable. You can also see that the forecasts (the dotted lines after the current time point) stop after a future time point, about 20 points in the future in this example).

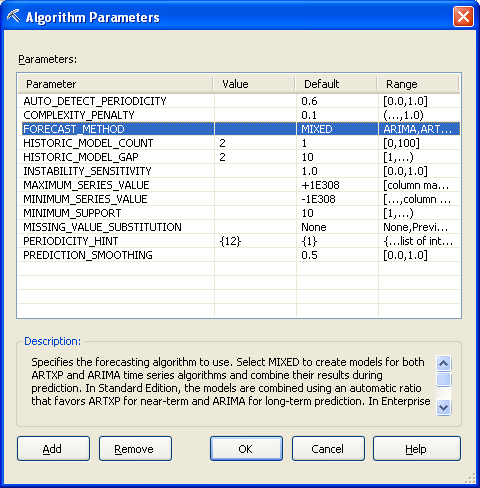


**Figure 12-14:** Historical and future forecasts in SSAS 2005

The reason for this instability is that SSAS 2005 Time Series use a single algorithm, Auto-Regression Trees (ART). This algorithm provides good short-term forecasts only. SSAS notes this instability in long-term forecasts and stops forecasting.

In SSAS 2008, you can use a blend of two Time Series algorithms for forecasting. In addition to ART, SSAS 2008 provides the Auto-Regressive Integrated Moving Average (ARIMA) algorithm, which is much better for long-term forecasts. After you upgrade your Time Series models to SSAS 2008, you should refine the blend of ART and ARIMA in your models by changing the FORECAST\_METHOD and PREDICTION\_SMOOTHING algorithm parameters. The first parameter uses an automatic method to determine the mixture of the algorithms, and the second parameter (available only in Enterprise) lets you define the blend manually.

As you can see in Figure 12-15, the upgraded version of the Time Series algorithm uses a MIXED forecast method. Therefore, ART is used for short-term forecasts and ARIMA for long-term forecasts.



**Figure 12-15:** Time Series algorithm parameters in SSAS 2008

For more information, see [Microsoft Time Series Algorithm Technical Reference (Analysis Services—Data Mining)](http://go.microsoft.com/fwlink/?LinkId=132572) in SQL Server 2008 Books Online.

## Conclusion

There are many good reasons to upgrade your data mining models to SQL Server 2008. If you are using SSAS 2000, SSAS 2008 gives you a complete set of modern algorithms, and you can create additional models that are based on the same structure to find the best-performing one for deployment in production. If you are using SSAS 2005, you probably already measure the accuracy of your predictive models, but you might decide to deploy a different model, depending on reliability. In addition, you can get much better long-term forecasting with the Time Series algorithm in SSAS 2008. Finally, you can combine multiple mining structures into one if you have to compare mining models that are trained on only a subset of the structure data.

For upgrading your data mining models, a side-by-side upgrade is preferred to an in-place one. The most important reason is that with a side-by-side installation, you leave your original models intact. However, if you have insufficient hardware power, you can perform an in-place upgrade, and with thorough testing and planning, an upgrade can go smoothly whether your mining models are in SSAS 2000 or SSAS 2005.

## Additional References

For an up-to-date collection of additional references for upgrading data mining to SQL Server 2008, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkId=132510).

For the latest updates to SQL Server 2008 Books Online, see the [Microsoft SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132511).

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkId=132512)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132514)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkId=132515)

# Integration Services

## Introduction

This chapter is addressed to existing SQL Server customers who have developed extraction, transformation, and loading (ETL) solutions with SQL Server 2000 and SQL Server 2005 and want to upgrade these to SQL Server 2008. Before we dive into this topic, we should review the history of ETL functionality in SQL Server.

SQL Server 2000, which included Data Transformation Services (DTS), was the first major database vendor product release to include ETL functionality in the core product. This resulted in DTS being widely adopted by customers. However, DTS lacked various features and functionality that were available in other enterprise ETL products.

SQL Server 2005 introduced SQL Server Integration Services (SSIS), which was architected from the ground up to support customers’ scalable enterprise ETL needs. Because SSIS and DTS were different code bases, Microsoft provided add-ons for running and editing existing DTS packages, as well as a Package Migration Wizard, which converted DTS packages to SSIS.

SQL Server 2008 extends the capabilities of SSIS and introduces new relational engine capabilities, such as MERGE and Change Data Capture that you can use in ETL applications. To learn more about these and other SSIS features, see [What’s New (Integration Services)](http://go.microsoft.com/fwlink/?LinkId=132613) in SQL Server 2008 Books Online.

This chapter will lead you through both the DTS and SSIS upgrade process and is organized into the following sections: preparing to upgrade, installing SSIS 2008 and add-ons for DTS support, migrating DTS packages, and upgrading SSIS packages.

## Preparing to Upgrade to SSIS 2008

Customers upgrading to SSIS 2008 from either SSIS 2005 or DTS 2000 have multiple installation options available, including upgrading in-place, performing a side-by-side installation, and performing a new installation.

Putting all of these options aside for now (we will cover them later), the key to planning an upgrade to SSIS 2008 is to view the upgrade as a three-step process:

1. **Performing the installation/upgrade process.** First, determine what files and directories are created (and removed for upgrades) as well as what system data gets moved or converted to the SQL Server 2008 format for upgrades. In addition, you need to know what other component installs are required if you want DTS compatibility.
2. **Running the DTS Package Migration and the Package Upgrade wizards.** You run these wizards to migrate existing DTS and SSIS packages, respectively, to SSIS 2008.
3. **Performing post-upgrade steps.** This step includes performing any tasks that should occur after the wizards have produced SSIS 2008 packages.

To learn more about Step 1, see [Considerations for Upgrading Integration Services](http://go.microsoft.com/fwlink/?LinkId=132615) in SQL Server 2008 Books Online. Steps 2 and 3 are the primary focus of this chapter.

It takes less effort to upgrade existing SSIS packages than DTS packages, given that SSIS 2008 is more about enhancing existing functionality and performance and less about introducing broad, sweeping changes. However, you still need to do some planning for SSIS upgrades. You will need to convert existing SSIS 2005 packages to the SSIS 2008 package format, and you might need to make some changes to the packages after the upgrade.

Whether you have DTS packages or SSIS 2005 packages, you should run the SQL Server 2008 Upgrade Advisor to find upgrade issues and resolve them. These issues are covered in more detail in the “Upgrade Advisor” and “Backward Compatibility” sections, later in this chapter.

### Preparing to Upgrade from DTS

Existing DTS customers have the following options for upgrading DTS packages to SSIS 2008:

* Use the Migration Wizard to convert DTS packages to SSIS (with some restrictions).
* Rewrite the DTS packages to fully leverage SSIS capabilities.
* Use a third-party tool (such as DTS xChange from Pragmatic Works Software).
* Continue to run and modify existing DTS packages (with some restrictions).

**Note:** DTS is a deprecated feature in SQL Server 2008. This means that the next SQL Server release might not include support for DTS. To prepare for this, customers should have a migration strategy in place for each of their existing DTS packages. This also applies to all programs and scripts that call the DTS object model.

You should start planning your approach in preparation for the day when DTS is no longer supported. The following is one approach that you can take for your existing DTS packages:

1. Run Upgrade Advisor before installing SQL Server 2008 to better understand how your DTS packages will migrate to SSIS 2008.
2. Move DTS packages to your SQL Server 2008 environment and continue to run them as DTS packages.
3. Use SSIS to build any new packages you need.
4. Create an inventory of your existing DTS packages and assign them a priority and order for migration. Do not forget applications that use the DTS object model.
5. Use the DTS Migration Wizard as a starting point for moving strategic packages to SSIS.
   1. Review any DTS tasks that were successfully migrated. You can leverage these in SSIS.
   2. Replace DTS code that was not successfully migrated with new SSIS code.
6. Plan for a rolling strategy to rework all packages, leveraging the complete SSIS feature set in the redesign.

For more information, see [Upgrading Data Transformation Services How-to Topics](http://go.microsoft.com/fwlink/?LinkId=132615) in SQL Server 2008 Books Online.

#### Migrating DTS Packages

The DTS Package Migration Wizard is included in SQL Server 2008 and uses a “best effort” strategy to generate new SSIS packages that replicate each DTS package’s functionality. This means that DTS functionality that cannot be migrated is encapsulated into a DTS 2000 Execute Package task, which is supported by SSIS 2008. Given this, the primary consideration for DTS package migration is the existing DTS functionality, which we discuss in more detail in the “Comparing DTS and Integration Services Functionality” section, later in this chapter.

A second consideration is that the DTS Package Migration Wizard does not support reading DTS packages stored in the SQL Server 2000 Meta Data Services repository. Therefore, you must first open DTS packages stored in the SQL Server 2000 Meta Data Services repository and then save the DTS package to a supported package store, namely the Windows file system (.dts) and the msdb database.

#### Rewriting DTS Packages

SSIS has many functional enhancements over DTS, including data flows, looping constructs, event handlers, robust variables, logging, configurations, and performance enhancements. This means that many existing DTS applications could better leverage SSIS features and performance enhancements if they were rewritten to use SSIS instead.

One example is SSIS data flows and their ability to apply robust, high-performance transformations to the data pipeline as it passes from source to destination. DTS data pumps did not have these capabilities, so a migration of existing DTS packages could never take full advantage of the SSIS Data Flow component.

Note that rewriting DTS applications is beyond the scope of this chapter, but the links in the “Additional References” section at the end of this chapter point you to online best practices and guidance for rewriting DTS applications to SSIS.

#### DTS xChange

[DTS xChange](http://go.microsoft.com/fwlink/?LinkId=132625) is an enterprise solution offered by a Microsoft partner, Pragmatic Works Software. This solution migrates DTS packages to SSIS while applying a series of best practice rules to the packages. The solution has three parts:

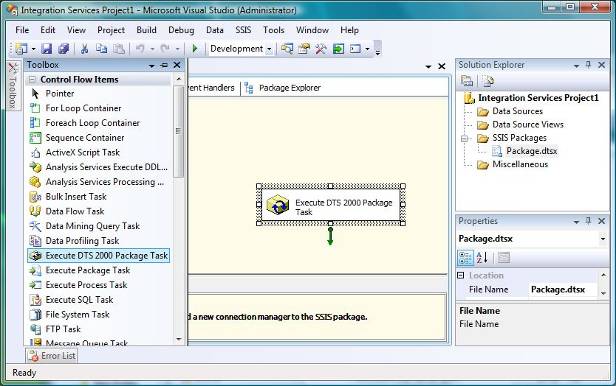
* **Profile** – DTS xChange Profiler helps you estimate your migration project in hours and dollar cost, whether you choose to use an automation tool or not.
* **Convert** – DTS xChange will migrate your packages, applying rules to each DTS package during the process to enforce best practices.
* **Monitor** – The SSIS Performance Warehouse is a software development kit (SDK) to help you get the most out of your new SSIS environment. It contains a series of reports and a data warehouse to monitor your SSIS package execution.

The DTS xChange Profiler feature lets you profile how much of a migration effort, in hours and dollars, it will be to completely migrate to SSIS. The process lets you specify how long you believe each type of task will take to migrate, whether you choose to use DTS xChange or manually re-engineer the package. The result is a report showing the migration cost in hours and dollars for each package, as well as the total cost for all packages.

#### SQL Server 2008 Support for DTS Packages

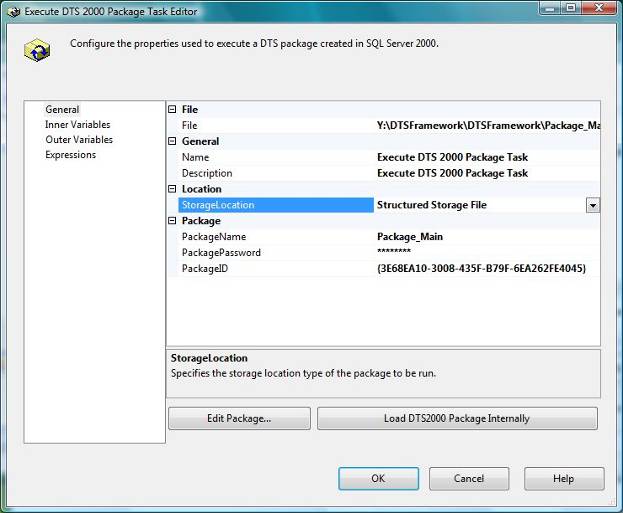
SSIS 2008 supports the execution, development, and management of DTS packages. Before continuing, let’s briefly review this DTS support.

DTS package execution is supported through the SSIS Execute DTS 2000 Package task, which Figure 13-1 shows.



**Figure 13-1:** Execute DTS 2000 Package Task

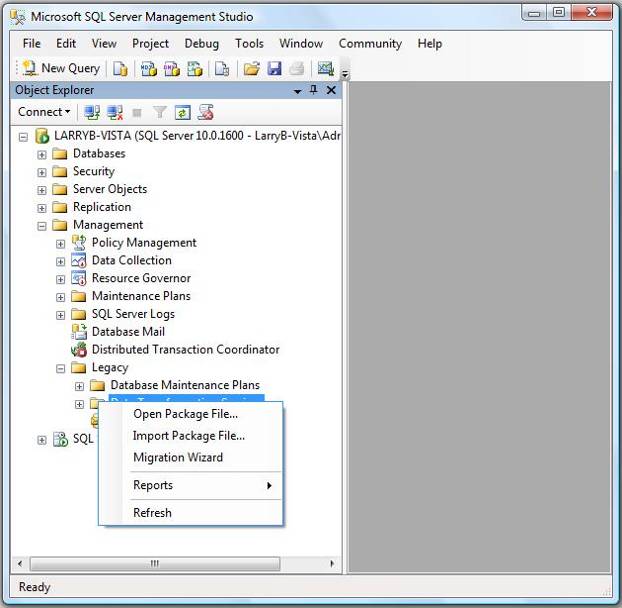
This SSIS 2008 DTS 2000 Package task provides the same functionality as the DTS Execute Package task, with some enhancements, as Figure 13-2 shows. These enhancements include the ability to store the DTS package in the task itself, as opposed to a separate package store. The DTS Migration Wizard leverages this capability when it encapsulates DTS code that cannot be migrated to SSIS.



**Figure 13-2:** Execute DTS 2000 Package Task Editor

For more information, see [Execute DTS 2000 Package Task](http://go.microsoft.com/fwlink/?LinkId=132617) in SQL Server 2008 Books Online.

You can also manage and migrate existing DTS packages from SQL Server 2008 Management Studio. You can access this functionality in the Data Transformation Services folder’s right-click context menu, under Management, Legacy in Object Explorer, as Figure 13-3 shows.



**Figure 13-3:** Managing DTS 2000 packages

For more information about migrating DTS to SSIS, see [Support for Data Transformation Services (DTS) in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132618) in SQL Server 2008 Books Online.

As you prepare for your upgrade, you need to consider SQL Server 2008 backward-compatibility issues, including deprecated features, discontinued features, changes that might require package modifications and changes that might affect package behavior. Let’s look at each of these items.

### Deprecated Features

This topic describes the SQL Server Integration Services (SSIS) features that are still available in SQL Server 2008 but that are scheduled to be removed in the next release or a subsequent release of SQL Server.

#### DTS Object Model

The DTS object model is deprecated in SQL Server 2008. You should plan to remove dependencies on this object model in anticipation of future releases of SQL Server. The DTS object model will continue to be supported in SQL Server 2000, SQL Server 2005, and SQL Server 2008 releases. However, the ability to migrate DTS packages might not be supported in the next release.

The deprecated DTS functionality includes the DTS API, the DTS Package Migration Wizard, support for DTS package maintenance in SQL Server Management Studio, and the Execute DTS 2000 Package task.

For information about how to migrate DTS packages to the SSIS package format, see [Migrating Data Transformation Services Packages](http://go.microsoft.com/fwlink/?LinkId=132619) in SQL Server 2008 Books Online.

#### ActiveX Script Task

The ActiveX Script task in SQL Server 2008 is provided only for backward compatibility with DTS and will be removed in the next release of SQL Server. Use the Integration Services Script task for all new scripting tasks, and migrate your existing ActiveX scripts to Integration Services Script tasks or use SSIS capabilities such as expressions to recreate the ActiveX scripts’ functionality.

For more information about deprecated features, see [Deprecated Integration Services Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132620) in SQL Server 2008 Books Online.

### Discontinued Functionality

This topic describes the SSIS features that are no longer available in SQL Server 2008.

#### Visual Studio for Applications

In SQL Server 2008, Microsoft Visual Studio Tools for Applications has replaced Visual Studio for Applications as the scripting environment for the Script task and Script component. Although this change does not require you to rewrite scripts, it does require you to convert Script tasks and transformations that contain Visual Studio for Applications scripts to Visual Studio Tools for Applications when you upgrade existing SSIS packages to SQL Server 2008.

The SQL Server 2008 Package Upgrade Wizard will convert Visual Studio for Applications scripts (contained in Script tasks or transformations) to Visual Studio Tools for Applications before SSIS runs the scripts. Microsoft no longer supports Visual Studio for Applications or Script tasks or transformations that contain Visual Studio for Applications scripts.

For more information about discontinued functionality, see [Discontinued Integration Services Functionality in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132621) in SQL Server 2008 Books Online.

### Breaking Changes

The following are changes from SSIS 2005 to SSIS 2008 that you should be aware of when upgrading:

* **Connection strings.** Connection strings have changed for some key providers. Examples include the native SQL Server Native Client provider (SQLNCLI.1 to SQLNCLI10.1) and Analysis Services (MSOLAP.3 to MSOLAP.4).
* **Third-party components.** Existing third-party components will need to be recompiled to support SSIS 2008’s new programming API.

For more information, see [Breaking Changes to Integration Services Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132622) in SQL Server 2008 Books Online.

### Behavior Changes

The following are changes that might affect existing SSIS packages when they are upgraded to SSIS 2008:

* Different values are returned to package variables from an Execute SQL task.
* You use different tools for developing scripts in the Script task and data flow Script component. Visual Studio Tools for Applications is used for scripting in SSIS 2008, whereas Visual Studio for Applications was used for scripting in SSIS 2005.
* The data flow Lookup component has been enhanced. Developers should open, review, and possibly revise all Lookup component settings in SSIS data flows.
* The SQL Server 2008 SQL Server Agent SSIS Package job step now invokes DTEXEC.EXE, as compared to the SQL Server 2005 SSIS Package job step, which invoked DTEXECUI.EXE.

For more information, see [Behavior Changes to Integration Services Features in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132623) in SQL Server 2008 Books Online.

After taking these changes into consideration, you are now ready to run Upgrade Advisor.

### Upgrade Tools

#### SQL Server 2008 Upgrade Advisor

Upgrade Advisor can examine both DTS and SSIS 2005 packages to locate package tasks and data flow transformations that need to be migrated and/or modified.

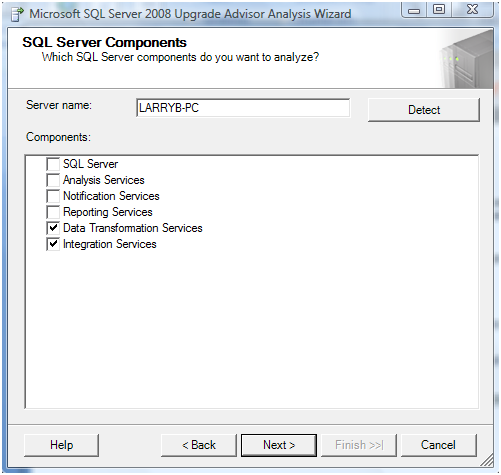
For details about Upgrade Advisor general installation and execution, see Chapter 1, “Upgrade Planning and Deployment,” and [Using Upgrade Advisor to Prepare for Upgrades](http://go.microsoft.com/fwlink/?LinkId=132624) in SQL Server 2008 Books Online.

The following are prerequisites for running Upgrade Advisor for DTS packages:

* SQL Server 2000 Client components are required to scan SQL Server 2000 DTS packages.
* SQL Server 2005 Backward Compatibility Components are required to scan SQL Server 2005 DTS packages that were migrated from SQL Server 2000.

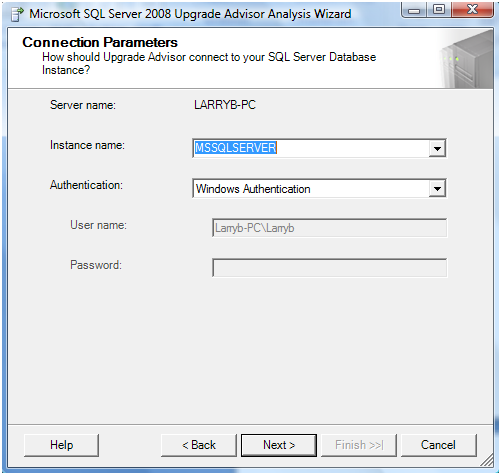
See the “Installing DTS Support” section in this chapter to learn more about installing these components.

Upgrade Advisor gives you the option of analyzing both DTS packages and SSIS 2005 packages. Figure 13-4 shows an example where we have selected both components for analysis.



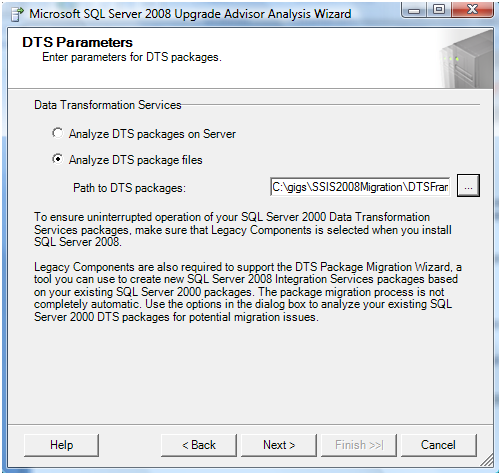
**Figure 13-4:** Selecting the DTS and Integration Services components to analyze

Figure 13-5 shows the next Upgrade Advisor screen, which prompts the user for the SQL Server 2000 or SQL Server 2005 database instance name. Note that this screen cannot be skipped, even if all of your existing DTS and SSIS packages are stored outside SQL Server as files in the Windows file system.



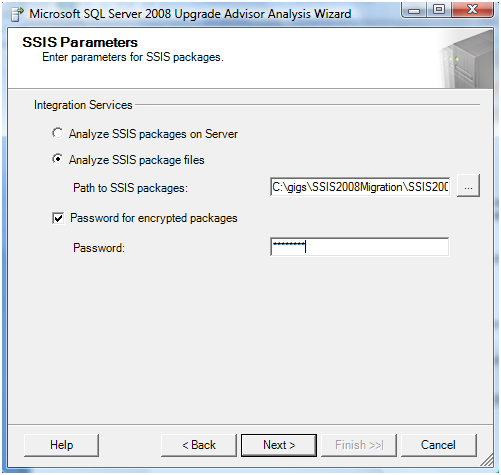
**Figure 13-5:** Connecting to a SQL Server database

Next, you select the location of your DTS packages. You need to specify a Windows folder path if you have selected the *Analyze DTS package files* option shown in Figure 13-6. If you select the *Analyze DTS packages on Server* option, you do not need to provide additional parameters.



**Figure 13-6:** Specifying the DTS package location

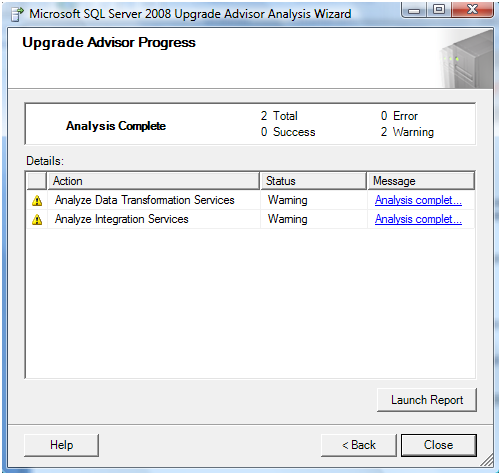
After clicking Next, you will be prompted for the password for encrypted packages, as Figure 13-7 shows.



**Figure 13-7:** Specifying the SSIS package location

**Note:** The Upgrade Advisor Wizard prompts for only one password. Installations that contain encrypted packages with different passwords will need to run Upgrade Advisor once for each password, modify the packages to be unencrypted, or modify the packages to use only one password. See Figure 13-8 for the message that is displayed in the report when this condition occurs.

After you click Next, Upgrade Advisor examines all selected packages and provides a high-level status, as Figure 13-8 shows.



**Figure 13-8:** Analysis complete

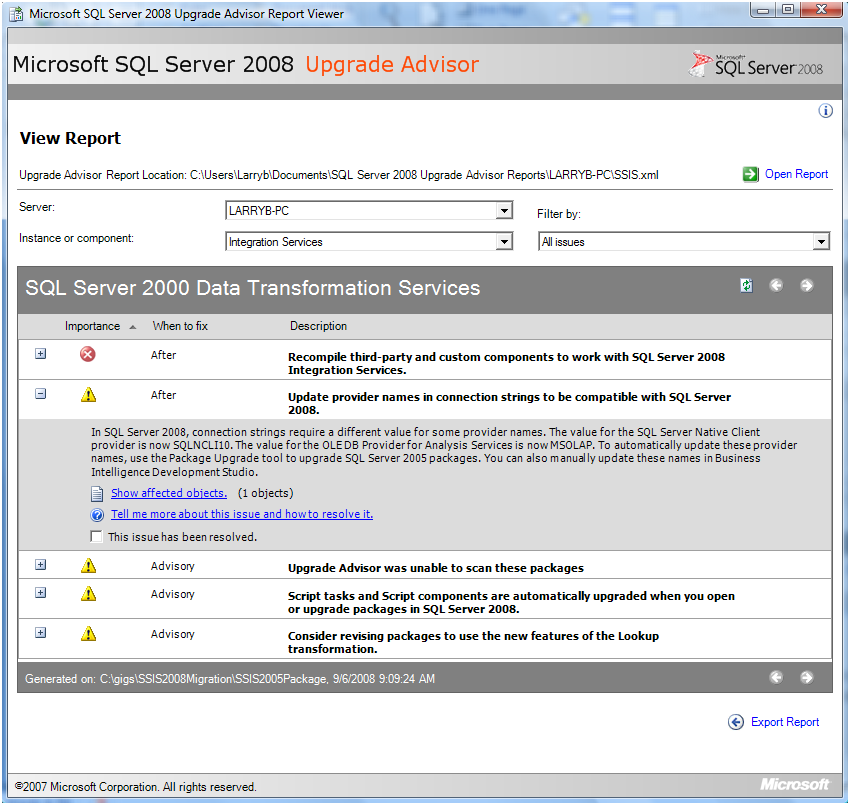
Click Launch Report to see Upgrade Advisor’s findings. Figure 13-9 shows an example of an Upgrade Advisor report.



**Figure 13-9:** Upgrade Advisor DTS report

In this example, Upgrade Advisor found instances of the Dynamic Properties task in the DTS packages. Clicking the plus sign for this entry provides details about this warning and a link to which DTS packages contain the Dynamic Properties task.

You can then select Integration Services from the *Instance or component* drop-down list. Figure 13-10 shows the Upgrade Advisor report on the SSIS packages that were examined.

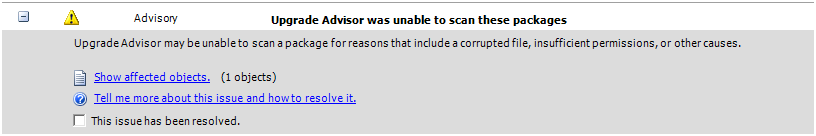


**Figure 13-10:** Upgrade Advisor SSIS report

In this example, Upgrade Advisor found the following problems:

* **A custom component.** This will need to be recompiled with SSIS 2008 before it can be used.
* **Connection strings.** These will need to be modified to reflect the latest provider name.
* **Script tasks and components.** Visual Studio for Applications script tasks and script transformations will be upgraded to Visual Studio Tools for Applications.

Note that Upgrade Advisor was unable to open an encrypted package with the password provided in the SSIS Parameters screen. Figure 13-11 shows the message detail.



**Figure 13-11:** Unable to scan package error

The example above shows only a subset of DTS issues that Upgrade Advisor detects. For more information, see [Known DTS Package Migration Issues](http://go.microsoft.com/fwlink/?LinkId=133165) in SQL Server 2008 Books Online.

The one issue that Upgrade Advisor does not detect that you should be aware of is the need to delete and recreate ODBC connections after package migration.This issue has been corrected and is fixed in SQL Server 2008 Service Pack 1 (SP1).

### Coexistence with Previous Versions

The bottom line for SSIS 2008 coexistence with DTS and SSIS 2005 is that existing support for developing, managing, and executing existing DTS packages will be unchanged if an existing instance of the SQL Server 2000 or SQL Server 2005 relational engine remains on the server after the installation of SQL Server 2008—that is, in a side-by-side installation.

SSIS 2008 also provides support for SSIS 2005. Because the SSIS package format has changed in SQL Server 2008, remember the following:

* SSIS 2005 does not support the new SQL Server 2008 package format.
* SSIS 2008 supports the SQL Server 2005 package format by first converting it to the SQL Server 2008 package format. This support includes the SQL Server 2008 dtexec utility, which converts the SQL Server 2005 package into the SQL Server 2008 format in memory before running the package.

**Note:** The package will not run if issues are encountered during the conversion of the SQL Server 2005 package format to the SQL Server 2008 package format. To avoid this problem, first upgrade your existing SSIS packages to SQL Server 2008 before running them in SQL Server 2008.

Because each version of every SQL Server ETL component has a different package format, each also has its own development tool for editing and maintaining that particular package format, as follows:

* For SQL Server 2000 databases and DTS, use SQL Server 2000 Enterprise Manager.
* For SSIS 2005 packages, use BIDS 2005 and SQL Server Management Studio 2005.
* For SSIS 2008 packages, use BIDS 2008 and SQL Server Management Studio 2008.

Note that SQL Server 2008 installations or installations that upgrade in-place will require SQL Server 2008 support for DTS, which we cover later in this chapter’s Installation section.

In addition, see the “64-bit Considerations” section below for information about DTS limitations in a 64-bit environment.

For more information about coexistence, see [Interoperability and Coexistence (Integration Services)](http://go.microsoft.com/fwlink/?LinkId=133166) in SQL Server 2008 Books Online.

### 64-bit Considerations

In general, you will want to run SQL Server Integration Services in 64-bit mode to take advantage of the additional memory space. However, in the following scenarios you might want to run packages in 32-bit mode even if you have an x64-bit system:

* To run uncompiled scripts.
* To run SQL Server 2000 DTS packages.
* To use a managed .NET Framework data provider or native OLE DB provider that is not available in a 64-bit version.

All available 32-bit and 64-bit design-time and run-time SSIS features are installed when you select both the Integration Services and Business Intelligence Development Studio SQL Server 2008 installation options. Installing Integration Services also installs 32-bit run-time support for SQL Server 2000 DTS packages.

64-bit features are installed in the Program Files directory, and 32-bit features are installed separately in the Program Files (x86) directory. (This behavior is not specific to SSIS or to SQL Server.)

The 64-bit editions of SQL Server support SSIS, with some restrictions. Some SSIS features are available only in 32-bit versions, some have limitations on 64-bit computers, and some are not supported on Itanium-based operating systems. Table 13-1 lists SSIS tool and feature support by environment.

Table 13-1: SSIS Feature Support on 64-bit Platforms

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **X64 32-bit** | **X64** | **Itanium 64** |
| Dtexec.exe | Yes | Yes | Yes |
| Dtutil.exe | Yes | Yes | Yes |
| Dtswizard.exe | Yes | Yes | Yes |
| SQL Server Agent SSIS Support | Yes | Yes | Yes |
| Dtexecui.exe | Yes | **No** | **No** |
| DTS (design time) | Yes | **No** | **No** |
| DTS (run time) | Yes | **No** | **No** |
| BIDS 2008 | Yes | **No** | **No** |

#### Itanium Limitations

The following are important considerations for Itanium-based systems:

* BIDS, the 32-bit development environment for SSIS packages, is not supported on the Itanium 64-bit operating sytem and is not installed on Itanium-based computers.
* DTS support is not included in Itanium. Therefore, you cannot create, view, modify, or run DTS packages on Itanium-based operating systems.

#### 64-bit Limitations

The following are DTS limitations that you must consider for x64 systems:

* **Dtsexecui.exe.** This 32-bit tool runs packages in 32-bit mode. You should use the 64-bit version of the dtexec utility to test the commands in 64-bit mode before installing your package(s) on production servers.
* **DTS run-time support.** SQL Server 2000 DTS run-time support is available only in 32-bit mode on x64 systems.
* **DTS design-time support.** There is 32-bit design-time support for DTS packages.

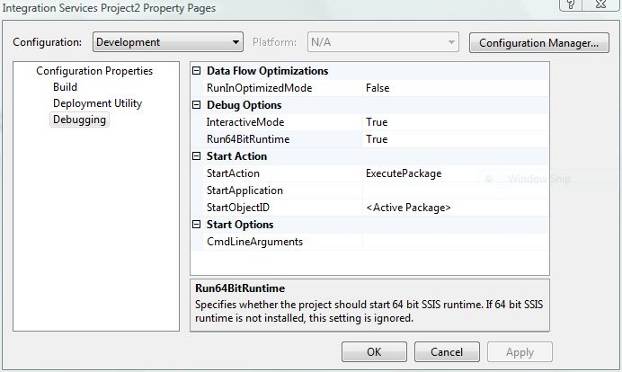
For DTS support details, see [Support for Data Transformation Services (DTS) in SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=132618) in SQL Server 2008 Books Online.

For more information about 64-bit considerations, see [64-bit Considerations for Integration Services](http://go.microsoft.com/fwlink/?LinkId=133167) in SQL Server 2008 Books Online.

#### 64-bit Data Provider Considerations

The primary consideration for data providers is whether there is a 64-bit version of the driver. Some .NET Framework data providers and native OLE DB providers are not available in 64-bit versions, such as the Microsoft OLE DB Provider for Jet, which connects to Access databases and Excel spreadsheets, and the SQL Server Compact Provider. Addition considerations exist for both design time and run time.

**The Integration Services Designer requires the 32-bit data provider.** You must install the 32-bit version of the provider for all 64-bit providers on the development computer. Note that even though the designer is a 32-bit application, you can still run the package from the designer in 64-bit mode by setting the Run64BitRuntime project property to True (the default setting), as Figure 13-12 shows.



**Figure 13-12:** Setting the 64-bit runtime option for Design mode

This setting applies only to the execution within your design-time session.

The SSIS runtime will select the appropriate version of the provider to use depending on whether it is running in 32-bit or 64-bit mode (both versions of the data provider have the same ID).

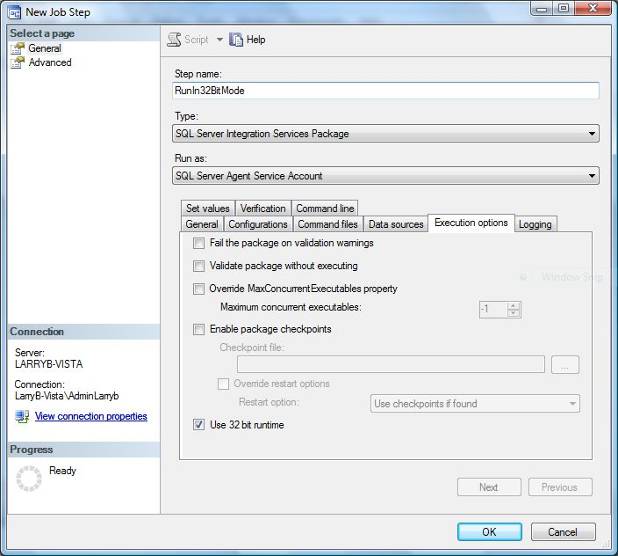
#### SQL Server Agent SSIS Package Execution Options

A SQL Server 2008 SQL Server Agent job with a job step type of SQL Server Integration Services Package will invoke the dtexec utility. This is new behavior in SQL Server 2008; the SQL Server 2005 Agent always invokes the 32-bit dtexeui.exe utility.

The version of the dtexec utility that the job invokes depends on what versions of SQL Server and SQL Server Agent have been installed and are running on the 64-bit computer along with the options you set for the job step.

The SSIS package specified in the SQL Server Integration Services Packagejob step will**:**

* Always run in 32-bit mode on systems where the 32-bit versions of SQL Server and SQL Server Agent have been installed and are running.
* Run in 64-bit mode by default on systems where the 64-bit versions of SQL Server and SQL Server Agent have been installed and are running.
* Runs in 32-bit mode (on 64-bit systems) when the *Use 32 bit runtime* option is selected (as Figure 13-13 shows).



**Figure 13-13:** The SSIS 2008 Package Job step 32-bit runtime option

In addition, you can invoke the 32-bit version of the dtexec utility from either a command line or a batch. In these cases, the Execute Package Utility (dtexecui.exe) command-line option is a useful feature for building the dtexec command-line syntax.

### Data Providers

A key architecture question for SSIS is often which data provider to use when there are multiple providers available (for example, when accessing an Oracle database). This document does not attempt to provide an answer to this question for each possible data source. But you can find useful information at the following two links:

* Matt Masson’s blog entry [New Connectivity Options in 2008](http://go.microsoft.com/fwlink/?LinkId=133170) discusses the SQL connection pack, which will be shipping later this year. This release will provide enhanced support for Oracle, SAP and Teradata.
* The [Integration Services wiki](http://go.microsoft.com/fwlink/?LinkId=133171) provides a lot of useful content. Note that because this is a community resource, Microsoft cannot take responsibility for any content on this site.

See the “Additional References” section at the end of this chapter for more information about these links.

### Failover Clustering

The SQL Server 2008 Integration Services service is not a Windows failover cluster or cluster-aware service. Therefore, Microsoft does not recommend that you install SSIS on clustered systems. However, administrators can manually configure SSIS in a clustered environment.

Note that architects might want to install databases that do not require a clustered environment on the same server as SSIS 2008 for performance reasons. This includes staging and working databases as well as some reporting databases.

For more information about SSIS and clustering, see [Configuring Integration Services in a Clustered Environment](http://go.microsoft.com/fwlink/?LinkId=133168) in SQL Server 2008 Books Online. For more information about upgrading failover clusters, see Chapter 4, “High Availability.”

### Known Issues and Workarounds

#### SSIS 2008 Installation Options

Administrators can choose from the following options when installing SSIS 2008:

* **New installation.** Install Integration Services and/or the Database Engine on a server without a previous version of SQL Server.
* **Side-by-side upgrade.** Install a new instance of SQL Server 2008 on a server that has an existing instance of SQL Server.
* **In-place upgrade.** Upgrade to SQL Server 2008 from an existing instance of SQL Server.

The impact of your upgrade decision on existing ETL processes, for both DTS and SSIS, depends on whether your goal is to:

* Keep existing SQL Server 2000 and SQL Server 2005 instances in place to manage and run existing DTS and SSIS packages (side-by-side installation).
* Upgrade and migrate all DTS and SSIS packages to SSIS 2008.
* Install or upgrade to SQL Server 2008 while still supporting DTS applications.

You will need to understand the options available to you in the SQL Server Installation Setup procedure if your goal is any approach other than keeping existing management tools in place (that is, side-by-side installation). These options were discussed in the “Coexistence with Other Versions” section, earlier in this chapter.

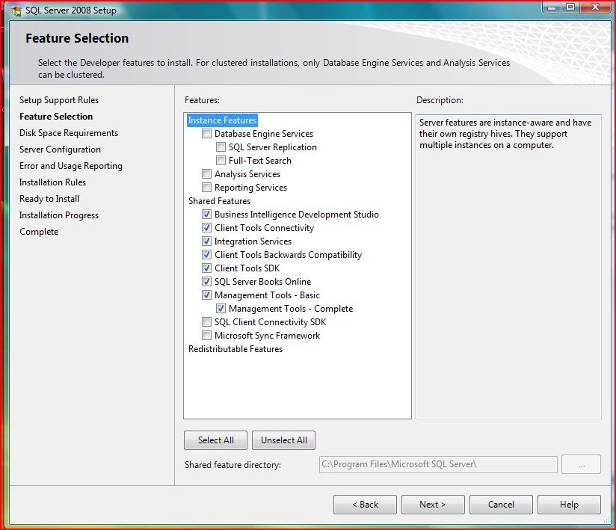
#### Running SQL Server 2008 Setup for SSIS

SSIS is installed using the SQL Server 2008 unified Setup application. The features that you select depend on whether you want your target system to serve as an SSIS run-time and/or development system. In addition, you will select additional options for DTS run-time and design-time support.

Note that installing the Integration Services components will not affect the current DTS installation or existing development and management tools, such as SQL Server 2000 Enterprise Manager or SQL Server Management Studio 2005.

Figure 13-14 shows the component selection screen used by the Setup application, with Integration Services (as well as the workstation components) selected for installation.

**Figure 13-14:** Select components during SQL Server 2008 setup



The recommended feature selection options for Integration Services are Business Intelligence Development Studio, Client Tools Connectivity, Integration Services, Client Tools SDK, SQL Server Books Online, and Management Tools – Complete.

Selecting the Client Tools Backward Compatibility option will install the Data Transformation Services 2000 Execute Package Task.

Review these installation notes:

* Although Integration Services components can be installed without other SQL Server components, it is recommended that you install the Database Engine along with Integration Services. This will let you store packages in the msdb database as well as use SQL Server for working and staging tables.
* Installing the SQL Server 2008 Integration Services components would allow packages to be stored and executed, but not developed, on a server.
* The workstation components can be installed without installing the SQL Server 2008 Integration Services components. In this scenario, packages can be developed and tested in the BIDS 2008 environment but cannot be executed locally outside the BIDS environment.
* The Client Tools Backwards Compatibility option installs the Execute DTS 2000 Package Task. However, this alone does not provide complete support for DTS. See the “Installing DTS Support” section below for more information about installing components for DTS support.

Before you upgrade to SSIS 2008, see [Considerations for Installing Integration Services](http://go.microsoft.com/fwlink/?LinkId=133169) in SQL Server 2008 Books Online.

#### Installing DTS Support

Options available in the SQL Server 2008 installation do not include full support for DTS. You have to additionally install the following components for SSIS 2008 DTS run-time and design support.

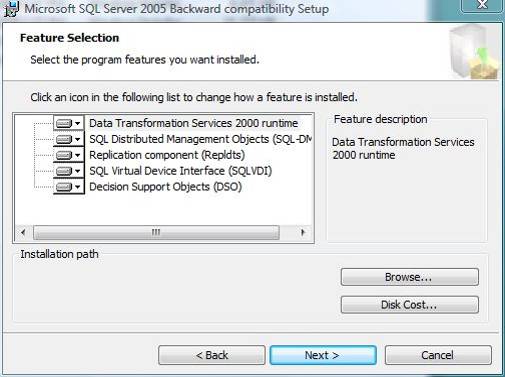
* The SQL Server 2005 Backward Compatibility installer package for DTS run-time support.
* The SQL Server 2000 DTS Designer Component installer package for DTS design support.

Note that the order of installation is important to enable support for DTS design capabilities. The SQL Server 2005 Backward Compatibility installer package must run after the SQL Server 2000 DTS Designer Components have been installed. You can also rerun the SQL Server 2005 Backward Compatibility installer and choose *repair* if it was installed before the installation of the SQL Server 2000 DTS Designer Components.

You can find the SQL Server 2005 Backward Compatibility install file in the following locations:

* On the SQL Server 2008 installation CD (SQLServer2005\_BC.msi). Make sure you use the correct directory (such as \x86, \x64, \ia64) for your target system.
* From the [SQL Server 2008 Feature Pack](http://go.microsoft.com/fwlink/?LinkID=133088) page. Make sure you download the correct version for your platform (such as SQLServer2005\_BC.msi, SQLServer2005\_BC\_x64.msi, or SQLServer2005\_BC\_ia64.msi).

Invoking the SQL Server 2005 Backward Compatibility installation will install the Data Transformation Services 2000 runtime by default, as Figure 13-15 shows.

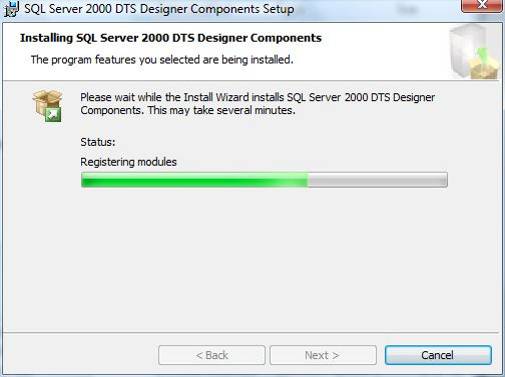


**Figure 13-15:** SQL Server 2005 Backward compatibility Setup: Feature Selection page

The SQL Server 2000 DTS Designer Component MSI can be downloaded from the [Feature Pack for Microsoft SQL Server 2005](http://go.microsoft.com/fwlink/?LinkID=51230) page.

Note that a more recent version of the DTS Designer will be included in SQL Server 2005 SP3.

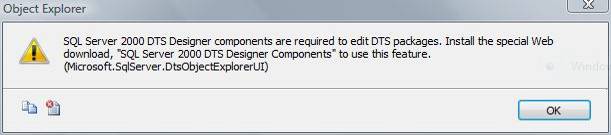
The SQL Server 2000 DTS Designer Components Setup has no feature selection options and will install after you click Next, as Figure 13-16 shows.



**Figure 13-16:** DTS Designer Components Setup

As previously stated, you must run or rerun the SQL Server 2005 Backward Compatibility installation if it ran before you installed the SQL Server 2000 DTS Designer Components.

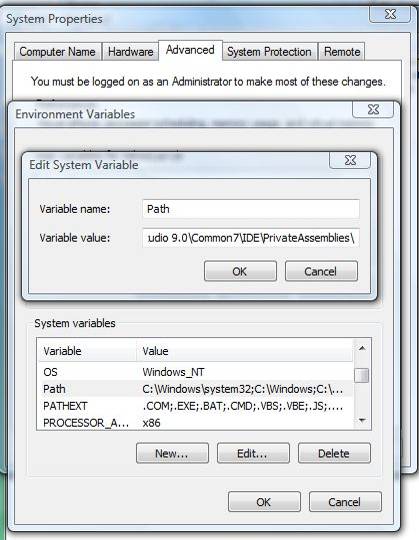
There is one last step required to enable the DTS Designer after the DTS Designer and Backward Compatibility components have been installed in the correct order. If this step is not executed, you will receive an error when attempting to open a package, as Figure 13-17 shows.



**Figure 13-17:** DTS Designer error

You have two options for implementing this last step.

* The first option moves DLL and RLL files from the SQL Server 2000 (80) directory to the SQL Server 2008 (100) directory. For more information, see [How to: Install Support for Data Transformation Services Packages](http://go.microsoft.com/fwlink/?LinkId=133164) in SQL Server 2008 Books Online.
* The second option is to modify the PATH environment variable to put the SQL Server 2000 tools binn directory before the SQL Server 2008 tools binn directory. The PATH environment variable can be modified in the Control Panel, System, Advanced tab, on the Environment Variables, Edit System Variable screen, as Figure 13-18 shows.



**Figure 13-18:** Modifying the PATH environment variable

The following is an example of the PATH variables after SQL Server 2008 and SQL Server DTS 2000 Designer Components have been installed:

%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem; %SYSTEMROOT%\System32\WindowsPowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\100\DTS\Binn\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\VSShell\Common7\IDE\;C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\PrivateAssemblies\**;C:\Program Files\Microsoft SQL Server\80\Tools\Binn\**

Copy the directory in bold and paste it in front of the SQL Server 2008 tools directory path, as follows:

%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem; %SYSTEMROOT%\System32\WindowsPowerShell\v1.0\**;C:\Program Files\Microsoft SQL Server\80\Tools\Binn\**;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\100\DTS\Binn\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\VSShell\Common7\IDE\;C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\PrivateAssemblies\

Note that you will need to restart SQL Server for this PATH variable to take effect.

## Upgrading from SQL Server 2000

The DTS Migration Wizard, available from both BIDS 2008 and SQL Server Management Studio 2008, is the tool you will use for the migration of your existing DTS packages to SSIS 2008. In this section, we will lead you through the DTS Migration Wizard using three examples that map to common DTS package patterns.

### Migrating DTS Packages to SSIS

Note that some DTS features cannot be migrated directly into SSIS, even though SSIS provides a superset of DTS functionality. Therefore, it is important that customers who plan to migrate their DTS packages understand how their existing DTS packages’ tasks and functionality map to SSIS.

One example is the DTS Dynamic Properties task. SSIS configurations and expressions provide developers with greater functionality, but the wizard is unable to migrate this task because there is no single task in SSIS that implements this capability.

In addition, some features are migrated to their SSIS equivalents only when the Copy Column transformation is used. For example, the DTS Transfer Data (DataPump) task cannot be migrated to its SSIS equivalent, the Data Flow task, if a transformation other than Copy Column was used.

The wizard will use a partial migration strategy when it cannot directly migrate an existing DTS task to an SSIS task. This partial migration strategy involves encapsulating these tasks in DTS packages, which in turn can be run by the SSIS Data Transformation Services 2000 Execute Package task.

For more information about how to migrate DTS packages, see [Migrating Data Transformation Services Packages](http://go.microsoft.com/fwlink/?LinkId=132619) in SQL Server 2008 Books Online.

### Comparing DTS and SSIS Functionality

This section provides more information about existing DTS tasks and features and whether they map one-to-one to a SQL Server Integration Services equivalent.

* Table 13-2 lists DTS tasks and features that are migrated to their SSIS equivalent. Note that DTS features followed by an asterisk (\*) have variants that are not directly migrated to SSIS. These variants are covered in either Table 13-3 or Table 13-4.
* Table 13-3 lists DTS tasks that are partially migrated.
* Table 13-4 lists DTS tasks and features that cannot be mapped to an SSIS equivalent.

Table 13-2: DTS Tasks and Features that Are Migrated

|  |  |
| --- | --- |
| **DTS feature** | **SQL Server 2008 SSIS equivalent** |
| ActiveX Script task | ActiveX Script task (deprecated in SQL Server 2008) |
| Bulk Insert task | Bulk Insert task |
| Copy SQL Server Objects task | Transfer SQL Server Objects task |
| Data Mining Prediction task | Data Mining Query task |
| Execute Package task | Execute DTS 2000 Package task |
| Execute Process task | Execute Process task |
| Execute SQL task | Execute SQL task |
| File Transfer Protocol task | FTP task |
| Message Queue task | Message Queue task |
| Send Mail task | Send Mail task |
| Transfer Data (Data Pump) task\* | Data Flow task |
| Transfer Databases task | Transfer Database task |
| Transfer Error Messages task | Transfer Error Messages task |
| Transfer Jobs task | Transfer Jobs task |
| Transfer Logins task | Transfer Logins task |
| Transfer Master Stored Procedures task | Transfer Master Stored Procedures task |
| Connection objects | Connection managers |
| Global variables | Package variables |
| Precedent constraints\* | Precedent constraints |

Table 13-3: DTS Tasks that Are Partially Migrated

|  |  |
| --- | --- |
| **DTS feature** | **Additional information** |
| Custom tasks | Encapsulated in a DTS 2000 Execute Package task. |
| Data Driven Query | Encapsulated in a DTS 2000 Execute Package task. |
| Parallel Data Pump task | Encapsulated in a DTS 2000 Execute Package task. |
| Transfer Data (Data Pump) task | Encapsulated in a DTS 2000 Execute Package task when one or more non-Copy transforms are used.  Later in this chapter we provide an example of how to recreate non-Copy transform logic in SSIS 2008. |

Table 13-4: DTS Tasks and Features that Are Not Migrated

This table lists the DTS features that do not have a direct SSIS equivalent. Each feature is listed along with some additional information about what post-migration steps should occur.

| **DTS feature** | **Additional information** |
| --- | --- |
| Analysis Services Processing task | The SSIS 2008 Analysis Services Processing task and Execute DDL Query tasks are not compatible with SQL Server Analysis Services (SSAS) 2000. These differences prohibit direct migration from a DTS task to an SSIS task. |
| Dynamic Properties task | SSIS does not have a direct task equivalent, although SSIS Configurations and Expressions provide a super-set of the functionality supported by the Dynamic Properties task.  This task is a special case where the wizard creates a Script task that documents (but does not implement) the Dynamic Property task operations.  We show an example of a migrated Dynamic Properties task later in this chapter. |
| Error handling | There is no direct conversion from DTS error handling to SSIS. Note that SSIS Event Handling capabilities support far more robust error handling than in DTS.  For more information, see [Integration Services Event Handlers](http://go.microsoft.com/fwlink/?LinkId=133442) in SQL Server 2008 Books Online. |
| Package logging | There is no direct conversion from DTS package logging to SSIS logging, although SSIS logging is far more robust than DTS logging.  For more information, see [Logging Package Execution](http://go.microsoft.com/fwlink/?LinkId=133443) in SQL Server 2008 Books Online. |
| Package passwords | SSIS includes new encryption features for protecting sensitive data, but DTS package passwords are not migrated.  The Migration Wizard will display a Package Authentication screen where you will need to enter the DTS package passwords order to open the password-protected DTS package. |
| Precedence constraints (workflow properties) | DTS precedence constraints (success, failure, and completion) are migrated to corresponding constraints within SSIS. However, any additional workflow properties are not migrated.  Advanced workflow capabilities can be recreated using SSIS precedence constraints, which include the ability to add expressions.  For more information, see [Precedence Constraints](http://go.microsoft.com/fwlink/?LinkId=133444) in SQL Server 2008 Books Online. |
| Text annotations | There is no direct conversion from DTS text annotations to SSIS annotations. SSIS annotations can be added after the DTS package has been migrated to the SSIS package. |

Examples of the DTS Migration Wizard’s behavior are provided in the next section, where we cover DTS package migration in more detail.

### Running the DTS Migration Wizard

This section guides you through the actual migration process. We cover how you start the DTS Migration Wizard, lead you through the wizard’s screens, and then show how our DTS package examples migrate to SSIS.

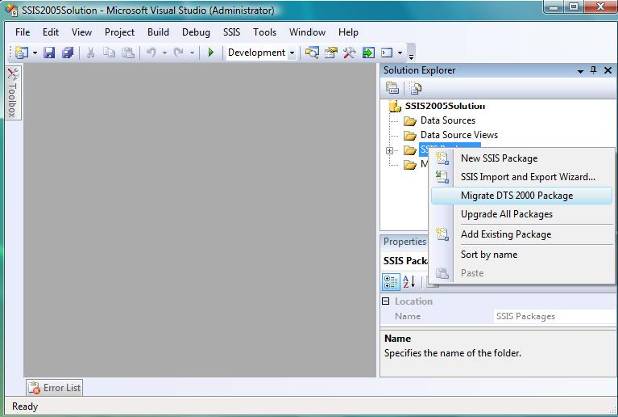
#### Starting the DTS Migration Wizard

The DTS Migration Wizard can be started in one of two ways in an Integration Services project. First, you can select the Migrate DTS 2000 Package menu option on the Project menu in BIDS 2008, as Figure 13-19 shows.



**Figure 13-19:** Starting the DTS Migration Wizard from the Project menu

Second, you can right-click the context menu of the SSIS Packages folder in Solution Explorer, as Figure 13-20 shows.



**Figure 13-20:** Starting the Package Migration Wizard from the SSIS Packages folder in BIDS

In both cases, the DTS Migration Wizard will automatically add the migrated packages to the current Integration Services project when it completes.

Alternatively, the wizard can be started from the Data Transformation Services folder’s right-click context menu, as we saw earlier. Or, you can start the Package Migration Wizard from a command prompt by typing DTSMigrationWizard from the SQL Server 2008 installation folder (typically in the %PROGRAMFILES%\Microsoft SQL Server\100\DTS\binn subdirectory).

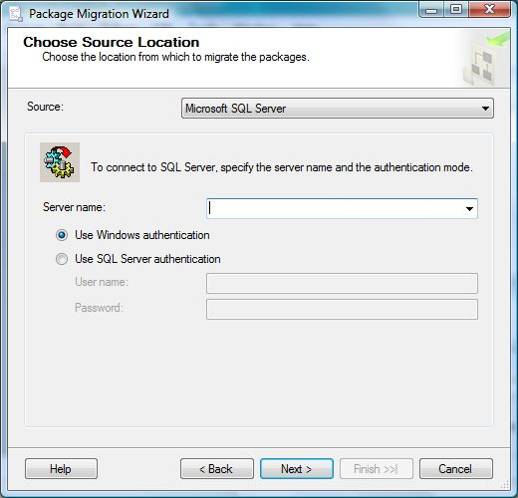
#### Running the DTS Migration Wizard

When the Package Migration Wizard starts, it initially displays a welcome screen, as Figure 13-21 shows. Note that in this example, we are running the Package Migration Wizard from SQL Server Management Studio 2008.

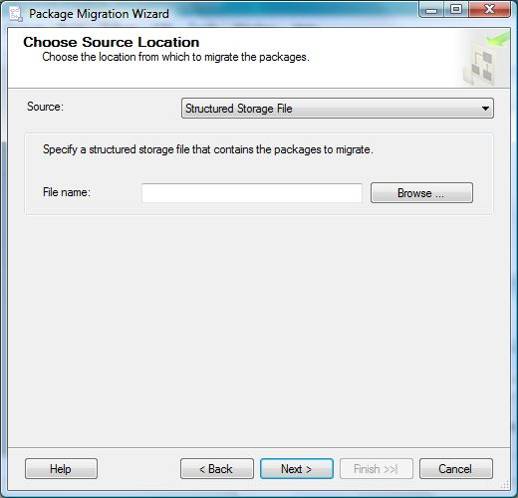


**Figure 13-21:** Package Migration Wizard welcome screen

Click Next to start the wizard. The wizard will prompt for the DTS package’s storage location. Your options are a SQL Server 2000 or SQL Server 2005 msdb database or a .dts file. These options are shown in Figure 13-22 and Figure 13-23, respectively.



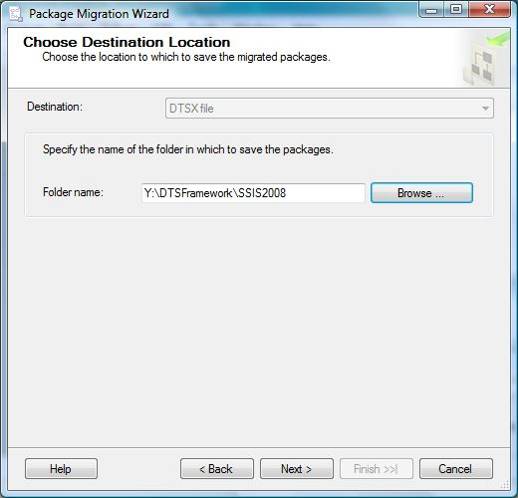
**Figure 13-22:** Package Migration Wizard Choose MSDB Source Location page



**Figure 13-23:** Package Migration Wizard Choose File Source Location page

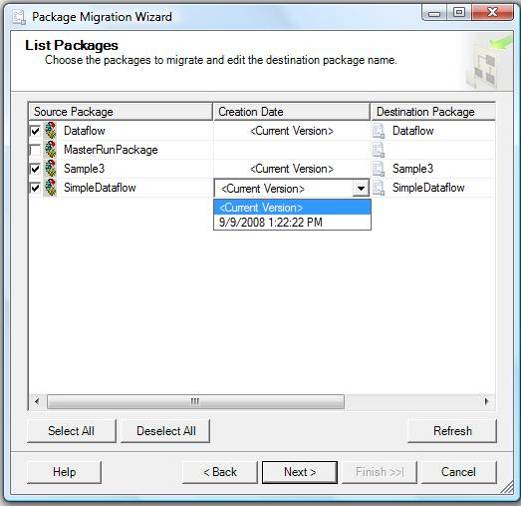
Click Next to continue to the Choose Destination Location page, where you specify the storage location for SSIS packages generated by the wizard. The location should be either a SQL Server 2008 database instance that contains the msdb database or a Windows file system folder.

Figure 13-24 shows the file folder destination option.



**Figure 13-24:** Package Migration Wizard Choose File Destination Location page

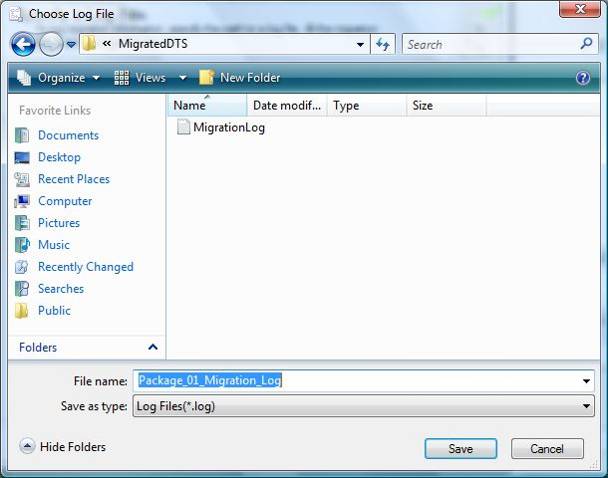
Clicking Next moves you to the List Packages page. By default, the most recent version of each DTS package will be migrated. However, the wizard lets you select a previous version of a package by clicking the Creation Date drop-down list, as Figure 13-25 shows.



**Figure 13-25:** Selecting DTS packages and versions for migration

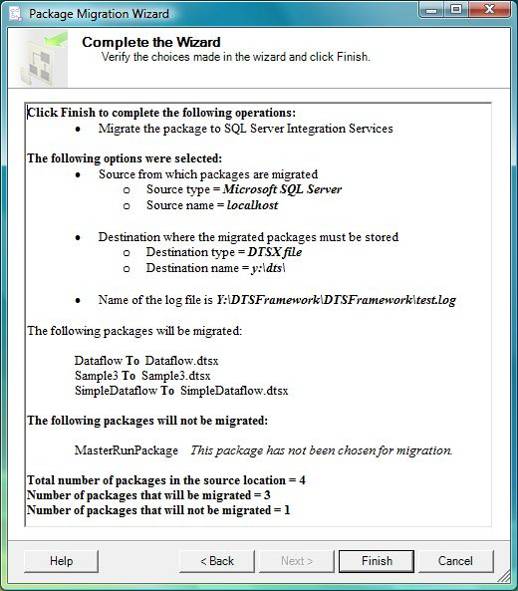
Note that the msdb option will present you with multiple packages, while the file option will prompt for one file at a time. To proceed, select which packages should be migrated and specify prior versions as needed. Note that if a prior version of a DTS package is selected for migration, the resulting SSIS package name will have the version ID appended to its name. The resulting SSIS packages can be renamed as desired after the wizard completes.

The wizard prompts for password information if the DTS package contains passwords. Clicking Next brings you to the Choose Log File screen, which Figure 13-26 shows. Simply specify a directory and log file name, and then click Next to proceed.



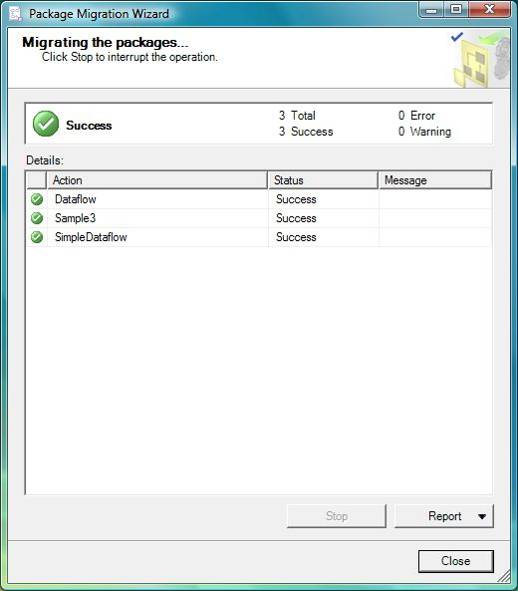
**Figure 13-26:** Specifying a migration log file

Before proceeding with the migration, the wizard will display a summary of the selections made, as Figure 13-27 shows. Review the summary to ensure that all the information is correct, and then click Finish to proceed.



**Figure 13-27:** Package Migration Wizard’s Complete the Wizard screen

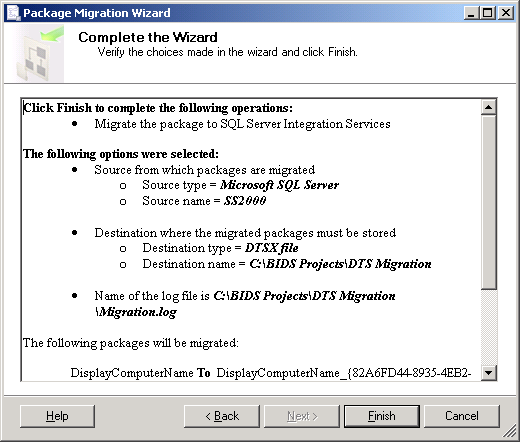
Figure 13-28 shows the summary information for a migration effort that specifies SQL Server as the source location, the Windows file system as the destination location, and the set of three packages that we selected for migration.



**Figure 13-28:** Package Migration Wizard Migrating the Packages page

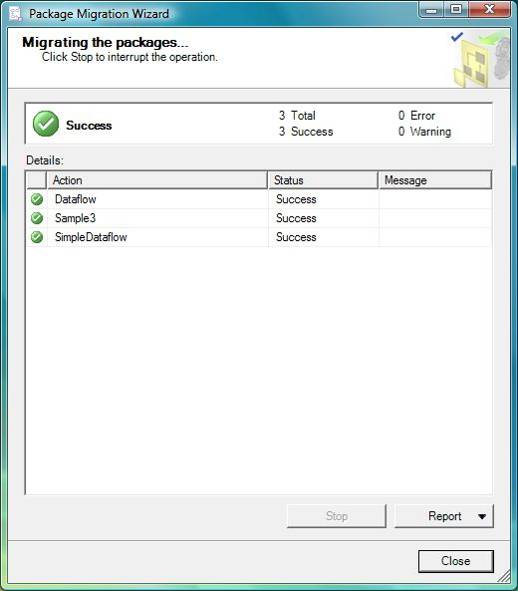
Before proceeding with the migration, the wizard will display a summary of the selections made. Review the summary to ensure that all of the information is correct, and then click Finish to proceed.

Figure 13-29 shows the summary information for a migration effort that specifies SQL Server as the source location, the Windows file system as the destination location, and the set of three packages we selected for migration.



**Figure 13-29:** Package Migration Wizard options summary

When the wizard begins migrating packages, a progress window will show which packages have been migrated and the status of each. Figure 13-30 shows this screen after the wizard completes its migration.



**Figure 13-30:** Package Migration status

After the process is complete, click Report to obtain more detail about the migration or click Close to shut down the Package Migration Wizard.

#### Post-Migration Steps

After the migration process finishes, review every migrated package to determine what additional work needs to be accomplished before the packages can be used. For example:

* Look for partial migrations (that is, SSIS packages that have DTS 2000 Execute Package tasks in their workflow).
* Review ActiveX Script tasks to ensure that the functionality provided by the script will continue to work.
* Review Dynamic Properties tasks, which are converted to Script tasks, and make any changes to the package needed to recreate the functionality provided by the Dynamic Properties tasks.

In the next section, we look at three sample DTS packages and the resulting SSIS packages generated by the DTS Migration Wizard. In addition, we include some post-migration tasks required to duplicate the original DTS package functionality.

### DTS Migration Examples

As we walk through how to run the DTS Migration Wizard, let’s look at the following three examples of DTS packages before and after migration:

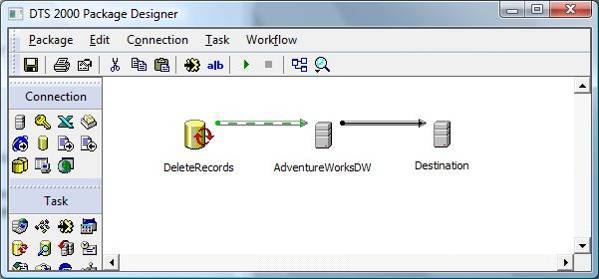
* **Simple Data Transform package.** This package represents many existing DTS packages and contains an Execute SQL task and a simple Data Flow.
* **Data Transform package.** This package is identical to the Simple Data Transform package except that we have added an Uppercase transformation.
* **Support Task package.** This package contains two commonly used support tasks used in DTS: an ActiveX Script task and the Dynamic Properties task.

These examples contain one case in which the DTS Migration Wizard migrates all tasks and two cases where the DTS Migration Wizard implements a partial migration.

#### Example 1: Simple Data Transform Package

The first example presents a common pattern used for loading staging databases. The package first deletes all records in a destination table before loading the table with source data of a similar structure from another database, typically on another server.

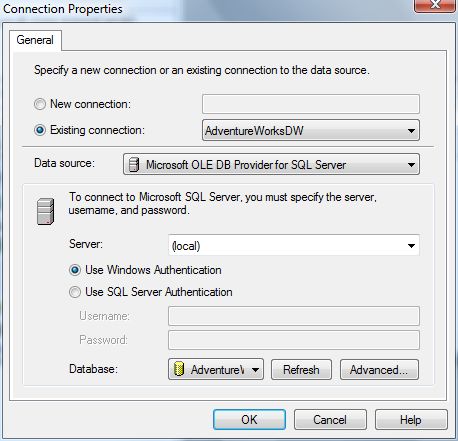
This functionality is implemented by first using an Execute SQL task to delete all records in the destination table and then branching to a DTS Data Transform task to load a destination table with source data, as Figure 13-31 shows.



**Figure 13-31:** Example 1—DTS workflow

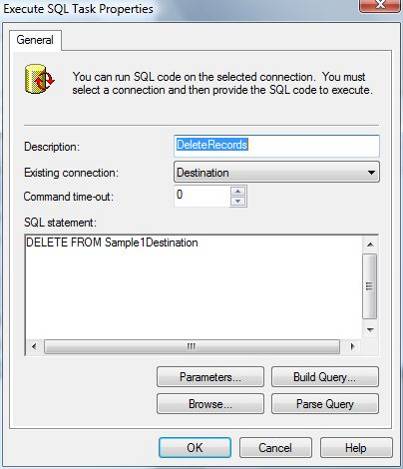
In this example, both the AdventureWorksDW and Destination connections are OLE DB Provider for SQL Server connections.

Figure 13-32 shows the Connection Properties editor for the AdventureWorksDW connection.



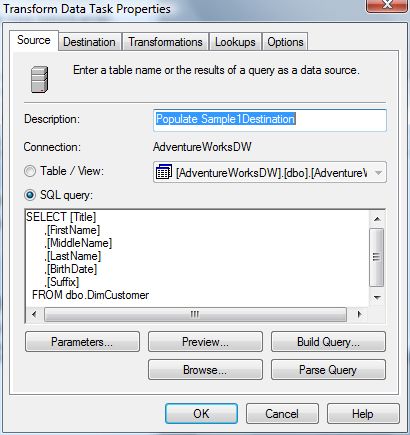
**Figure 13-32:** AdventureWorksDW Connection Properties window

Figure 13-33 shows the Execute SQL Task Properties window for the DeleteRecords task.



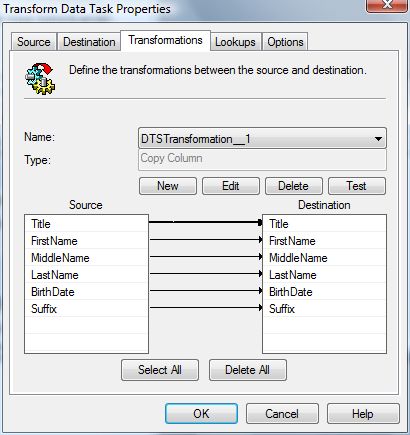
**Figure 13-33:** DeleteRecords Execute SQL Task Properties window

Figure 13-34 shows the Transform Data Task Properties Source tab. Note that we use an SQL SELECT statement to define our Source data set.



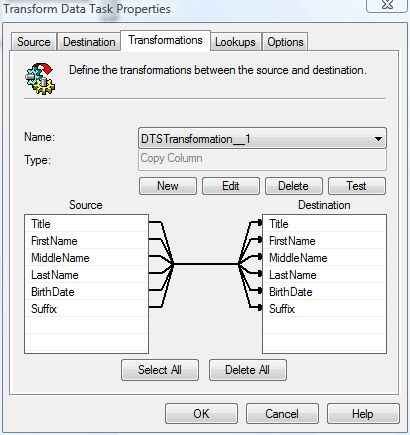
**Figure 13-34:** Transform Data Task Properties, Source tab

Figure 13-35 shows the Transformations tab, in which six Copy Column transformations have been defined. Each Copy Column transformation populates one destination column with one source column of the same name and data type.



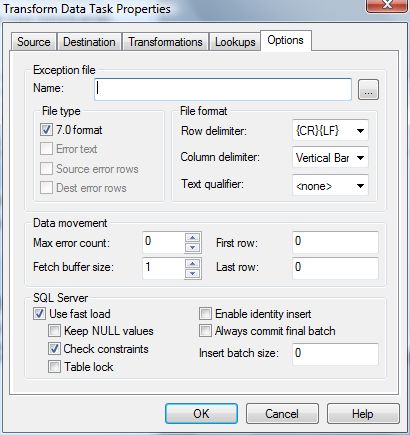
**Figure 13-35:** Transform Data Task Properties, Transformations tab

Note that the DTS Migration Wizard will successfully migrate all combinations of the Copy Column transformations, whether it is one transformation per column or one transformation for all columns, as Figure 13-36 shows.



**Figure 13-36:** Transform Data task will successfully migrate all combinations of the Copy Column transformation

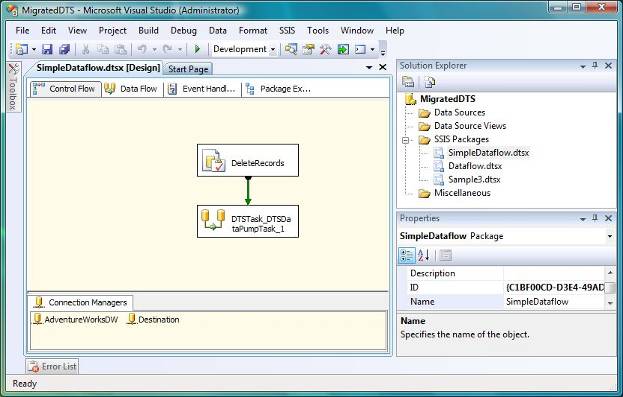
Figure 13-37 shows the Options tab for the Transform Data task. (We will skip the Destination tab, which simply shows the destination table structure.) For this example, the options are set to the default values supplied by the DTS Transform Data task.



**Figure 13-37:** Transform Data Task Properties: Options tab

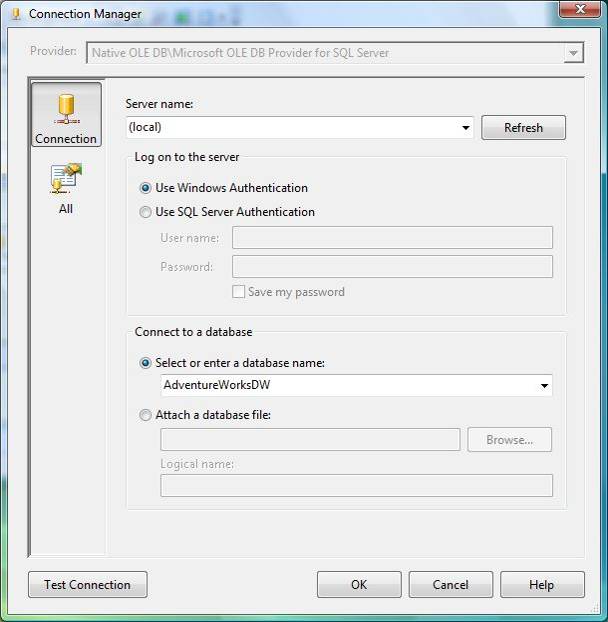
Note that many DTS packages have changed these default values (such as, *Fetch buffer size* and *Insert batch size*) for increased performance.

Example one is now successfully migrated, and Figure 13-38 shows the resulting SSIS package.



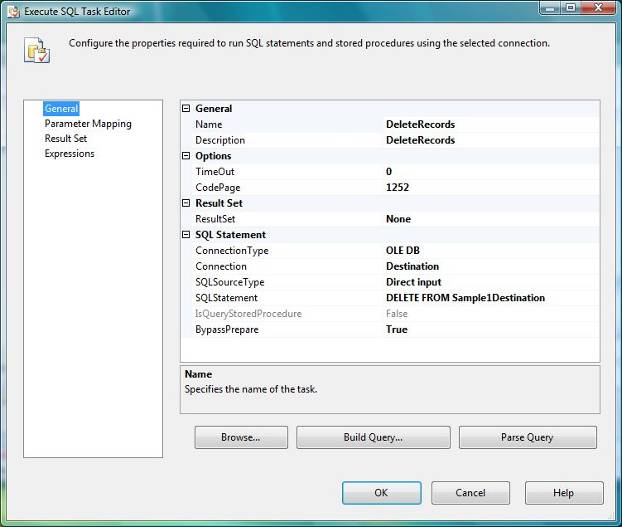
**Figure 13-38:** The SSIS package from example 1

Note that the wizard has migrated the DTS connections to SSIS connections, which are displayed in the Connection Managers tab. Figure 13-39 shows the Connection Manager Properties window for the AdventureWorksDW connection.



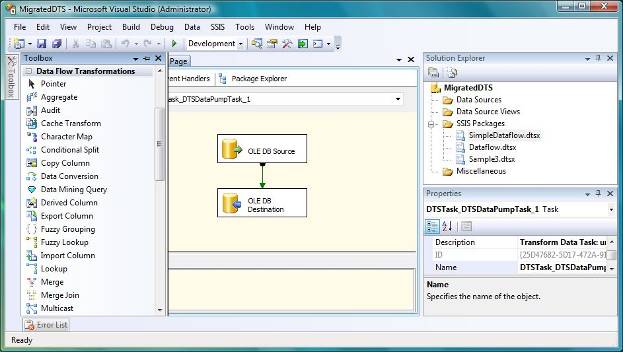
**Figure 13-39:** AdventureWorksDW connection properties editor

The DeletedRecords DTS Execute SQL task was successfully migrated, as Figure 13-40 shows. Note that in this simple case, the DTS and SSIS tasks look very similar. However, SSIS supports advanced capabilities, including the Result Set, which lets you choose where and how you want to store the results of a SELECT statement into SSIS variables.



**Figure 13-40:** SSIS Execute SQL Task Editor

Next, let’s look at the SSIS Data Flow task to which the DTS Data Transform task was successfully migrated, as Figure 13-41 shows. The Data Flow task defines how data moves and how it is transformed between a source and a destination. In this case, columns in a table in the OLE DB source are simply mapped to similar columns in the OLE DB destination.



**Figure 13-41:** Example 1—SSIS Data Flow Task Editor

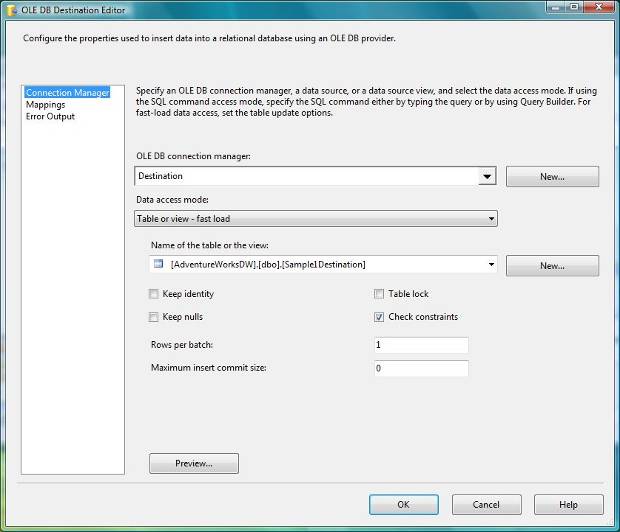
Notice the partial list of Data Flow transformations in the Toolbox to the left. Compare this to the DTS Transform task transformations, and you will start to see the huge jump in functional capabilities between DTS Data Transforms and SSIS Data Flows. We will be using a transformation later for our second example.

The DTS Data Transform Source tab migrates to a Data Flow source in the Data Flow workflow, which Figure 13-42 shows. Notice how the SQL query was successfully migrated from DTS to SSIS.



**Figure 13-42:** Example 1—OLE DB Source Editor

The DTS Data Transform Destination, the Transformation Destination mapping, and the Options tab were migrated to the SSIS OLE DB Data Flow Destination component, as Figure 13-43 shows.



**Figure 13-43:** Example 1—OLE DB Destination Editor

Notice in the OLE DB Destination Editor how information from the DTS Transform Options tab was migrated to the Connection Manager screen. The Transformations tab was migrated to the Mappings screen, as Figure 13-44 shows.

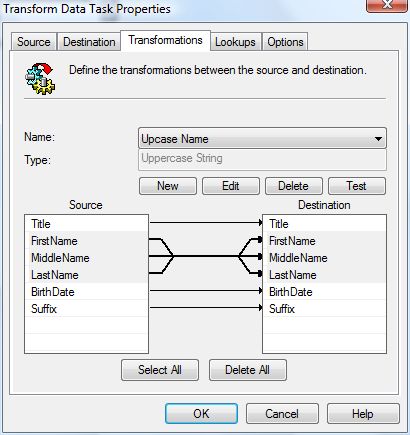


**Figure 13-44:** Example 1—Destination Mappings tab

As noted above, one area in which SSIS provides far more capabilities than DTS is in the transformations and work flow that can occur between sources and destinations. For more information about the Data Flow task, see [Working with Data in Data Flows](http://go.microsoft.com/fwlink/?LinkId=133445) in SQL Server 2008 Books Online.

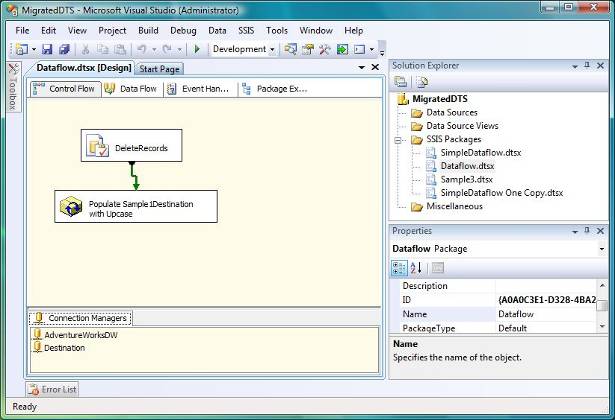
#### Example 2: Adding a “Non-Copy” Transformation

The second example is identical to the first except that instead of using the Copy Column transformation, we are using the Uppercase String transformation for the First Name, Middle Name, and Last Name columns, as Figure 13-45 shows.



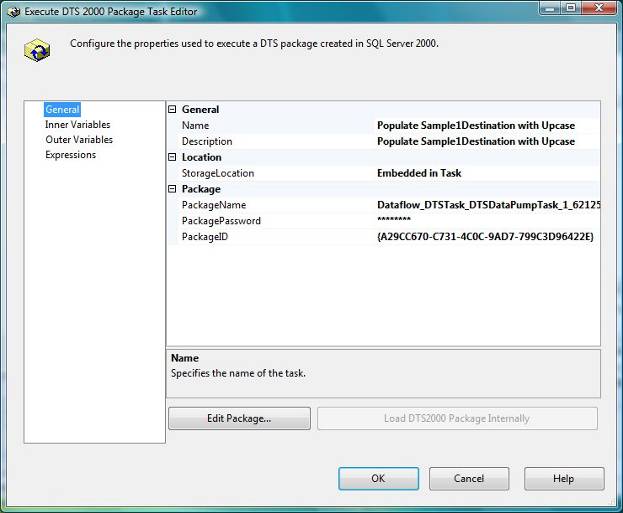
**Figure 13-45:** Adding an Uppercase transformation

Note that this one change results in a partial migration because the DTS Migration Wizard does not support DTS transformations other than the Copy Transform. Figure 13-46 shows the resulting SSIS package.



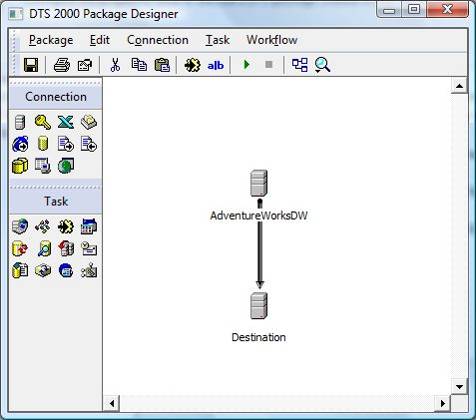
**Figure 13-46:** Example 2—Migration result

The Transform Data task has been preserved by the DTS Migration Wizard by encapsulating it in a DTS 2000 package. Figure 13-47 shows the Execute DTS 2000 Package Task Editor window.



**Figure 13-47:** Example 2—Execute DTS 2000 Package Task Editor containing the Transform Data Flow

Clicking the Edit Package option in the Execute DTS 2000 Package Task Editor displays the non-migrated portion of the DTS example, which Figure 13-48 shows.



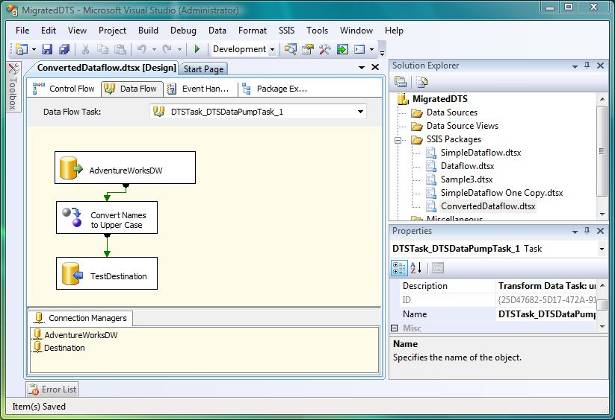
**Figure 13-48:** Example 2—Data Transform

Note that the original DTS Data Transform was preserved after the DTS Migration Wizard determined that it could not migrate the Uppercase transformation to equivalent SSIS functionality. In such cases, you must rewrite the code to leverage SSIS functionality.

Two approaches that you can use to accomplish this are as follows:

* Document the DTS Data Transform source to destination mappings and build the SSIS Data Flow from scratch.
* Take a screen shot of the DTS Data Transform task and then modify the transformations to all use “Copy column.” This way, the DTS Migration Wizard will migrate the task, which then lets you open the Data Flow task and add the required functionality.

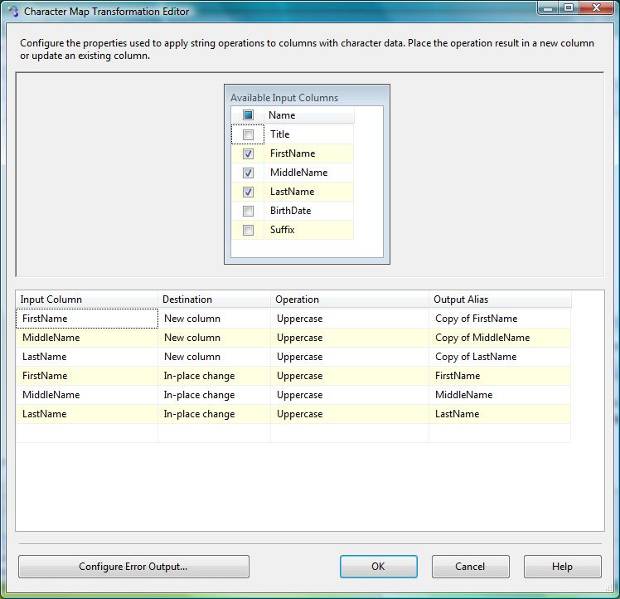
Figure 13-49 shows the Data Flow task with the Uppercase transformation implemented by using the SSIS Character Map transform.



**Figure 13-49:** Example 2—Converted to SSIS functionality

Notice how we also changed the generic component descriptions to more meaningful names. This highlights one benefit of an ETL tool such as SSIS—namely, the visual work flow is easier to understand and maintain than a series of stored procedures and/or Transact-SQL scripts.

The Character Map transformation is one of the many transformations available to an SSIS developer. Figure 13-50 shows how we use the Character Map transform to duplicate the Uppercase functionality implemented in the DTS Transform Data task.



**Figure 13-50:** Example 2—Character Map Transformation Editor

Notice how we have the option of choosing whether we want this transformation to populate a new column or to overwrite the existing column (that is, make an “In-place change”). This starts to highlight the additional capabilities of SSIS, in which you have a data pipeline that is operated on by a series of transformations. You have the ability to add columns, remove columns, multicast to multiple destinations, merge multiple pipelines into one, and so on.

Something to consider when migrating DTS packages to SSIS is that the Data Flow task’s enhanced capabilities over the DTS Data Transform task might be a strong reason to rewrite the DTS package to take full advantage of SSIS capabilities. For example, many intermediate data-staging areas can be eliminated now that developers can implement complex and varied transformations in one SSIS data flow task.

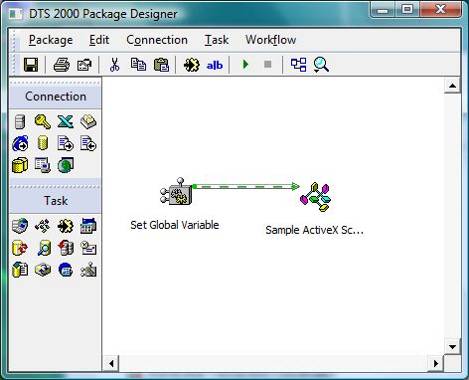
#### Example 3 – Migrating Common DTS Helper Tasks

As we stated earlier, the wizard follows a “best effort” migration process. The Dynamic Properties task is one example of a DTS feature that cannot be directly migrated by the Package Migration Wizard to equivalent SSIS functionality.

The Dynamic Properties task was commonly used by DTS developers to dynamically set DTS connection and global variables at runtime, thus enabling one package to run in multiple environments (such as, development, test, QA, and production) without code changes.

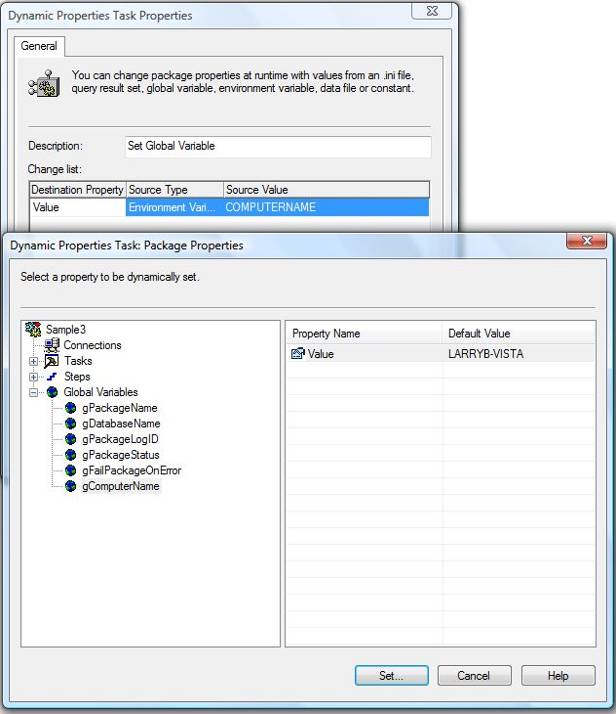
Note that SSIS has enhanced the Dynamic Properties functionality by integrating robust configuration support as well as introducing expressions, which provide unlimited flexibility for developers. However, given this different approach, it is not possible for the wizard to migrate the old functionality to the new. Instead, the wizard creates a Script task “placeholder,” which can be used to implement the functionality in SSIS.

Figure 13-51 shows our third example, which contains a Dynamic Properties task that sets a DTS global variable followed by an ActiveX script that calls Msgbox to display this variables value.



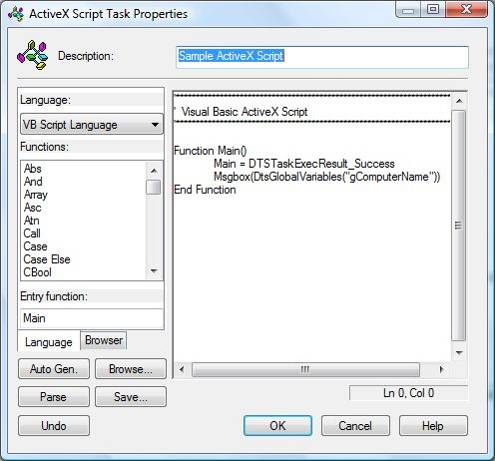
**Figure 13-51:** Example 3—DTS common helper task package

Figure 13-52 shows that we are dynamically setting the global variable gSampleGlobalVariable with the operating system Environment Variable. Note that the Dynamic Property Task: PackageProperties Editor is invoked by selecting Edit in the Dynamic Properties Task Properties Editor.



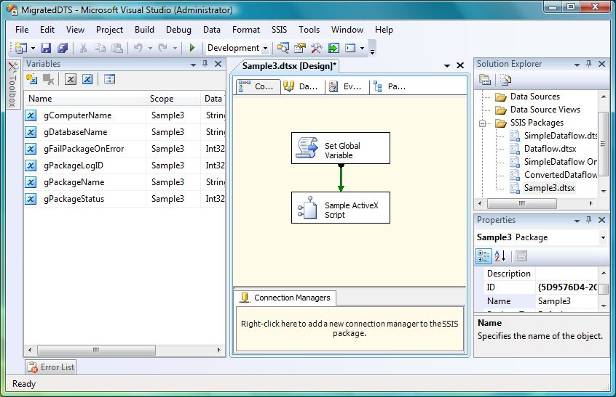
**Figure 13-52:** Dynamic Properties Task Editor windows

Figure 13-53 shows the second task, an ActiveX Script that displays the value of this global variable by using the Msgbox function. This represents another common DTS pattern: DTS ActiveX Script tasks that interface with DTS global variables.



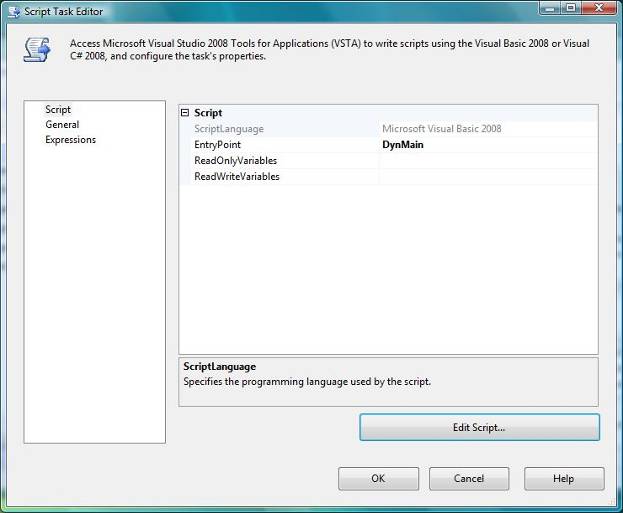
**Figure 13-53:** Example 3—ActiveX Script Task Editor

Figure 13-54 shows the SSIS package created by the Migration Wizard.



**Figure 13-54:** Example 3—Migrated to SSIS

Notice how the Migration Wizard successfully migrated the DTS global variables as well as the ActiveX Script task. Also notice how the Dynamic Properties task was migrated to a Script task. Figure 13-55 shows the Script Task Editor window.



**Figure 13-55:** Example 3—Script Task Editor

Clicking Edit Script will open the Script Task Editor and let you view the migrated results. The code created by the wizard that documents the Dynamic Property operation follows:

Public Sub DynMain()

' Source Type = 3

' Environment variable = COMPUTERNAME

' Destination = 'Global Variables';'gComputerName';'Properties';'Value'

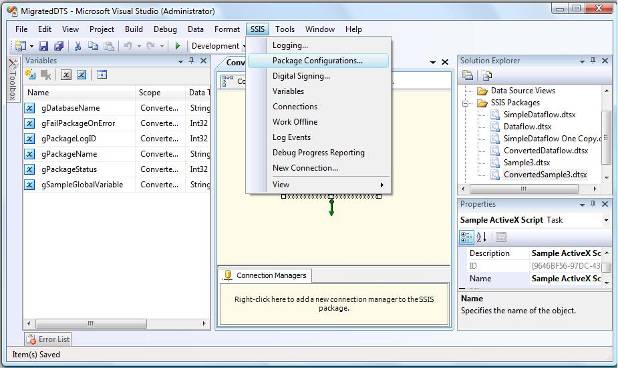
' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dts.TaskResult = ScriptResults.Success

End Sub

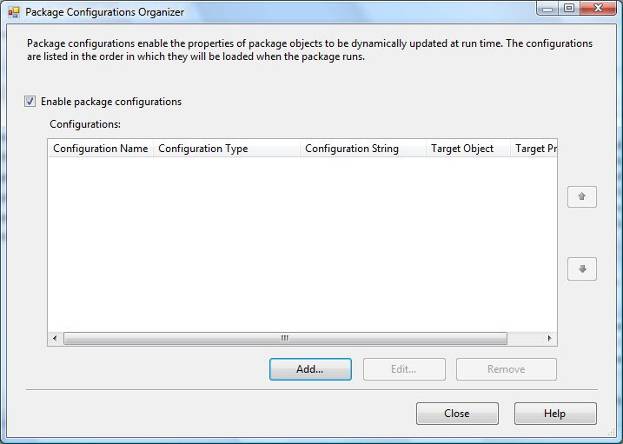
Note how the wizard’s output is not functional; its purpose is to document the behavior of the DTS Dynamic Properties task.

The next step is to use this information to implement the functionality by using SSIS package configurations. Figure 13-56 shows how to access this capability in SSIS.



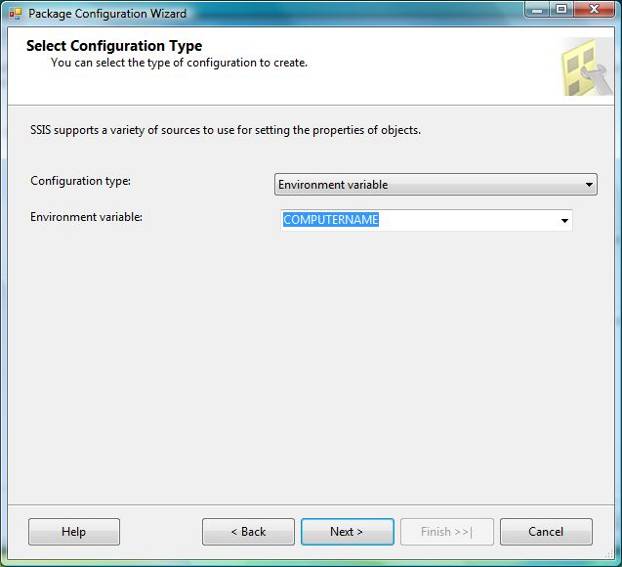
**Figure 13-56:** Example 3—Accessing Package Configurations

Selecting Package Configurations will bring you to the Package Configurations Organizer, which Figure 13-57 shows.



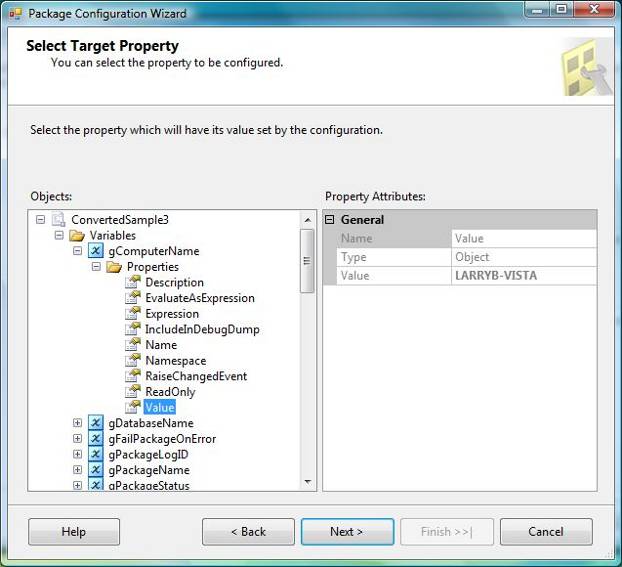
**Figure 13-57:** Package Configurations Organizer

Clicking Add will start the Package Configuration Wizard. Figure 13-58 shows the wizard’s second screen. We start by selecting our configuration source, which in our example is the COMPUTERNAME Environment variable.



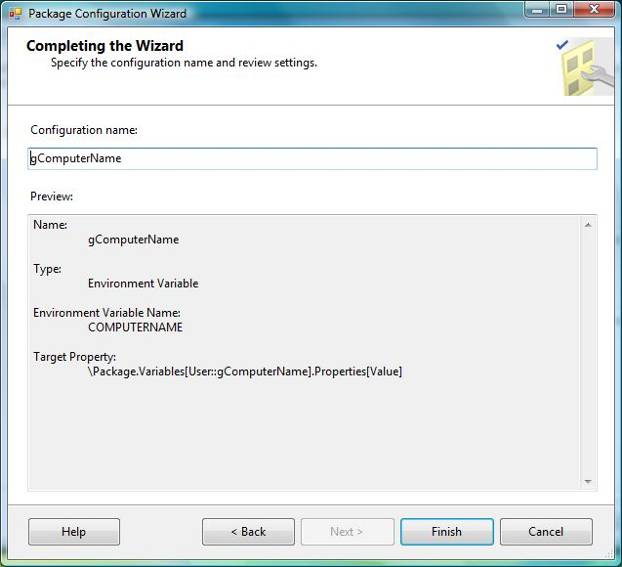
**Figure 13-58:** Package Configuration Wizard: Select Configuration Type screen

Next, we specify the destination to receive this value, which in our example is the gComputerName variable’s value property, as Figure 13-59 shows.



**Figure 13-59:** Package Configurations: Selecting the target property

We then give this configuration a name and click Finish to complete the process, as Figure 13-60 shows.



**Figure 13-60:** Completing the Package Configuration Wizard

Now our SSIS package has the same functional capabilities as our DTS package.

In review, we have covered three DTS package examples that represent common patterns used in DTS. Highlighting all combinations of the wizard’s handling of existing DTS packages is beyond the scope of this chapter. But remember that the DTS Migration Wizard’s “best-case” migration approach means that you must review every SSIS package to ensure that the package functionality is the same after migration.

Note that SSIS by default validates a package when it is opened with the Integration Services Designer. This means that any package that dynamically creates tables in the workflow or initializes Connection Manager values at run time will throw an error because the table has not yet been created or the Connection Manager’s connection properties have not been initialized.

You can resolve cases such as these by setting the DelayValidation property to true on the task or other container object, or by setting the ValidateExternalMetadata property to false on the affected data flow component.

For more information, see [Migrating Data Transformation Packages](http://go.microsoft.com/fwlink/?LinkId=132619) in SQL Server 2008 Books Online.

Also keep in mind the issue noted earlier in the “Upgrade Advisor” section: You will need to delete and recreate ODBC connections after package migration. As mentioned above, this issue has been corrected and will be fixed in SQL Server 2008 SP1.

For more information about DTS package migration issues, see [Known DTS Package Migration Issues](http://go.microsoft.com/fwlink/?LinkId=133165) in SQL Server 2008 Books Online.

## Upgrading from SQL Server 2005

Upgrading SSIS 2005 packages to SSIS 2008 is a far simpler process than DTS package migration. It mainly requires you to convert your existing packages from the SSIS 2005 package format (version 2) to the SSIS 2008 format (version 3).

The SSIS 2008 package format, like the SSIS 2005 package format, is an XML representation of the package definition that can be stored either in SQL Server (the msdb database) or in the Window file system (as a .dtsx file).

SSIS 2005 packages can be upgraded using the following methods:

* In BIDS 2008, right-click an SSIS project’s SSIS Packages folder and select Upgrade All Packages.
* Run the Upgrade Packages option from the SQL Server Management Studio Integration Services right-click context menu.
* Run ssisupgrade.exe from the command line

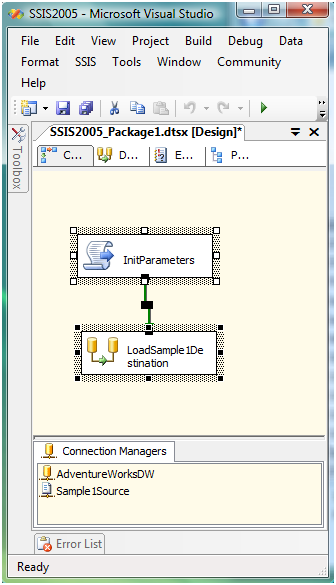
Note that the wizard will also run automatically when you open an SSIS project in BIDS 2008 and one or more SSIS 2005 packages are detected in the project’s directory.

The rest of this section takes you through the Package Upgrade Wizard process for these options. We will use one sample SSIS 2005 package during this walk-through to highlight the areas where SQL Server 2005 and SQL Server 2008 differ.

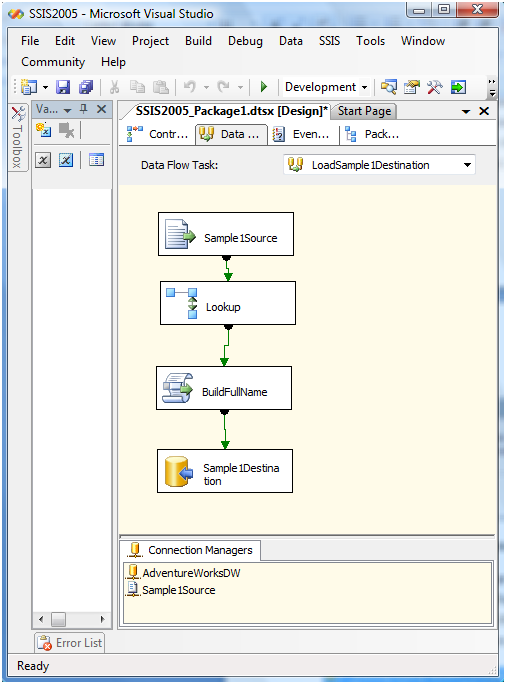
Our sample package reads data from a flat file and loads it into a database. The package includes tasks that will be flagged by the SSIS 2008 upgrade process, including a task script, a data flow component script, and a SQL Server Native Client connection.

One thing missing from this package is a third-party task. Third-party tasks will need to be recompiled with SSIS 2008 before they can run successfully. Note that only a small percentage of existing SSIS packages includes third-party tasks.

Figure 13-61 shows the SSIS\_Package1 task flow, and Figure 13-62 shows the data flow.



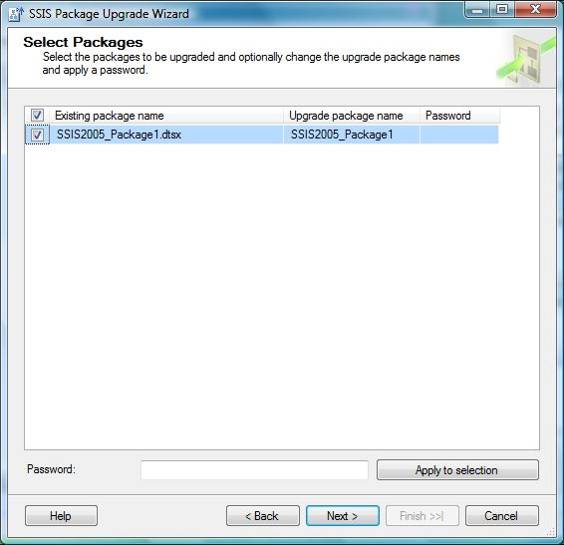
**Figure 13-61:** SSIS Example 3 task flow



**Figure 13-62:** SSIS Example 3 data flow

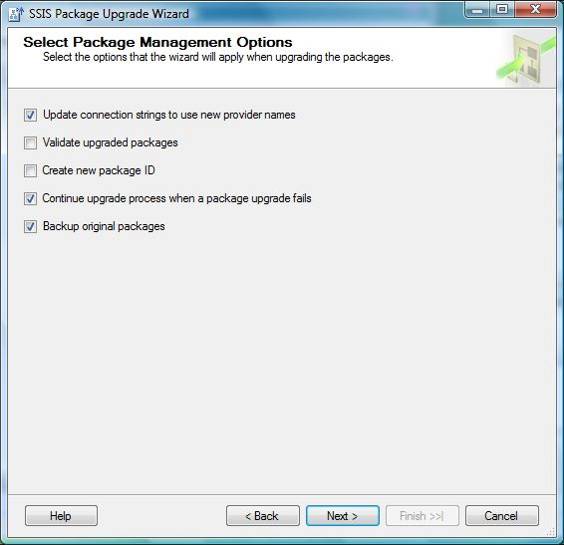
#### Upgrading SSIS Packages by Using BIDS 2008

BIDS 2008 will start the Visual Studio Conversion Wizard when it first opens a solution created in BIDS 2005. After it converts the solution to BIDS 2008, the wizard starts the SSIS Package Upgrade Wizard when it detects an SSIS project, defined in the .dtproj file, in the solution. The wizard first prompts you for the packages in the solution that you are ready to upgrade, as Figure 13-63 shows.



**Figure 13-63:** SQL Server 2008 SSIS Package Upgrade Wizard: Select Packages screen

Note that you will need to enter a password for each SSIS package that is encrypted. Clicking Next moves you to the Select Package Management Options page, which is shown in Figure 13-64.



**Figure 13-64:** The Select Package Management Options screen

These options, when selected, will result in the following actions:

**Update connection strings**. New provider names will be substituted for connection strings if the new name is known to the Upgrade Wizard (such as SQLCLI.1 becomes SQLCLI10.1).

Note that these changes will also have to be made in the data source (.ds) files when the SSIS Connection was created using the New Connection From Data Source option. Otherwise, the upgraded connection strings in the DTSX package files will be overwritten by the non-updated versions in the Data Source file when the package is opened.

Changing connection strings in data source files is discussed below in the “Post-Upgrade Tasks” section.

**Validate upgraded packages.** When set, this option will trigger package validation during the upgrade. The package will not be upgraded if any validation errors occur. This is the same package validation that occurs when you open an SSIS package (with the DelayValidation property set to False) in BIDS 2005 and BIDS 2008.

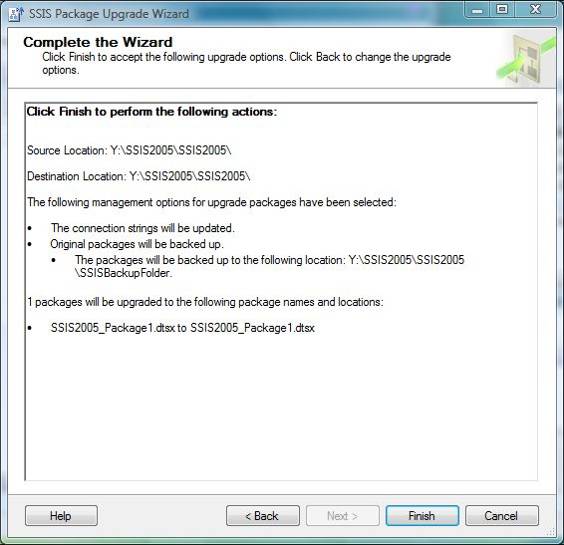
Note that this validation process is optional because it can slow down the upgrade, especially when external servers are defined in Connection Manager connections. Validation of metadata involves querying all data source and destination metadata.

**Create new package ID.** This creates a new GUID for the package. This GUID is accessible through the System::PackageID variable.

**Continue upgrade process when a package fails.** When this option is set, the upgrade process will continue when a package upgrade fails. See the “Post-Upgrade Tasks” section below for information about what can cause package upgrade failures.

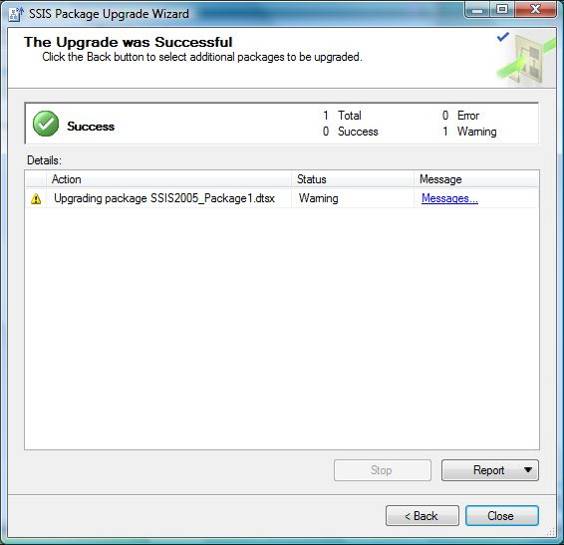
**Back up original packages.** This option appears when you are using the same source and destination locations. It will create a backup folder in the source directory (SSISBackupFolder) and copy the original packages into it to prevent them from being overwritten.

Clicking Next will move you to a review page, which lets you recheck your options before continuing with the upgrade, as Figure 13-65 shows.



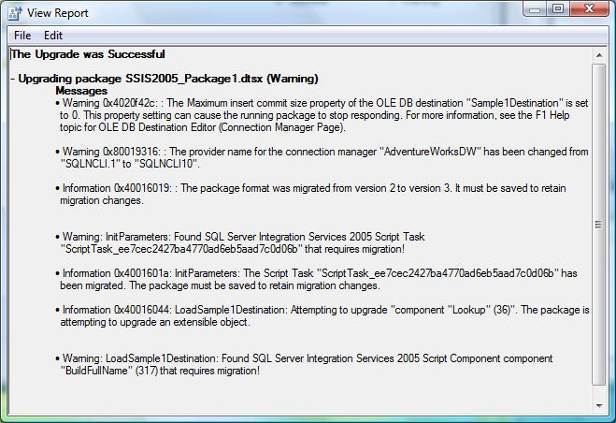
**Figure 13-65:** Package Upgrade Wizard review screen

Clicking Finish will start the upgrade process. After the wizard finishes the upgrade, it provides an upgrade status per package, as Figure 13-66 shows.



**Figure 13-66:** Package Upgrade Wizard status screen

You can either click the Messages link or click the Report button to obtain a report of the informational, warning, and error messages by package that occured during the upgrade. Figure 13-67 shows the message for our example.



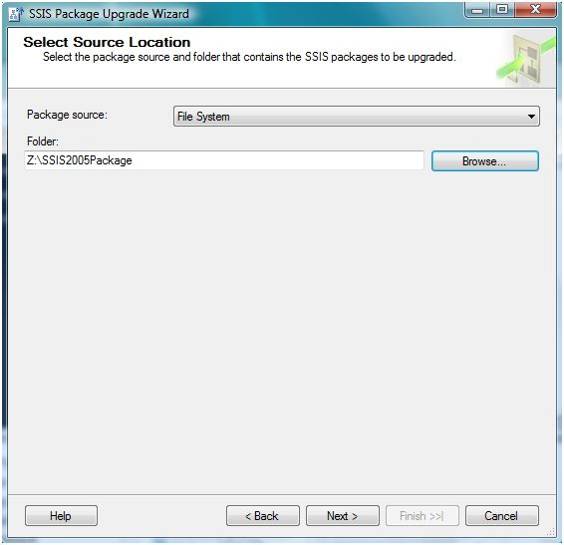
**Figure 13-67:** Package Wizard upgrade report

This example includes most of the features and functionality that the Package Upgrade Wizard will detect and change when upgrading from SQL Server 2005 to SQL Server 2008.

For more information about upgrading SSIS packages, see [Upgrading Integration Services Packages](http://go.microsoft.com/fwlink/?LinkId=133446) in SQL Server 2008 Books Online.

#### Running the Package Upgrade Wizard Directly

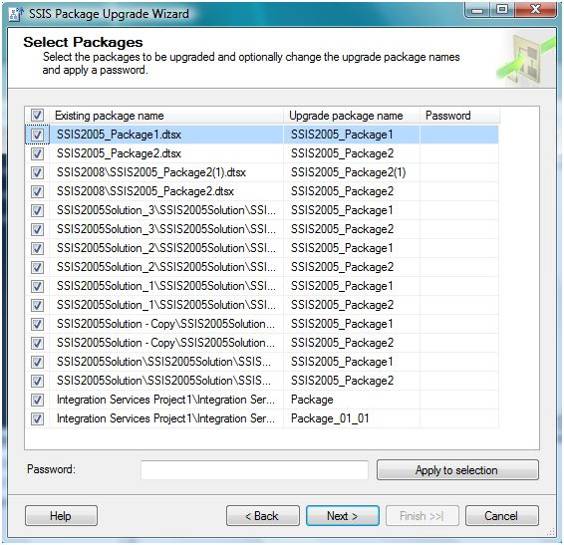
Running the Package Upgrade Wizard outside of BIDS requires you to first specify the location of your SSIS 2005 packages. This is true whether you run ssisupgrade.exe from a command prompt or you invoke the wizard from SSMS. Figure 13-68 shows the Select Package Location screen that follows the optional welcome screen.



**Figure 13-68:** Package Upgrade Wizard: Select Source Location screen

The available SSIS package source options are the file system, the SSIS Package Store, and the Microsoft SQL Server database. Note that the SSIS Package Store is a predefined directory created during SQL Server installation. The default path for the SQL Server 2005 SSIS Package Store is the %PROGRAMFILES%\Microsoft SQL Server\90\DTS\Packages folder.

Specifying a File System Folder and clicking Next will result in the wizard retrieving all SSIS packages from the specified folder and its sub-folders, as Figure 13-69 shows.



**Figure 13-69:** Package Upgrade Wizard: Select Packages screen

Clicking Next brings you to the Select Package Management Options screen, shown earlier in Figure 13-64. Once there, you follow the same steps as if you invoked the wizard from BIDS.

## Post-Upgrade Tasks

There are still some steps remaining after your packages have been migrated from DTS or upgraded from SSIS 2005.

### DTS Post-Migration Tasks

As we discussed earlier, it is a best practice to review (and run) every SSIS package produced by the DTS Migration Wizard to check that the migration process was successful. This section lists specific areas requiring further review and, in some cases, changes to the SSIS package.

In addition, please review the DTS migration examples 2 and 3 presented above for guidance and examples for post-migration steps given common DTS package patterns.

**Analysis Services Processing task.** You will need to re-implement these tasks using the SSIS task equivalents—that is, the Analysis Services Processing task and the Analysis Services Execute DDL task. Note that neither of these tasks supports SSAS 2000, which means that you will need to upgrade to a newer version of SSAS.

**Annotations.** You will need to recreate all DTS annotations as SSIS annotations after the packages have been migrated.

**ActiveX Script Tasks.** Every migrated ActiveX Script task should be reviewed and tested to ensure that it is functioning properly. In addition, analysis to determine the effort to migrate this task to a SSIS Script task will prepare you for day when this feature is not supported in newer versions of SQL Server.

**Data Flows with non-Copy transforms.** This partially migrated task is core to ETL processing and should be considered separately because it is at the core of most DTS packages. For such cases, consider the approach used in “Example 2– Adding a non-Copy Transformation” above. In this example, we removed all non-Copy transforms (which allowed the task to migrate) and then added these capabilities to the migrated SSIS 2008 Data Flow task.

**Dynamic Properties.** You will need to re-implement every Dynamic Properties task, using the SSIS Script task produced during the migration as a guide. This is necessary because the Script task produced by the DTS Migration Wizard does not contain code; rather it documents the Dynamic Properties task’s destination property along with the source used to populate it.

See the “Example 3 – Migrating Common DTS Helper Tasks” section above for an example of how to use SSIS Package Configurations and Expressions to recreate this task’s functionality.

**Non-Migrated Features.** SSIS provides improvements over DTS for core ETL capabilities such as error handling, logging, and workflow precedence. ETL developers will need to recreate these capabilities by using the new and improved SSIS features.

**Package Security.** SSIS has improved upon DTS security. Developers will need to recreate all package security by using the SSIS security features.

**Partially Migrated Tasks.** Any task that is partially migrated will still run in the Execute DTS 2000 Package task. However, you should start preparing for the day when DTS will no longer be supported by newer versions of SQL Server. Note that the amount of rework depends on the task; some will take longer than others.

### SSIS 2005 Post-Upgrade Tasks

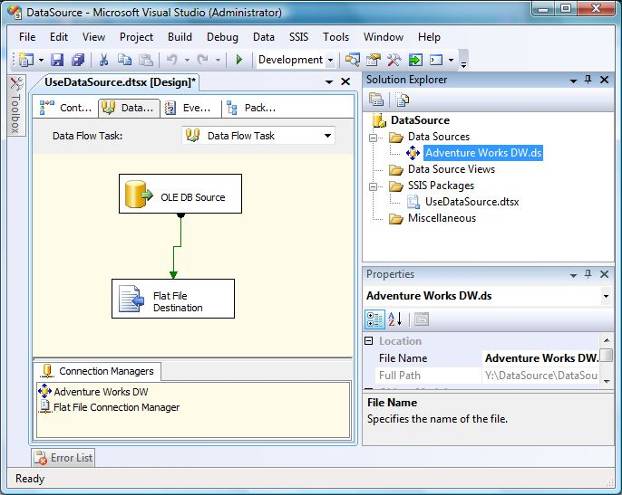
The following is your task list after the Upgrade Wizard has converted your SSIS 2005 packages to the new SSIS 2008 packages format.

#### Upgrading Data Source File Data Provider Names

The Upgrade Wizard will upgrade all the Data Provider names for the SQL Server Native Client (SQLCLI.1 is upgraded to SQLCLI10.1) and SSAS (MSOLAP.3 is upgraded to MSOLAP.4) in the SSIS 2005 package.

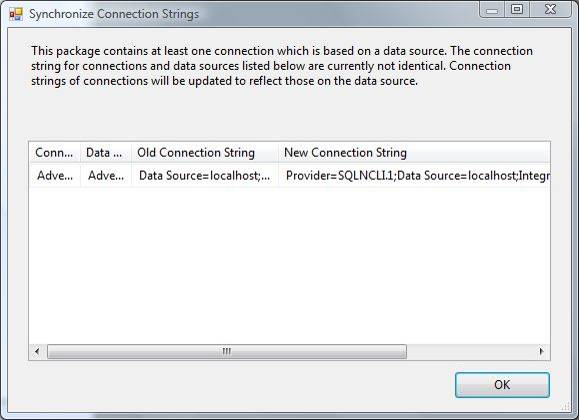
However, the Upgrade Wizard **does not** upgrade these data provider names in data sources (.ds files). This means that you must manually make these changes to each data source after the package is upgraded. Otherwise, BIDS 2008 will replace all of the recently updated Connection Manager’s data provider names with the previous (and invalid) version.

Figure 13-70 shows an example of this. This sample package has one data flow that populates a comma separated value (CSV) file with data from the Adventure Works Data Warehouse’s DimCustomer table. Notice that the Adventure Works DW.ds data source was used to create the Adventure Works DW connection.



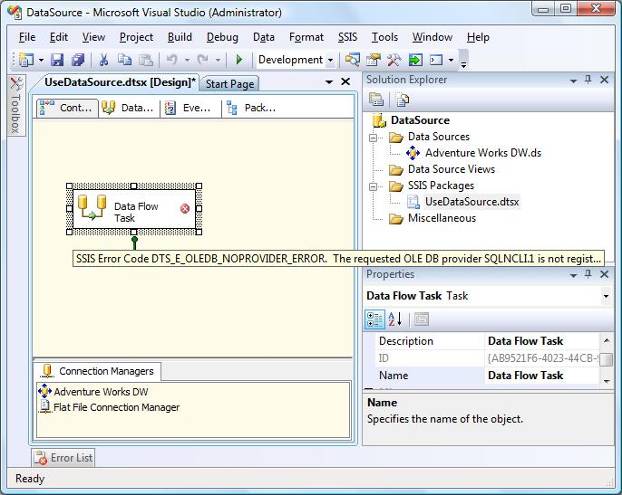
**Figure 13-70:** SSIS package containing a data source connection

The first time that you open this upgraded package, you will receive the message that Figure 13-71 shows.



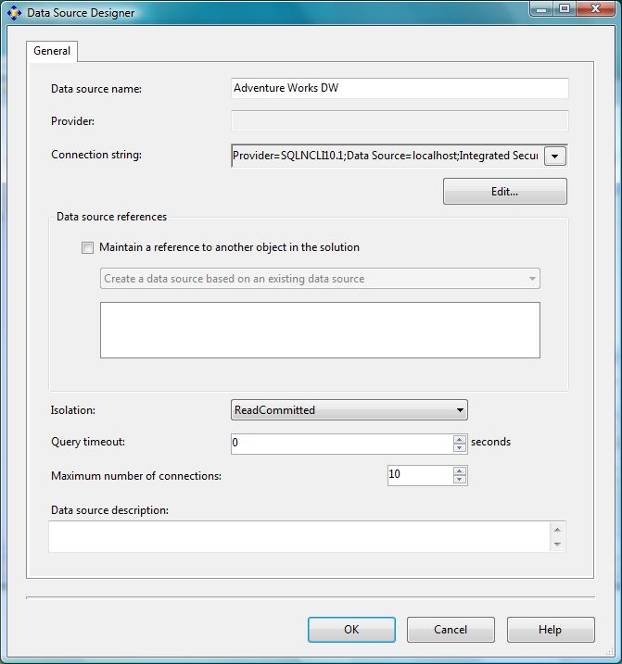
**Figure 13-71:** Synchronize Connection Strings screen

Notice how the new connection string (from the data source) still contains the old provider name, SQLNCLI.1. The end result is that the upgraded data provider name will now be overwritten by the original, which in turn produces an error in the package, as Figure 13-72 shows.



**Figure 13-72:** Error after the connection string synchronization

You will need to manually change the connection string in the Data Source Designer to eliminate this error, as Figure 13-73 shows.



**Figure 13-73:** Changing the data provider name

This corrected connection string is then propagated to the package’s connection, and the data flow is now operational.

An alternative approach is to change the data provider name directly in the data source file by using a text editor before upgrading the package. This approach would eliminate the overwriting of the upgraded data provider name with the previous version.

#### Custom Objects

All existing SSIS 2005 custom objects will need to be modified to work with SSIS 2008. Custom objects let developers extend SSIS capabilities when SSIS script tasks and components are not sufficient to meet the solution’s needs. These custom objects are usually developed by a third party or by an advanced SSIS developer in an IT department.

Developers will need to upgrade their custom objects to SSIS 2008. Third-party suppliers will need to be contacted for an SSIS 2008-compatible version of their custom objects.

Developers who want to upgrade their custom objects should review [Upgrading Custom Objects for SQL Server 2008 Integration Services](http://go.microsoft.com/fwlink/?LinkId=133447) in SQL Server 2008 Books Online.

Developers who want to learn more about custom objects should also review [Extending Packages with Custom Objects](http://go.microsoft.com/fwlink/?LinkId=133448) in SQL Server 2008 Books Online.

#### Script Tasks and Script Components

As we noted early in this chapter, Visual Studio Tools for Applications has replaced Visual Studio for Applications as the scripting environment for the Script task and Script transformation for SSIS 2008. Your task and component scripts will be automatically upgraded to Visual Studio Tools for Applications by the Upgrade Wizard.

However, there are some scenarios in which the Upgrade Wizard will not be able to upgrade the script. To learn more about these conditions, see [Migrating Scripts to VSTA](http://go.microsoft.com/fwlink/?LinkId=133449) in SQL Server 2008 Books Online.

In addition, it is always a best practice to review and test all converted Script tasks and script components to ensure that their behavior in SSIS 2008 is similar to their SSIS 2005 behavior.

#### Lookup Component

The Upgrade Wizard converts the SSIS 2005 Lookup transformation to its SSIS 2008 Lookup transformation equivalent. However, you will need to change your Lookup component to leverage the SSIS 2008 enhanced capabilities, which include using the Lookup Cache Transform and Cache Connection Manager to preload a cache and save it to disk (for use by Lookup Transforms in multiple packages).

For more information about Lookup components’ new capabilities, see [Lookup Transformation](http://go.microsoft.com/fwlink/?LinkId=133450) in SQL Server 2008 Books Online.

For more information about the Cache Transform, see [Cache Transform](http://go.microsoft.com/fwlink/?LinkId=133451) in SQL Server 2008 Books Online.

#### SQL Server Agent Jobs

SQL Server Agent jobs that use an explicit file path to invoke DTEXEC in an Operating System (CmdExec) job step must be changed to reference the SSIS 2008 instance of DTEXEC, as follows:

%PROGRAMFILES%\Microsoft SQL Server\90\DTS\Binn\DTEXEC

Is changed to:

*%PROGRAMFILES%\Microsoft SQL Server\100\DTS\Binn\DTEXEC*

A better solution would be to use the SSIS 2008 Package Job Step. This is the recommended approach and will ensure that you are referencing the correct version of DTEXEC.

## Conclusion

Customers upgrading to SSIS 2008 will follow different paths depending on whether their ETL solutions were developed with DTS or SSIS. However, each of these paths consists of four essential steps: preparation, installation, package upgrade, and post-upgrade tasks.

Microsoft recommends that customers run Upgrade Advisor in the upgrade preparation phase to better understand the effort required to upgrade existing DTS and SSIS packages to SSIS 2008. In addition, customers should be aware of some 64-bit limitations that apply to SSIS 2008 features, including run-time and design-time support for DTS.

The DTS upgrade path requires some additional installation steps if you require legacy support of DTS. In addition, the DTS Package Migration Wizard provides a “best-effort” migration, which means that you must review all SSIS packages generated by the wizard to ensure that they will run correctly. The key point is that DTS is a deprecated feature in SQL Server 2008. So customers with existing DTS packages need to develop a migration plan to prepare for the day when DTS support will no longer be included in SQL Server.

The SSIS 2005-to-SSIS 2008 upgrade path requires less effort than moving from DTS. The Package Upgrade Wizard converts the SSIS 2005 package format (version 2) to the SSIS 2008 package format (version 3) as well as making selected changes after the upgrade completes.

## Additional References

For an up-to-date collection of additional references for upgrading to SSIS 2008, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://msdn.microsoft.com/en-us/library/ms130214.aspx).

For the latest updates to SQL Server 2008 Books Online, see the [Microsoft SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkID=132511).

[SQL Server Integration Services Web site](http://go.microsoft.com/fwlink/?LinkId=133452)

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkID=132512)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkID=132514)

[SQL Server TechCenter](http://go.microsoft.com/fwlink/?LinkID=132515)

# Reporting Services

## Introduction

SQL Server 2000 Reporting Services (SSRS), which shipped in January 2004, provided users with the ability to design and deploy reports within their organizations. The release of this important new component of SQL Server 2000 allowed IT departments, development groups, database administrators, and infrastructure specialists to reduce reporting total cost of ownership (TCO), development cycles, and reliance on non-Microsoft reporting technologies.

With the release of SQL Server 2008, SSRS has been updated with significant new features and ease-of-use improvements. Customers currently using SSRS 2000 or SSRS 2005 need to determine the best approach to upgrading their existing reports and/or their existing environment to SSRS 2008. The options available for upgrading depend on how your SSRS 2000 or SSRS 2005 environment is currently deployed and what level of availability and upgrade testing you need.

### Reporting Services Configurations

When deploying SSRS, many customers chose to install the components using a local instance of SQL Server for housing the report database. This approach is often called a single-server installation. The diagram in Figure 14-1 shows this type of installation.

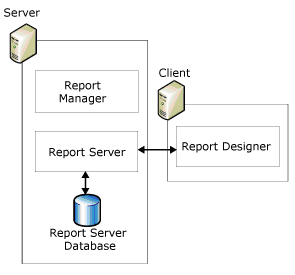


Figure 14-1: Single-server installation of SSRS

SSRS also supports a remote installation in which the report server database is hosted on a different server running SQL Server. This installation provides better scalability, separating report processing and rendering from the database operations needed to manage and maintain report server content, such as reports and snapshots. The diagram in Figure 14-2 shows a remote installation.

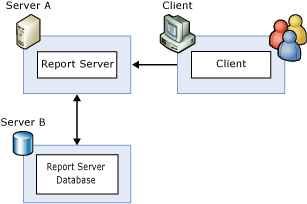
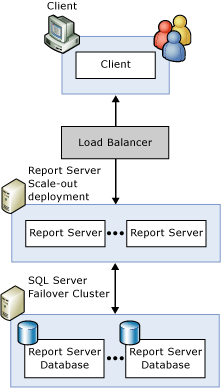


Figure 14-2: Remote installation of SSRS

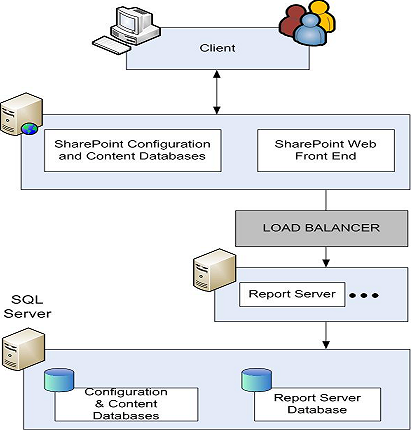
Alternatively, when customers require a highly scalable and available reporting environment, they can deploy SSRS by using scale-out architecture, with or without a clustered environment for the instance of SQL Server that hosts the report server database. Each report server in the scale-out deployment shares a common report server database, providing an extensible architecture in which new report servers can be easily added as the user population and load increases. The diagram in Figure 14-3 shows a scale-out installation.



**Figure 14-3:** Scale-out installation of SSRS

For more information about deploying an SSRS scale-out architecture, see [Reporting Services Scale-Out Architecture](http://go.microsoft.com/fwlink/?LinkId=134491) on SQLCAT.

You can integrate a report server instance within a stand-alone or server-farm deployment of a Microsoft SharePoint product or technology. Figure 14-4 shows a SharePoint server farm with an SSRS installation.



**Figure 14-4:** SharePoint integrated mode deployment

### Reporting Services Editions

SSRS 2008 comes with all editions of SQL Server 2008, with each edition meant to address specific reporting needs throughout an organization:

**Express.** Designed for a single-server configuration with limited data source support, limited management functionality, and no support for ad hoc reporting via Report Builder.

**Workgroup.** Designed for a single-server configuration with limited data source support, full management functionality, support for ad hoc reporting via Report Builder, and support for custom authentication extensions.

**Standard.** Designed for a single-server or remote database configuration with complete data source support, full management capabilities, complete report processing and storage functionality, support for ad hoc reporting via Report Builder, and full support for custom extensions (rendering, data sources, delivery, and authentication).

**Enterprise.** Designed to meet high-volume report processing requirements through scale-out deployment of report servers accessing a centralized (often clustered) report storage environment. Enterprise includes all the features of the Standard edition plus scale-out deployment, data-driven subscriptions, and infinite click-through support for Report Builder.

**Developer.** Designed for developers who want to integrate or extend the report server engine. Developer includes all the features of Enterprise but is licensed for use as a development and test system only.

**Evaluation.** Designed to let organizations evaluate and test SSRS features. Evaluation includes all the features of Enterprise but is valid for only 120-days after installation.

You can find more information about SSRS versions and editions in [How to: Detect Version Information (Reporting Services)](http://go.microsoft.com/fwlink/?LinkId=134569) in SQL Server 2008 Books Online.

And you can review the full list of features available in each edition by reading [Features Supported by the Editions of SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=134493) in SQL Server 2008 Books Online.

It is generally recommended that you upgrade each edition of SSRS 2000 or SSRS 2005 to the same edition of SSRS 2008. However, certain cross-edition upgrades are supported. Specifically, you can upgrade the Standard edition of SSRS 2000 or SSRS 2005 to the Enterprise and Developer edition of SSRS 2008 (as well as to the Standard edition of SSRS 2008).

If you upgrade the Standard edition of SSRS 2000 or SSRS 2005 to the Enterprise or Developer edition of SSRS 2008, you will be able to use some new features, such as data-driven subscriptions, custom security extensions, and scale-out capabilities. However, keep in mind that upgrading to Developer will prevent the upgraded report server from being used as a production server (until it is upgraded to a production version of SSRS 2008).

For complete information about version and edition upgrade paths, see Chapter 1, “Upgrade Planning and Deployment,” and [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkId=134494) in SQL Server 2008 Books Online.

**Note**: This document does not discuss the integration of new features in conjunction with an upgrade of a database instance to SQL Server 2008.

### Upgrade Considerations

When upgrading from SSRS 2000 or SSRS 2005 to SSRS 2008, you should consider the following possible issues.

**SSRS 2000 or SSRS 2005 is always installed as a default instance.** If the report server database resides within the default instance of SQL Server 2000 or SQL Server 2005 on the same server, the relational engine and the report server must be upgraded together (if you are performing an in-place upgrade). In this case, the SQL Server 2008 Setup program upgrades the relational engine first and then upgrades the report server components. When the report server database is upgraded, the Setup program modifies the table structures to reflect the schema needed for SSRS 2008, Most (if not all) schema changes occur when the upgraded Report Server service starts up and runs its Auto-upgrade functionality.

**You can upgrade the Reporting Services component without upgrading the relational engine.** If the report server database resides within a named instance of SQL Server 2005 on the same server or resides on a remote server, you can upgrade the Reporting Services component without upgrading the relational engine. In this case, the On startup of the upgraded report server service Auto-upgrade feature modifies the table structures of the report server database to reflect the schema needed for SSRS 2008. The SQL Server 2008 report server service will continue to connect to the SQL Server 2005 relational engine, with the new database schema in place.

**SSRS includes client and server components.** If you upgrade an SSRS 2005 installation to SSRS 2008 (i.e., upgrade the server components), you should also upgrade the client components used by all report developers. Although it is possible to use the prior version of Report Designer with an SSRS 2008 server, report developers might see a disparity between report preview in Report Designer and how the report is rendered at run time. Note, however, that once you upgrade Report Designer on a given client, you can no longer use it to publish reports to an SSRS 2000 or SSRS 2005 server. Report namespace differences prevent publishing to the prior version of the report server.

**If you have the client components of SSRS 2000 or SSRS 2005 installed on a report server, upgrading the server to SSRS 2008 will remove the previous client components.** If you need the previous SSRS 2000 or SSRS 2005 client components, you can reinstall them after the upgrade is complete.

**If you need to upgrade a scale-out deployment, you must upgrade each SSRS 2000 or SSRS 2005 report server in the scale-out deployment.** You can upgrade the servers in any order, but you should stop all the report servers until all the upgrades are complete. To stop a report server, simply stop Microsoft Internet Information Services (IIS) and the Reporting Services Windows service. When the first report server is upgraded, the shared report server database will be upgraded. After finishing the upgrades, simply restart IIS and the Reporting Services Windows service on each report server.

Here are some general upgrade notes and best practices you should understand before building your upgrade plan for SSRS:

* Cross-version instances of SQL Server 2008 are not supported. Version numbers of the Database Engine, Analysis Services, and Reporting Services components must be the same in an instance of SQL Server 2008.
* Before upgrading SQL Server, enable Windows Authentication for SQL Server Agent and verify the default configuration (for example, that the SQL Server Agent service account is a member of the SQL Server sysadmin group).
* Before upgrading from one edition of SQL Server 2008 to another, verify that the functionality you are currently using is supported in the edition to which you are upgrading. For more information, see the section for your components in [Features Supported by the Editions of SQL Server 2008](http://go.microsoft.com/fwlink/?LinkId=134493) in SQL Server 2008 Books Online.
* Cross-platform upgrade is not supported. You cannot upgrade a 32-bit instance of SQL Server to native 64-bit. However, you can upgrade a 32-bit instance of SQL Server to WOW64, the 32-bit subsystem on a 64-bit server. You can also back up or detach databases from a 32-bit instance of SQL Server and then restore or attach them to an instance of SQL Server (64-bit) if the databases are not published in replication. In this case, you must also recreate any logins and other user objects in the master, msdb, and model system databases. For details about version and edition upgrade paths, see [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkId=134494) in SQL Server 2008 Books Online.
* To upgrade to SQL Server 2008, you must be running a supported operating system. You can review the hardware and software requirements for SQL Server 2008 by reading [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132172) in SQL Server 2008 Books Online.
* The upgrade will be blocked if there is a pending restart.
* The upgrade will be blocked if the Windows Installer service is not running.
* The upgrade will be blocked if performance counters are corrupt.
* To upgrade an instance of SQL Server to a SQL Server failover cluster, the instance being upgraded must be a failover cluster. To upgrade a stand-alone instance of SQL Server to a SQL Server failover cluster, install a new SQL Server failover cluster, and then move user databases from the standalone instance by using the Copy Database Wizard. For more information about upgrading a cluster, see [How to: Upgrade a SQL Server Failover Cluster Instance (Setup)](http://go.microsoft.com/fwlink/?LinkID=132450) in SQL Server 2008 Books Online. For more information about database migration, see [Using the Copy Database Wizard](http://go.microsoft.com/fwlink/?LinkId=134497) in SQL Server 2008 Books Online.
* To upgrade SQL Server 2005 to SQL Server 2008 on a computer that is running Windows Server 2008, you must be running SQL Server 2005 Service Pack 2 (SP2). SQL Server 2005 SP1 is not a supported upgrade scenario.

### In-Place Upgrade vs. Side-by-Side Upgrade

You can upgrade SSRS 2000 or SSRS 2005 installations to SSRS 2008 in one of two ways: through an in-place upgrade (supported by setup) or migration (installing a clean 2008 instance and then moving data / metadata from 2000/2005 to 2008).

#### In-Place Upgrade

With an in-place upgrade, SSRS 2000 or SSRS 2005 is removed and replaced by SSRS 2008. During the upgrade process, the SSRS databases are upgraded, and users will not be able to access SSRS 2000 or SSRS 2005 reports. After the in-place upgrade is complete, only SSRS will remain. With an in-place upgrade, you test SSRS 2008 after removing the previous SSRS version.

An in-place upgrade is an all-or-nothing approach; if an in-place upgrade fails, you cannot quickly roll back to the SSRS 2000 or SSRS 2005 environment after the Setup program finishes the upgrade (there is a go/no-go point within the Setup program before which you can simply cancel the upgrade). To roll back to your previous SSRS environment after an upgrade to SSRS 2008 is completed, you need to uninstall SSRS 2008, reboot, reinstall SSRS 2000 or SSRS 2005, and then restore the SSRS 2000 or SSRS 2005 data and configuration files. Downtime in the event of upgrade problems can be significant.

#### Side-by-Side Upgrade

With a side-by-side upgrade, you install an instance of SSRS 2008 alongside SSRS 2000 or SSRS 2005, which remains until uninstalled. During the upgrade process, users can continue to access the SSRS 2000 or SSRS 2005 reports (unaffected by the upgrade process), but performance might be slower.

After SSRS 2008 is fully tested, you can uninstall your previous SSRS version.

Note: With a side-by-side upgrade, you can either use a copy of the existing report server database for the new installation or redeploy reports and recreate server settings on a new server. You can find more information regarding this at [How to: Migrate a Reporting Services Installation](ttp://go.microsoft.com/fwlink/?LinkId=134498) in SQL Server 2008 Books Online.

**Important:** The side-by-side upgrade option provides for greater availability during the upgrade process, simplifies rollback (if it is required), and results in simpler testing scenarios because both versions are available at the same time.

Table 14-1 shows which upgrade option can be applied to each of the SSRS 2000 and SSRS 2005 configurations described earlier in this chapter. Note that you can use these options regardless of which edition of SSRS 2000 or SSRS 2005 is in place.

Table 14-1: Upgrade Options for SSRS 2000 and SSRS 2005

|  |  |  |
| --- | --- | --- |
| **Reporting Services 2000/2005 Configuration** | **In-Place Upgrade?** | **Side-by-Side Upgrade?** |
| Single-server installation (2000/2005) | Yes | Yes |
| Remote catalog installation on SQL Server 2000 | No | Yes |
| Remote catalog installation on SQL Server 2005 | Yes | Yes |
| Scale-out installation (2000/2005) | Yes | Yes |

### Preparing to Upgrade

Before beginning an in-place upgrade of SSRS, take steps to ensure that a failed upgrade can be rolled back. Although the in-place upgrade process has been designed and tested to handle almost all situations, unforeseen problems might occur and result in a failed upgrade. In extreme cases, a failed upgrade might even result in an unusable SSRS 2000 or SSRS 2005 installation. Thus, planning for a failed upgrade process is critical.

A side-by-side upgrade of an existing SSRS 2000 or SSRS 2005 installation to SSRS 2008 should not encounter the same types of problems that can affect an in-place upgrade. However, you should follow the same steps because the files generated by the steps we cover in this section will be needed for the upgrade process.

If a failed in-place upgrade occurs, in many cases the easiest resolution is to reinstall SSRS 2000 or SSRS 2005 and restore the installation to its state before the upgrade process was started. To ensure that all the data and configuration files needed to restore the existing installation are available, complete the following steps before the upgrade process begins.

1. Verify that SQL Server 2008 hardware and software requirements are met. If you do not meet these requirements, the System Configuration Checker (SCC) portion of the SQL Server Setup program will not permit Setup to continue.
2. Run SQL Server 2008 Upgrade Advisor to analyze installed SQL Server 2000 or SQL Server 2005 relational engine components; Chapter 1, “Upgrade Planning and Deployment,” describes how to run this valuable tool. Then, review the generated report to verify that you have addressed all issues that must be resolved before the upgrade and that you understand the upgrade issues that you must resolve after Setup completes.
3. Back up the report server’s symmetrical encryption key by using the SSRS 2000 or SSRS 2005 rskeymgmt utility. This command-line utility is used to extract or restore the encryption key used by SSRS to store sensitive data within the report server database. This utility is typically found in the <Drive:>*\Program Files\Microsoft SQL Server\80\Tools\Binn\* directoryif you are running SSRS 2000 or in the <Drive :>*\Program Files\Microsoft SQL Server\90\Tools\Binn\* directory if you are running SSRS 2005. To use this utility, simply open a command line, change to this directory, and issue the following command:

rskeymgmt –e –f*<File>* –p*<Password>*

Replace the *<File>* parameter with a valid file specification. This file will contain the symmetric key information. Also, replace the *<Password>* parameter with a password, which will be used to encrypt the symmetric key before it is stored in the file. For more information about the rskeymgmt utility, see [rskeymgmt Utility](http://go.microsoft.com/fwlink/?LinkId=134500) in SQL Server 2008 Books Online.

1. Back up the report server’s databases by using any supported method for backing up a SQL Server database. For a default installation of SSRS 2000 or SSRS 2005, make sure that the ReportServer and ReportServerTempDB databases have been backed up.
2. Back up critical configuration files related to SSRS 2000 or SSRS 2005, including the configuration files discussed in the next section of this chapter.

### Important Reporting Services 2000 Configuration Files

SSRS stores component information in the registry and in configuration files that are copied to the file system during setup. Configuration files contain a combination of internal-use-only and user-defined values. User-defined values are specified through Setup, the configuration tools, the command-line utilities, and by manually editing the configuration files.

Modifying the configuration files is necessary only if you are adding or configuring advanced settings. Configuration settings are specified as either XML elements or attributes. If you understand XML and configuration files, you can use a text or code editor to modify user-definable settings. For more information about how to modify a configuration file or to learn more about how the report server reads new and updated configuration settings, see [How to: Modify a Reporting Services Configuration File](http://go.microsoft.com/fwlink/?LinkId=134499) in SQL Server 2008 Books Online.

**Note:** In previous releases, Report Manager had its own configuration file named RSWebApplication.config. That file is now obsolete. If you upgraded from a previous installation, the file will not be deleted, but the report server will not read any settings from it. If the file exists on your computer, you should delete it. In SQL Server 2008, all Report Manager configuration settings are stored in and read from the RSReportServer.config file. To review a list of which settings were deleted or moved, see [Breaking Changes in SQL Server Reporting Services](http://go.microsoft.com/fwlink/?LinkId=133940) in SQL Server 2008 Books Online.

### Storing Configuration Settings

Table 14-2 describes where SSRS configuration settings are stored. Most configuration settings are stored in configuration files included with SSRS. By default, the installation directory is <drive>:\Program Files\Microsoft SQL Server\MSSQL.*n*.

Table 14-2: Configuration Files and Their Location

| **File** | **Description** | **Location** |
| --- | --- | --- |
| RSReportServer.config | Stores configuration settings for feature areas of the Report Server service: Report Manager, the Report Server Web service, and background processing. For more information about service features, see [Service Architecture (Reporting Services)](http://go.microsoft.com/fwlink/?LinkId=134504) in SQL Server 2008 Books Online. For more information about each setting, see [RSReportServer Configuration File](http://technet.microsoft.com/en-us/library/ms157273.aspx) in SQL Server 2008 Books Online. | <Installation directory> \Reporting Services \ReportServer |
| RSSrvPolicy.config | Stores the code access security policies for the server extensions. For more information about this file, see [Using Reporting Services Security Policy Files](http://go.microsoft.com/fwlink/?LinkId=134506) in SQL Server 2008 Books Online. | <Installation directory> \Reporting Services \ReportServer |
| RSMgrPolicy.config | Stores the code access security policies for Report Manager. For more information about this file, see [Using Reporting Services Security Policy Files](http://go.microsoft.com/fwlink/?LinkId=134508) in SQL Server 2008 Books Online. | <Installation directory> \Reporting Services \ReportManager |
| Web.config for the Report Server Web service | Includes only those settings required for ASP.NET. | <Installation directory> \Reporting Services \ReportServer |
| Web.config for Report Manager | Includes only those settings required for ASP.NET. | <Installation directory> \Reporting Services \ReportManager |
| ReportingServicesService.exe.config | Stores configuration settings that specify the trace levels and logging options for the Report Server service. For more information about the elements in this file, see [RSReportServer Configuration File](http://technet.microsoft.com/en-us/library/ms157273.aspx) in SQL Server 2008 Books Online. | <Installation directory> \Reporting Services \ReportServer \Bin |
| Registry settings | Stores configuration state and other settings used to uninstall SSRS. If you are troubleshooting an installation or configuration problem, you can view these settings to get information about how the report server is configured.  Do not modify these settings directly because this can invalidate your installation. | HKEY\_LOCAL\_MACHINE \SOFTWARE \Microsoft \Microsoft SQL Server \<InstanceID> \Setup  - And -  HKEY\_LOCAL\_MACHINE \SOFTWARE \Microsoft \Microsoft SQL Server \Reporting Services |
| RSReportDesigner.config | Stores configuration settings for Report Designer. For more information, see [RSReportServer Configuration File](http://technet.microsoft.com/en-us/library/ms157273.aspx) in SQL Server 2008 Books Online. | <drive>:\Program Files \Microsoft Visual Studio 8 \Common7 \IDE \PrivateAssemblies. |
| RSPreviewPolicy.config | Stores the code access security policies for the server extensions used during report preview. For more information about this file, see [Using Reporting Services Security Policy Files](http://go.microsoft.com/fwlink/?LinkId=134508) in SQL Server 2008 Books Online. | <drive>:\Program Files \Microsoft SQL Server \100 \Tools \ReportDesigner |

### Deprecated Features

This section describes any SSRS features that have been deprecated and will not be supported in future releases. For information about deprecated functionality, see [Deprecated Features in SQL Server Reporting Services](http://go.microsoft.com/fwlink/?LinkId=133936) in SQL Server 2008 Books Online.

#### Internet Explorer 5.5 Support

Deprecation of Microsoft Internet Explorer 5.5 support was announced in SQL Server 2005 SP2. This version of Internet Explorer is not supported in the SQL Server 2008 release of SSRS.

### Discontinued Functionality

This section covers the SSRS features no longer available in SQL Server 2008. For more information about any of these features, see [Discontinued Functionality in SQL Server 2008 Reporting Services](http://go.microsoft.com/fwlink/?LinkId=134512) in SQL Server 2008 Books Online.

We do not include announcements about discontinued support for specific versions of the operating system or IIS. For information about system prerequisites, see [Hardware and Software Requirements for Installing SQL Server 2008](http://go.microsoft.com/fwlink/?LinkID=132172) in SQL Server 2008 Books Online.

#### Manual Upgrade for Report Server Database

Starting in SQL Server 2005 SP2, the Report Server service is able to auto-detect the version of the report server database and upgrade it to the schema that matches the version of the current report server instance. When an older version of the database is detected, you are automatically upgrade. If you proceed to upgrade it, the schema is updated to the new format, and you cannot roll it back to a previous format.

Because of the new auto-upgrade feature, you no longer need to create a database upgrade script and there is no need for a manual upgrade option in the Reporting Services Configuration tool. In this release of SQL Server 2008, the button to create the script and the button that performs the upgrade action are removed from the Database Setup page in the Reporting Services Configuration tool.

#### SQL Server 2000 Report Server Web Service Endpoint

The SQL Server 2000 Report Server Web service endpoint is discontinued in SQL Server 2008. For information about current endpoints, see [Report Server Web Service Endpoints](http://go.microsoft.com/fwlink/?LinkId=134513) in SQL Server 2008 Books Online.

#### HTML OWC Rendering Extension

The HTML with Office Web Components (OWC) rendering extension is discontinued in SQL Server 2008.

#### HTML 3.2 Rendering Extension

The HTML 3.2 format in the HTML rendering extension is discontinued in this release. The rendering extension is no longer included in an SSRS installation.

#### Report Builder Runs in Full Trust Mode Only

For SSRS in native mode, running Report Builder in partial trust mode is discontinued in SQL Server 2008. SSRS supports running Report Builder in full trust mode only. This is true for both native and SharePoint mode.

#### Surface Area Configuration Tool

The Surface Area Configuration Tool is discontinued for SQL Server 2008. For more information, see [Backward Compatibility](http://go.microsoft.com/fwlink/?LinkID=132171) in SQL Server 2008 Books Online.

### Breaking Changes

Breaking changes in SSRS are those that might break applications, scripts, or functionalities that are based on earlier versions of SQL Server. You might encounter these issues when you upgrade or in custom scripts or reports. SQL Server 2008 Upgrade Advisor identifies many breaking changes; Chapter 1, “Upgrade Planning and Deployment,” and [Using Upgrade Advisor to Prepare for Upgrades](http://go.microsoft.com/fwlink/?LinkID=132624), in SQL Server 2008 Books Online, describe how to run this tool to find and fix problems before the upgrade.

For complete information about breaking changes in SSRS 2008, see [Breaking Changes in SQL Server 2008 Reporting Services](http://go.microsoft.com/fwlink/?LinkId=134515) in SQL Server 2008 Books Online.

#### Report Server Breaking Changes

**IIS and ASP.NET.** SSRS no longer depends on IIS to provide access to the SOAP endpoint. URLs no longer include Web sites in IIS. Reporting Services uses HTTP.SYS directly to listen for requests on a specific port that you define for report server URLs.

**Port conflicts on Windows XP.** On supported editions of 32-bit Windows XP SP2, IIS 5.1 and SSRS cannot use the same port. You cannot configure both IIS 5.1 and a report server to listen on the default HTTP port (port 80).

IIS 5.1 does not use HTTP.SYS for Web applications hosted on the Web server. This means that there is no common queue management for requests that come over the same port, and there is no common repository of registered and reserved URLs.

**Reporting Services Windows Management Instrumentation (WMI) provider** The Reporting Services Windows Management Instrumentation (WMI) provider is not compatible with the previous version. The new version includes additional methods to support URL registration. Because there can only be one version of the Reporting Services WMI provider for a report server installation, this version replaces the previous version. This change represents a breaking change for some deployments. If you created scripts or tools that call the WMI provider, you must revise your code to use the new version. For more information, see [Reporting Services WMI Provider](http://go.microsoft.com/fwlink/?LinkId=134520) in SQL Server 2008 Books Online.

This change also prevents users from connecting to a SQL Server 2005 instance in Management Studio when the user specifies the <*server\_name*>\<*instance\_name*> format to connect. Instead, users must type the report server URL to connect.

**Consolidation of services and applications.** The Report Server Web service, Report Manager, and the background processing application are consolidated into a single service. You cannot start or stop them separately.

**SSRS configuration files.** SSRS configuration files are also consolidated. The RSReportServer.config file is the primary configuration file for Report Manager and the Report Server Web service. The RSWebApplication.config file is obsolete. The following RSWebApplication.config settings have been moved to the RSReportServer.config file:

* ReportServerUrl
* ReportServerExternalUrl
* ReportBuilderTrustLevel
* DeliveryUI settings for delivery extensions
* DisplayErrorLink

The following settings are obsolete and are no longer used:

* ReportServerVirtualDirectory
* MaxActiveReqForOneUser

**Reporting Services trace logs.** ReportServerService\_<timestamp>.log is the primary trace log for all applications that run in the service. The following files are obsolete and are no longer created in SQL Server 2008:

* ReportServerWebApp\_<timestamp>.log
* ReportServer\_<timestamp>.log
* ReportServerService\_main\_<timestamp>.log

**Reporting Services Configuration tool.** TheReporting Services Configuration tool no longer supports the Upgrade Database or Grant Rights features that let you upgrade or grant permissions as independent operations or generate script templates for performing these tasks. In SSRS 2008, both upgrading and database permissions are handled as internal operations.

**SQL Server Management Studio.** In Management Studio, the Home folder is removed in this release. You cannot view, manage, distribute, or secure report server content in Management Studio.

**Report Manager.** In Report Manager, the following links are removed from Site Settings:

* Configure item-level role definitions
* Configure system-level role definitions
* Manage jobs

Report Manager no longer supports the ability to create, modify, or delete role definitions. You must use Management Studio to manage which tasks are in specific roles. Similarly, job management has moved from Report Manager to Management Studio.

**E-mail subscriptions.** E-mail subscriptions will not work for e-mail aliases in the Sender, To, Cc, Bcc, and Reply-To fields when the report server or the remote SMTP server is upgraded to Windows Vista or Windows Server 2008.

**SQL Server 2008 Reporting Services Add-in for SharePoint Technologies.** [The SQL Server 2008 Reporting Services Add-in for SharePoint Technologies](http://go.microsoft.com/fwlink/?LinkId=134521) provides report rendering, processing, and management capabilities as well as data-driven subscriptions when you run a SQL Server 2008 report server instance in SharePoint integrated mode. The add-in download contains a Report Viewer Web part, Web application pages, and support for using either Windows SharePoint Services (WSS) or Microsoft Office SharePoint Services (MOSS).

**Basic authentication.** In SSRS 2008, only NETWORK and NETWORK\_CLEARTEXT logon types are supported with Basic authentication; Interactive and BATCH logon types are not supported.

#### Report Builder Breaking Changes

**Report Builder runs in full trust mode only.** In earlier versions of SSRS running in native mode, you could start SQL Server 2005 Report Builder by using the following URLS:

* **Full trust –** For example, http://<*servername*>/reportserver/reportbuilder/reportbuilder.application
* **Partial trust –** For example, http://<*servername*>/reportserver/reportbuilder/reportbuilderlocalintranet.application

For both URLs, <*servername*> is the name of the computer that specifies the report server. For both URLs, reportserver is the name of the report server instance.

In SSRS 2008, you must use the full trust URL to run Report Builder. When you use the full trust URL for the first time, you might be prompted to grant a higher level of permissions for the application. After you grant these permissions the first time, you do not have to set them again.

**Note:** If Report Builder does not run or if you get an error, contact the system administrator. You might not have the permissions that you need to grant a higher level of trust for this application

In SSRS 2008, if you use the partial trust URL, the following error appears when you open or save a report, or switch report servers: "Failed. An error occurred while processing your request. Save your report and restart the application."

#### Report Processing Breaking Changes

The report processing architecture is fundamentally changed in SSRS 2008, providing on-demand report processing. On-demand report processing significantly reduces memory usage on a report server.

#### RDL Upgrade Breaking Changes

The following RDL elements are not supported when you upgrade an existing report.

**Object identifiers in RDL limited to 256 characters.** Identifiers for objects in RDL (for example, textboxID) were previously unrestricted in length. In SSRS 2008, the length of object identifiers is restricted to 256 characters. Identifiers still must be CLS-compliant.

**Interactivity information saved only for the last request.** In earlier versions of SSRS, snapshots saved all possible combinations of interactive choices, such as drillthrough information and toggle choices. You could view page five of a report, but programmatically toggle an item on page one by keeping track of the correct ID for the toggle. In SSRS 2008, interactivity information is generated and saved only for the last rendering request. You cannot view a page and programmatically toggle an item on another page. You can toggle drill-down items only on the current report page.

**Report Object Model namespace change.** In SSRS 2008, the Report Object Model namespace has changed. This namespace provides read-only access from custom code to global collections such as Fields, Parameters, and ReportItems. If existing custom code explicitly uses a fully qualified reference to an earlier namespace, this change is a breaking change. It is recommended that you do not use fully qualified references to access built-in collections from your code. By not explicitly specifying the namespace, custom code references resolve to the version of the Report Object Model for the currently installed version of SSRS.

#### Report Rendering Breaking Changes

The report rendering architecture is fundamentally changed in SSRS 2008 to provide more consistent rendering for paging and layout among different renderers.

**New rendering object model and consistent pagination.** The rendering object model (ROM) has changed for SQL Server 2008. Earlier versions of the ROM are no longer supported. Accessing the ROM from a multithreaded rendering extension (and switching context from multiple threads) is not supported. The new ROM makes the rules for rendering pages more consistent. For more information, see [Understanding Pagination in Reporting Services](http://go.microsoft.com/fwlink/?LinkId=134523) in SQL Server 2008 Books Online.

**Redesigned CSV data renderer.** In earlier versions of SSRS, when you exported a report to a CSV file format, the data was formatted in a way that preserved how the data appeared on the report page. For matrix data regions, this resulted in a data format that was inconvenient to import into other applications so that you could continue to work with the data.

In this release, when you export a report to a CSV file, you can choose between two supported formats: Default mode and Compliant mode. Default mode is optimized for Microsoft Excel. Compliant mode is optimized for third-party applications. For more information, see [Exporting to a CSV File](http://go.microsoft.com/fwlink/?LinkId=134524) in SQL Server 2008 Books Online.

The earlier format for CSV files is no longer available. However, for reports that do not use matrix data regions, you can use Compliant mode to get a file format closest to the earlier CSV file format.

**Aggregates with conditional visibility in page headers and footers.** In earlier versions of SSRS, different renderers used different rules to determine which items with conditional visibility to include on a report page. For example, aggregate calculations were not performed for hidden items in printed reports but were calculated for hidden items in reports that you viewed with a browser or in Excel. In SSRS 2008, all renderers use the same set of rules to determine which items are on a page.

**No formula support in Excel.** Earlier versions of SSRS provided limited support for translating expressions in RDL to Excel formulas. In this release, when you export a report to Excel, RDL expressions are not translated to Excel formulas.

**Overlapping items.** In earlier SSRS versions, if a report had overlapping items on the report design surface, publishing the report produced a warning ("Overlapping report items are not supported in all renderers."), but the report items remained in their original location on the design surface. In SQL Server 2008, report items might be moved to correct overlapping boundaries when a report is viewed or exported to a renderer that does not support overlapping items. For more information, see [Understanding Rendering Behaviors](http://go.microsoft.com/fwlink/?LinkId=134525) in SQL Server 2008 Books Online.

### Behavior Changes

There are a number of behavior changes in SSRS 2008 that might require corrective action after the upgrade is complete. In this section, we look at fundamental changes to SSRS 2008 functionality that might affect how you work.

[Behavior Changes in SQL Server 2008 Reporting Services](http://go.microsoft.com/fwlink/?LinkId=134527), in SQL Server 2008 Books Online, describes behavior changes to the following SSRS 2008 components:

* Report Server Configuration and Management Tools
* Report Authoring
* Report Processing
* Report Rendering

Let’s review the behavior changes in each of these components.

#### Report Server Configuration and Management Tools

SSRS includes several tools and applications that you use to configure the server and manage content and operations. In SQL Server 2008, each tool is aligned to a specific purpose: configuration, administration, or content management. To promote consistency within a tool and to remove overlapping functionality, features and tasks have been added and removed from tools. If you were accustomed to using a tool to perform a given task, you might now need to use a different tool to accomplish the same task. Table 14-3 lists the behavior changes for report server configuration and management tools.

Table 14-3: Behavior Changes for Report Server Configuration and Management Tools

| **Feature** | **Description** |
| --- | --- |
| Reporting Services Configuration | Color-code status icons have been removed. New URL configuration pages replace the pages for creating virtual directories. The workflow for creating and configuring a report server database has been revised. You now use a wizard to create or update database connections. |
| SQL Server Management Studio | Management Studio supports only server administration tasks. You can connect to and configure a report server that runs in native mode or in SharePoint integrated mode. |
| Report Manager | You use Report Manager to view and manage report server content. SSRS 2008 introduces the ability to manage report models. You can now set model item security and associate click-through reports to entities in a model. When viewing a report in Report Manager, because of the changes introduced by on-demand report processing, the toolbar displays a page estimate with a question mark instead of the actual number of pages for a report. You can still click the Last Page button and navigate to the end of the report. |

Table 14-4 summarizes the tasks supported by the various SQL Server 2008 tools.

Table 14-4: SSRS Tasks Supported by Different Tools

| **Task** | **Report Server Configuration** | **SQL Server Management Studio** | **Report Manager** | **SharePoint UI** | **Command-Line Utilities** |
| --- | --- | --- | --- | --- | --- |
| Reserve URLs | X |  |  |  | X |
| Set the service account and password | X |  |  |  | X |
| Create the report server database | X |  |  |  |  |
| Change connection string | X |  |  |  | X |
| Configure report server scale-out | X |  |  |  | X |
| Back up, restore, change keys, or delete encrypted data | X |  |  |  | X |
| Configure unattended execution account | X |  |  |  | X |
| Configure report server email | X |  |  |  |  |
| Enable My Reports |  | X |  |  |  |
| Enable logging on report execution |  | X |  |  |  |
| Enable client-side printing |  | X |  |  |  |
| Set server defaults for report history |  | X |  |  |  |
| Create or modify role definitions |  | X |  |  |  |
| View status of a running report or model process and stop it if it is taking too long |  | X |  |  |  |
| All Report Server System Properties |  | X |  |  |  |
| Grant permissions to report server items and operations by creating role assignments at the item and system level |  |  | X |  |  |
| Define and manage the report server folder hierarchy |  |  | X | X |  |
| View reports, report models, shared data sources, resources, and folders |  |  | X | X |  |
| Upload report definition (.rdl), report model (.smdl), and resource files |  |  | X | X |  |
| Create and manage shared schedules |  |  | X | X |  |
| Create and manage linked reports |  |  | X |  |  |
| Create and manage report history |  |  | X | x |  |
| Create and manage shared data sources and any data source properties defined in an individual report |  |  | X | X |  |
| Schedule when data processing occurs for a report, or configure a report to run as a report execution snapshot |  |  | X | X |  |
| Subscribe to report deliveries and create and manage data-driven subscriptions |  |  | X | X |  |
| Create data-driven subscriptions |  |  | X | X |  |
| Use Report Builder to create, modify, and save reports |  |  | X | X |  |
| Generate models, associate click-through reports to entities in a model, and set model item security |  |  | X | x |  |

Report Authoring. In the SQL Server 2008 RTM version, the new functionality for SSRS Report Designer is integrated with Business Intelligence Development Studio. Here are some behavior changes related to report authoring.

For more information, see [What's New in Report Authoring](http://go.microsoft.com/fwlink/?LinkId=134533) in SQL Server 2008 Books Online.

**Publishing a report.** In SQL Server 2008 RTM, you can upload existing reports written for earlier versions of RDL to an SSRS report server. When uploaded to the report server, the report is automatically upgraded and passed to the report processor. The report you upload remains in its original format. If you save the report after uploading it to the report server, you get an identical copy of the report you uploaded.

**Data regions on the toolbox.** In earlier SSRS versions, the four data regions (Table, Matrix, List, and Chart) were distinct report items with their own layout behavior and properties. In SSRS 2008, the Table, Matrix, and List data regions have been replaced by a new flexible grid layout called a Tablix data region, which uses predefined templates to create the former data regions. The Tablix data region lets you combine aspects of tables and matrices into flexible report layouts. For more information about the Tablix data region, see [Working with a Tablix Data Region](http://go.microsoft.com/fwlink/?LinkId=134534) in SQL Server 2008 Books Online.

The Chart data region remains a separate report item. New chart types—such as Polar, Radar, and Funnel—have been added to the Chart data region. For more information about the new chart types, see [Working with Chart Data Regions](http://go.microsoft.com/fwlink/?LinkId=134535) in SQL Server 2008 Books Online.

**Preserving white space in a report body or rectangle container.** Extra white space is no longer removed by default. Now when you render a report that had extra white space on the report body when viewed on the report design surface, the trailing white space after the last report item on the page is preserved. This might result in more pages for an existing report. To remove the white space, set the report property ConsumeContainerWhitespace to true. For more information about this change, see [What's New in Report Authoring](http://go.microsoft.com/fwlink/?LinkId=134533) in SQL Server 2008 Books Online..

#### Report Processing

Report processing has been redesigned in SQL Server 2008, with reports now processed and rendered page by page as a report user interactively reads through a report. The amount of data on each page influences the rendering time for each page. The total number of pages is determined when a report is rendered. For some renderers, an estimated number of pages is displayed until all pages in a report have been rendered. Here are some other behavior changes in SSRS 2008 report processing.

**Images.** Images are no longer retrieved during the initial session when a report is rendered. Images are retrieved when they are accessed for the first time during on-demand processing. For history and execution snapshots, images are retrieved at snapshot-creation time.

**Execution Log: TimeDataRetrieval, TimeProcessing, TimeRendering.** Report log entries for TimeDataRetrieval, TimeProcessing, and TimeRendering are logged when a report has been rendered. See the [SSRS Samples on CodePlex](http://go.microsoft.com/fwlink/?LinkId=134537) for code examples that read and review the Execution Log.

**Error detection on export.** In earlier SSRS versions, the entire report was processed before any page could be viewed. Errors in expressions for the Visibility.Hidden RDL property were detected before a report could be exported. If you could view the first page of a report, you could export the entire report without error.

In SSRS 2008, however, reports are processed page by page. If errors exist in an expression for the Visibility.Hidden RDL property, the error might not be detected until the page on which the error exists is rendered for export. In this case, the entire export fails. Being able to view a few pages of a report successfully does not guarantee that you can export the entire report. You must try to export the report and wait for a successful completion before you know that the report exports without error.

Expression evaluation for group, sort, and filter operations continue to behave as in previous SSRS versions. Errors in these expressions are detected by the report processing component and are reported as critical errors before the first page of a report is rendered.

#### Report Rendering

Report rendering redesign in SSRS 2008 has introduced the following behavior changes when rendering an existing report.

**Page breaks.** In earlier SSRS versions, soft page break renderers handled report items in a container (in a rectangle or in the report body) in the following way: Page breaks from the top-most and bottom-most report items were applied to the container to minimize extra blank pages. In the new ROM, page breaks that you set on report items, known as logical page breaks, always cause a new page to be rendered. No attempt is made to eliminate extra pages. For more information about this change, see [Understanding Pagination in Reporting Services](http://technet.microsoft.com/en-us/library/bb677374.aspx) in SQL Server 2008 Books Online.

**RepeatWith items.** In earlier SSRS versions, soft page break renderers included report items on a page when the RepeatWith property was set to true. These report items were not counted when calculating page size because of the flexible nature of page size for a soft page break renderer, nor were they counted when you set InteractiveHeight to control the amount of data on a page. In SQL Server 2008, these items are counted toward the total page size. The result is that pages might contain less data, but setting the value for InteractiveHeight has more influence on the page size. For more information about this change, see [Understanding Rendering Behaviors](http://technet.microsoft.com/en-us/library/bb677573.aspx) in SQL Server 2008 Books Online.

**Nested subreports and data regions in Excel.** In earlier versions of SSRS, nested data regions and subreports in table and matrix cells were not supported when you exported a report to Excel. In SQL Server 2008, this limitation has been removed. You can design reports that use nested data regions and subreports in a data region, export the report to the Excel renderer, and view the nested report items. For more information about this change, see [Exporting to Microsoft Excel](http://technet.microsoft.com/en-us/library/ms159836.aspx) in SQL Server 2008 Books Online.

#### Updating Report Projects and Definitions for Use in BI Development Studio

You must update existing report projects and definitions for use within BI Development Studio so that you can make updates and changes to existing reports. When an existing report project is opened within BI Development Studio, the project will need to be upgraded to Visual Studio 2008 format. After opening the project, the BI Development Studio environment will launch the Visual Studio Conversion Wizard, which you can use to perform the upgrade.

After you have upgraded the project for use within the BI Development Studio environment, you must also upgrade each individual report from the SSRS 2000 or SSRS 2005 format to the SSRS 2008 format. This will update the RDL within each report to ensure that it is compatible with the new version. Open each report in the project to launch the report converter. BI Development Studio will display a confirmation message and update the report RDL. The updated report will then be opened in the Report Designer within BI Development Studio. Save the report to complete the upgrade process. You need to upgrade and save each report within a project before you can fully deploy the project to SSRS 2008.

When reports are converted, a number of changes are made:

* The report definition namespace is upgraded to [RDL 2008](http://go.microsoft.com/fwlink/?LinkId=134538).
* The CustomReportItem element is modified to support data-bound controls. The element now includes child elements that describe the data used by the control as well as the properties and dimensions of the control in the report.
* The Custom element is replaced by a custom properties collection that contains name-value pairs. Upon upgrade, all instances of a custom element are mapped to a custom property in the CustomProperties collection.

Once you have converted and saved a given report, you can deploy it to an SSRS 2008 report server. If you have converted all the reports in a given project, you can deploy the entire project.

**Important**: Once a report has been converted to the SSRS 2008 schema, it can no longer be published to an SSRS 2000 or SSRS 2005 instance. However, SSRS 2008 can read previous version RDLs without conversion.

### Upgrade Tools

#### SQL Server 2008 Upgrade Advisor

SQL Server 2008 Upgrade Advisor helps you prepare for upgrades to SQL Server 2008. Upgrade Advisor analyzes installed components from earlier versions of SQL Server, and then generates a report that identifies issues to fix either before or after you upgrade.

**Installing and running Upgrade Advisor.** Where you install Upgrade Advisor depends on what you want to analyze. Upgrade Advisor supports remote analysis of all supported components except Reporting Services. If you are not scanning instances of SSRS, you can install Upgrade Advisor on any computer that can connect to your instance of SQL Server and that meets the Upgrade Advisor prerequisites. For details, see [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=134494) in SQL Server 2008 Books Online. If you are scanning instances of SSRS, you must install Upgrade Advisor on the report server.

* Upgrade Advisor is available in the Servers\redist\Upgrade Advisor folder of the SQL Server installation media, and from the [Microsoft Download Center](http://go.microsoft.com/fwlink/?LinkId=113913).

Prerequisites for installing and running Upgrade Advisor are as follows:

* Windows XP SP2 or later, Windows Vista, Windows Server 2003 SP1 or later, or Windows Server 2008
* Windows Installer 4.5 or later. The .NET Framework 2.0 requires Windows Installer 4.5. You can install Windows Installer from the [Windows Installer Web site](http://go.microsoft.com/fwlink/?LinkId=49112).
* The .NET Framework 2.0 or later. The .NET Framework 2.0 is available on the SQL Server 2005 product media as well as from the [.NET Framework 2.0 SDKs, Redistributables & Service Packs Web site](http://go.microsoft.com/fwlink/?LinkId=48882). To install .NET Framework 2.0 from the SQL Server 2005 media, locate the root of the disk drive, double-click the redist folder, double-click the 2.0 folder, and run dotnetfx.exe (for 32 bit) or dotnetfx64.exe (for 64 bit), depending on your operating system.
* SQL Server 2000 decision support objects (DSO) are required to scan upgrade issues in SSAS. To install DSO, insert the SQL Server 2000 media into the disk drive. This starts the SQL Server 2000 Setup program. Click Install SQL Server 2000 Components, and then click Analysis Services to start the Analysis Services Setup program. In Select Components, make sure that the Decision Support Objects component is selected.
* SQL Server 2000 Client components are required to scan SQL Server 2000 DTS packages. Use the SQL Server 2000 installation disk to install client components.
* SQL Server 2005 Backward Compatibility Components are required to scan SQL Server 2005 DTS packages that were migrated from SQL Server 2000. Use the SQL Server 2005 installation disk to install backward compatibility components.

To install Upgrade Advisor from the Web, click the download button on the download page. You can then run installation immediately, or save the SQLUA.msi file to run later. If you are installing from the product disc, run SQLUA.msi directly from the product disk. After you install Upgrade Advisor, you can open it from the Start menu:

* Click Start, point to All Programs, point to Microsoft SQL Server 2008, and then click SQL Server 2008 Upgrade Advisor.

For more information about using Upgrade Advisor, see Chapter 1, “Upgrade Planning and Deployment,” which also discusses using the SQL Server 2008 Upgrade Assistant and the Best Practices Analyzer for SQL Server 2000 and SQL Server 2005 to prepare for an upgrade. Also see [Using Upgrade Advisor to Prepare for Upgrades](http://go.microsoft.com/fwlink/?LinkID=132624) in SQL Server 2008 Books Online.

### 64-bit Considerations

Cross-platform upgrades are not supported. You cannot upgrade a 32-bit instance of SQL Server to native 64-bit. However, you can upgrade a 32-bit instance of SQL Server to Windows On Windows 64 (WOW64), the 32-bit subsystem on a 64-bit server, as noted in the table above. You can also back up or detach databases from a 32-bit instance of SQL Server and then restore or attach them to an instance of SQL Server (64-bit) if the databases are not published in replication. In this case, you must also recreate any logins and other user objects in master, msdb, and model system databases.

### Known Issues and Workarounds

Regardless of whether you choose an in-place upgrade or a side-by-side upgrade of SSRS 2000 or SSRS 2005 to SSRS 2008, there is a range of potential issues you might face during an upgrade, as we saw earlier in the “Preparing to Upgrade” section. To obtain a report that identifies many of these potential issues before you begin an upgrade, run Upgrade Advisor to analyze the instance that you want to upgrade. If any of these issues are reported, follow the Upgrade Advisor’s recommendations and guidance for possible mitigation options and strategies. There is also a category of issues that either cannot be detected by Upgrade Advisor or the detection of which would result in too many false-positive results.

Let’s look at the most important upgrade issues, whether detected by Upgrade Advisor or not. For a comprehensive list of backward-compatibility issues, breaking changes, and behavior changes to SSRS in SQL Server 2008, see [Reporting Services Backward Compatibility](http://go.microsoft.com/fwlink/?LinkId=134539) in SQL Server 2008 Books Online.

For a complete list of the SSRS upgrade issues that Upgrade Advisor detects, see “Reporting Services Upgrade Issues” in SQL Server 2008 Upgrade Advisor Help file.

**Issues preventing an in-place upgrade.** Certain SSRS 2000 or SSRS 2005 configurations might block the in-place upgrade process and prevent it from running. Before proceeding with an in-place upgrade, you should investigate these items to ensure that no problems will occur. Note that Upgrade Advisor detects and reports these blockers, so use this tool to check for these situations as well as to check for other problems and issues that might affect the upgrade process.

Although changes to the names of the virtual directories will not block an in-place upgrade, other configuration changes will. In particular, you should configure the virtual directories with the following default settings:

1. Security for the virtual directories must be set to Integrated Windows Authentication; Anonymous Access is not supported for an in-place upgrade.
2. The default Application Mappings should be set as follows:
   1. For the report server virtual directory, the wild card mapping must point to the v1.1 aspnet\_isapi.dll executable, and no other script maps should exist.
   2. For the Report Manager virtual directory, the .asax and .aspx extensions must point to the v1.1 aspnet\_isap.dll executable.

Reset the virtual directories to their original default configuration settings to allow an in-place upgrade to proceed. If resetting these configuration settings is not possible, perform a side-by-side upgrade rather than an in-place upgrade.

The ASP.NET account information cannot be encrypted within the registry. Although encrypting the account information is considered a security best practice for some IIS installations, SQL Server 2008 cannot upgrade an SSRS 2000 or SSRS 2005 installation configured in this manner. To proceed with an upgrade, temporarily add unencrypted account information to the Machine.config file by using the following steps:

1. Make a backup copy of the existing Machine.config file.
2. Open Machine.config in a text editor, and find the <processModel> element.
3. Find the User attribute. This attribute is created when you specify a custom domain account to run ASP.NET.
4. Modify the User attribute, and specify an unencrypted user name and password.
5. Upgrade SSRS.
6. After the upgrade is complete, modify the Machine.config file so that the User attribute within the <processModel> element specifies the encrypted values used before any changes were made.

If you have deployed custom extensions to your report server, remove references to these extensions from your report server configuration file to perform an in-place upgrade, or leave them in place and perform a side-by-side upgrade. A side-by-side upgrade is recommended in this case to ensure that reports continue to execute the way they did before the upgrade. For side-by-side upgrade, see [How to: Migrate a Reporting Services Installation](http://go.microsoft.com/fwlink/?LinkID=134498) in SQL Server 2008 Books Online. If any of these blocker situations exist and cannot be resolved, the installation cannot be upgraded in place; perform the upgrade using the side-by-side upgrade process instead..

If you are upgrading an Evaluation edition of SSRS 2000 or SSRS 2005, using an in-place upgrade requires that the Evaluation edition still be active (the evaluation period must not have expired). If the Evaluation edition has expired, upgrade the installation by using the side-by-side upgrade process.

If any of these blocker situations exist and cannot be resolved, the installation cannot be upgraded in place; perform the upgrade using the side-by-side upgrade process instead.

### Backup and Rollback Plan

Before upgrading to SSRS 2008, review the following requirements and make sure you have a backup and rollback plan in place:

1. Review requirements to determine whether your hardware and software can support SSRS 2008.
2. Use System Configuration Checker (SCC) to scan the report server computer for any conditions that might prevent a successful installation of SQL Server 2008. For more information, see [Check Parameters for the System Configuration Checker](http://go.microsoft.com/fwlink/?LinkID=133947) in SQL Server 2008 Books Online.
3. Review security best practices and guidance for SQL Server. For more information, see [Security Considerations for a SQL Server Installation](http://go.microsoft.com/fwlink/?LinkID=133079) in SQL Server 2008 Books Online.
4. Run Upgrade Advisor on the report server computer to determine any issues that might prevent you from successfully upgrading.
5. Back up your symmetric key. For details, see [Backing Up and Restoring Encryption Keys](http://go.microsoft.com/fwlink/?LinkId=134541) in SQL Server 2008 Books Online.
6. Back up your report server databases. For details, see [Moving the Report Server Databases to Another Computer](http://go.microsoft.com/fwlink/?LinkId=134542) in SQL Server 2008 Books Online.
7. Back up the following report server configuration files:
   1. Rsreportserver.config
   2. Rswebapplication.config
   3. Rssvrpolicy.config
   4. Rsmgrpolicy.config
   5. Reportingservicesservice.exe.config
   6. Web.config (for both the report server and Report Manager ASP.NET applications)
   7. Machine.config (for ASP.NET if you modified it for report server operations)
8. Back up any customizations to existing SSRS virtual directories in IIS.

Before you upgrade a production environment, always run a test upgrade in a pre-production environment that has the same configuration as your production environment. To view a list of considerations for upgrading SSRS, see [Considerations for Upgrading Reporting Services](http://go.microsoft.com/fwlink/?LinkID=134175) in SQL Server 2008 Books Online.

## Upgrading from SQL Server 2000

### In-Place Upgrade

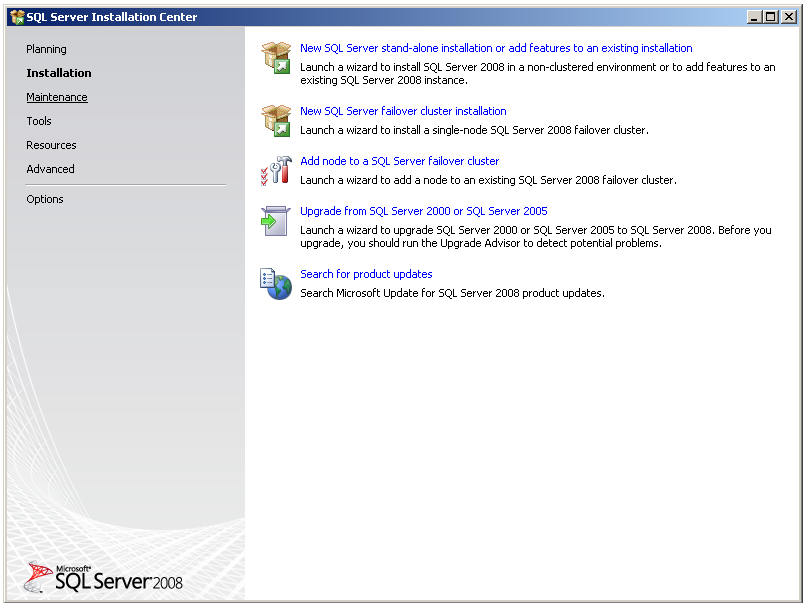
If you are performing an in-place upgrade, you need to know the SSRS 2000 is always installed as a default instance. If the report server database resides within the default instance of SQL Server 2000 on the same server, the relational engine and the report server must be upgrade together. There is no support to upgrade an SSRS 2000 instance with remote catalog on SQL Server 2000, thus SQL Server 2000 would have to be upgraded first to either SQL Server 2005 or SQL Server 2008.

In an in-place upgrade of an SSRS 2000 installation to SSRS 2008, the upgrade process handles all aspects of the upgrade, automatically updating report server content, report definitions, and component configurations. Note, however, that this upgrade does not automatically handle updates to client workstations and computers that have the Report Designer or management tools installed. You will have to upgrade those workstations and computers after you upgrade the report server.

#### Upgrading via the Setup Application

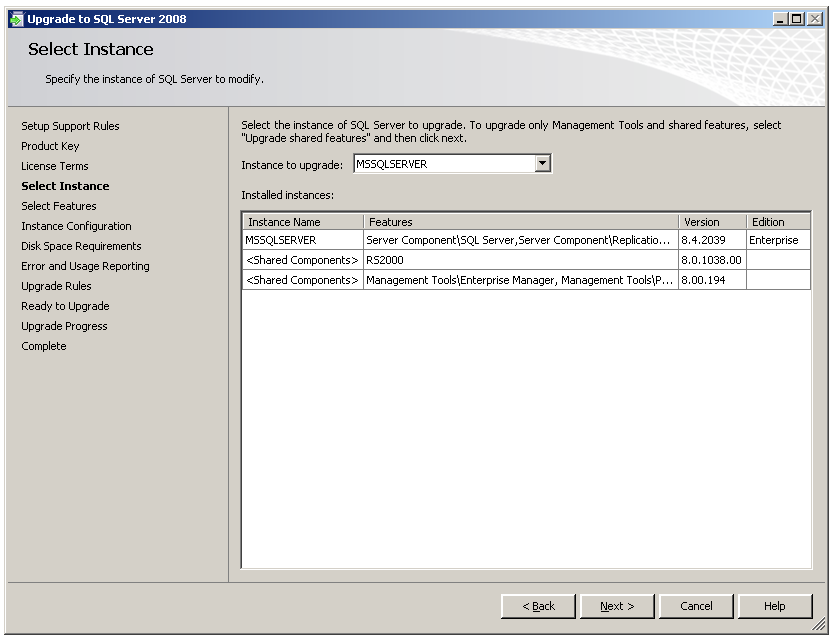
Here are the steps for upgrading SSRS 2000 to SSRS 2008 (you follow exactly the same steps for an in-place upgrade from SSRS 2005 to SSRS 2008):

1. Insert the SQL Server installation media, and from the root folder, double-click setup.exe. To install from a network share, navigate to the root folder on the share, and then double-click Setup.exe.
2. If the Microsoft .NET Framework version 2.0 installation dialog box appears, select the check box to accept the .NET Framework 2.0 License Agreement. Click Next. To quit SQL Server 2008 installation, click Cancel. When installation of .NET Framework 2.0 is complete, click Finish.
3. Windows Installer 4.5 is also required and might be installed by the Installation Wizard. If you are prompted to restart your computer, restart, and then run SQL Server 2008 setup.exe again.
4. When prerequisites are installed, the Installation Wizard will launch the SQL Server Installation Center. To upgrade an existing instance of SQL Server, click Upgrade from SQL Server 2000 or SQL Server 2005. Figure 14-5 shows the upgrade selection screen.



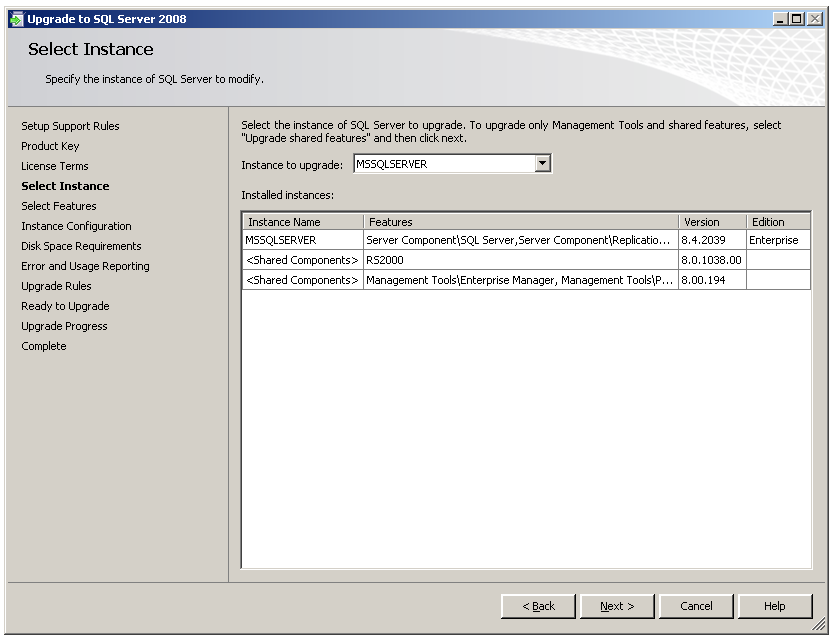
**Figure 14-5:** Upgrade from SQL Server 2000 or SQL Server 2005screen

1. If Setup support files are required, SQL Server Setup will install them. If you are instructed to restart your computer, restart before you continue.
2. The SCC will run a discovery operation on your computer. To continue, click OK. Setup log files have been created for your installation. For more information about log files, see [How to: View SQL Server 2008 Setup Log Files](http://go.microsoft.com/fwlink/?LinkId=134543) in SQL Server 2008 Books Online.
3. On the Product key page, click a radio button to indicate whether you are upgrading to a free edition of SQL Server or whether you have a PID key for a production version of the product.
4. On the License Terms page, read the license agreement, and then select the check box to accept the licensing terms and conditions. To continue, click Next. To end Setup, click Cancel.
5. On the Select Instance page, specify the instance of SQL Server to upgrade. Figure 14-6 shows the Select Instance screen.



**Figure 14-6:** The Select Instance screen

1. On the Select Features page, the features to upgrade will be pre-selected. A description for each component group appears in the right-hand pane after you select the feature name. Note that you cannot change the features to be upgraded, and you cannot add features during the upgrade operation. To add features to an upgraded instance of SQL Server 2008 after the upgrade operation is complete, see [How to: Add Features to an Instance of SQL Server 2008 (Setup)](http://go.microsoft.com/fwlink/?LinkId=134544) in SQL Server 2008 Books Online. Figure 14-7 shows the Select Features screen.



**Figure 14-7:** Upgrade to SQL Server 2008 Select Features screen

1. On the Instance Configuration page, specify whether to install a default or a named instance. For details, see [Instance Configuration](http://go.microsoft.com/fwlink/?LinkId=134545) in SQL Server 2008 Books Online.

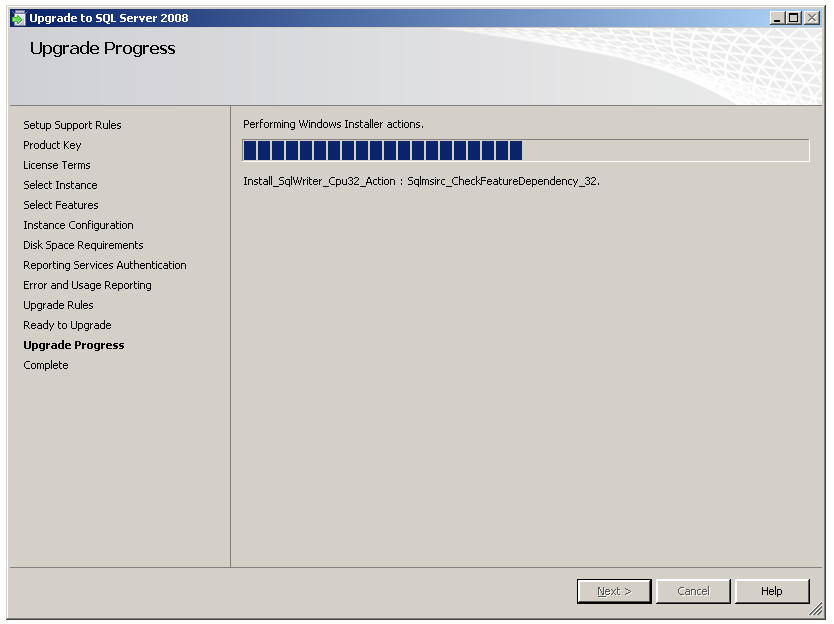
**Instance ID suffix** — By default, the instance name is used as the Instance ID suffix, which identifies installation directories and registry keys for your instance of SQL Server. This is the case for default instances and named instances. For a default instance, the instance name and instance ID suffix would be MSSQLSERVER. To use a non-default instance ID suffix, select the Instance ID suffix check box and provide a value.

**Note:** Typical standalone instances of SQL Server 2008, whether default or named instances, do not use a non-default value for the Instance ID suffix check box.

All SQL Server service packs and upgrades will apply to every component of an instance of SQL Server.

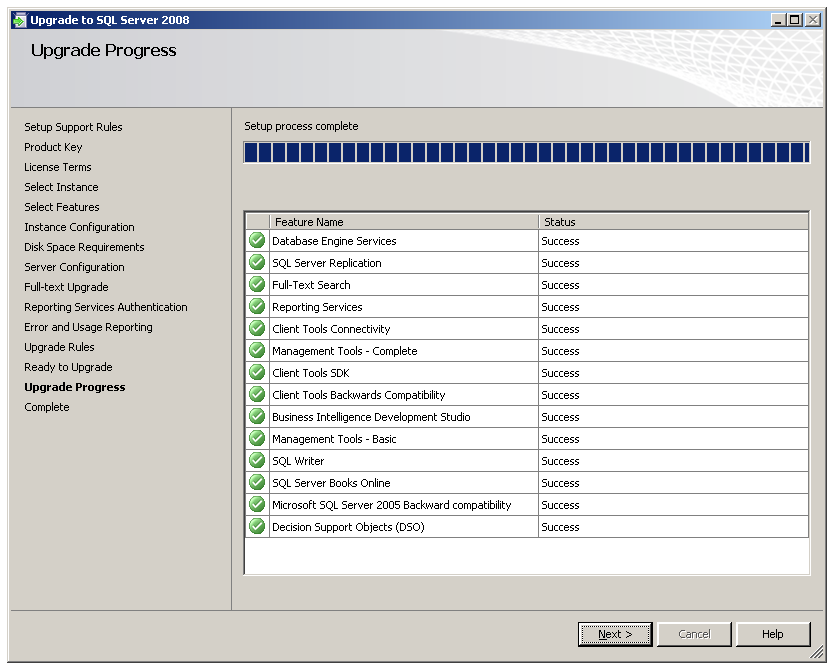
**Detected instances and features** — The grid will show instances of SQL Server that are on the computer where Setup is running. If a default instance is already installed on the computer, you must install a named instance of SQL Server 2008. To continue, click Next.

1. The Disk Space Requirements page calculates the required disk space for the features you specify and compares requirements to the available disk space on the computer where Setup is running. For more information, see [Disk Cost Summary](http://go.microsoft.com/fwlink/?LinkId=134546) in SQL Server 2008 Books Online.
2. Work flow for the remainder of this topic depends on the features you have specified for your installation. You might not see all of the pages, depending on your selections.
3. On the Full-Text Search Upgrade Options page, specify the upgrade options for the databases being upgraded. For more information, see [Full-Text Search Upgrade Options](http://go.microsoft.com/fwlink/?LinkId=134547) in SQL Server 2008 Books Online.
4. On the Error and Usage Reporting page, specify the information you would like to send to Microsoft that will help to improve SQL Server. By default, options for error reporting and feature usage are enabled. For more information, see [Error and Usage Report Settings](http://go.microsoft.com/fwlink/?LinkId=134548) in SQL Server 2008 Books Online.
5. The System Configuration Checker will run one more set of rules to validate your computer configuration with the SQL Server features you have specified before the upgrade operation begins.
6. The Ready to Upgrade page displays a tree view of upgrade options that were specified during Setup. To continue, click Install.
7. During upgrade, the progress page provides status so that you can monitor upgrade progress as Setup proceeds. Figure 14-8 shows the Upgrade Progress page.

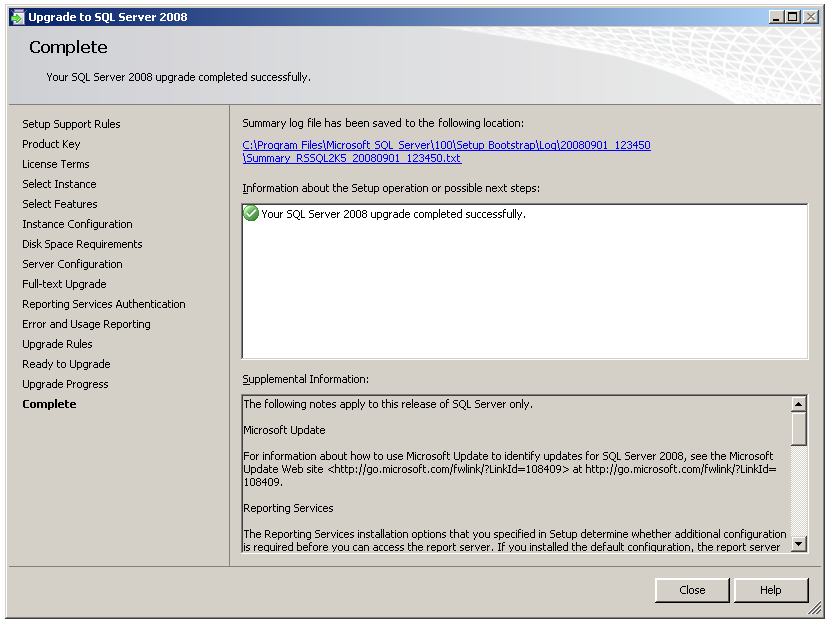


**Figure 14-8:** Upgrade Progress screen

1. After installation, the Complete page provides a link to the summary log file for the installation and other important notes. To complete the SQL Server installation process, click Close. Figure 14-9 shows the Upgrade Progress by Feature Name report, and Figure 14-10 shows the Complete screen.



**Figure 14-9:** Upgrade Progress by Feature Name report screen



**Figure 14-10:** Complete screen

1. If you are instructed to restart the computer, do so now. It is important to read the message from the Installation Wizard when you are done with Setup. For information about Setup log files, see [How to: View SQL Server 2008 Setup Log Files](http://go.microsoft.com/fwlink/?LinkID=134543) in SQL Server 2008 Books Online.

**Note:** For more information about upgrading to SQL Server 2008, see [How to: Upgrade to SQL Server 2008 (Setup)](http://go.microsoft.com/fwlink/?LinkId=134549) in SQL Server 2008 Books Online.

### Side-by-Side Upgrade

Alternatively, you can upgrade SSRS 2000 installations to SSRS 2008 by using the side-by-side upgrade method. You can perform a side-by-side upgrade on a single server (the existing report server) or by using two servers (to take advantage of new hardware, for example).

When you perform the upgrade on a single server, you install a new instance of SSRS 2008 alongside the existing SSRS 2000 installation and then manually move report server content, report definitions, and other configuration information to the new instance.

When you perform the upgrade by using two servers, you install SSRS 2008 on the new server (as the default instance or as a named instance) and then perform the same manual movement of report server content, report definitions, and configuration information.

**Note:** Regardless of the upgrade process you use, workstations and computers hosting the Report Designer or SSRS 2000 management tools will have to be upgraded after the report server is upgraded.

#### Installing the New Instance

The first step for a side-by-side upgrade is to install (but not configure) SSRS 2008. The following points should be considered when planning a single-server or a two-server upgrade process:

If an additional server is not available, you can use a single-server upgrade process. During the upgrade process, you must install SSRS 2008 as a named instance. After the upgrade (and testing) is complete, you can rename the named instance to serve as the default instance once you have uninstalled SSRS 2000.

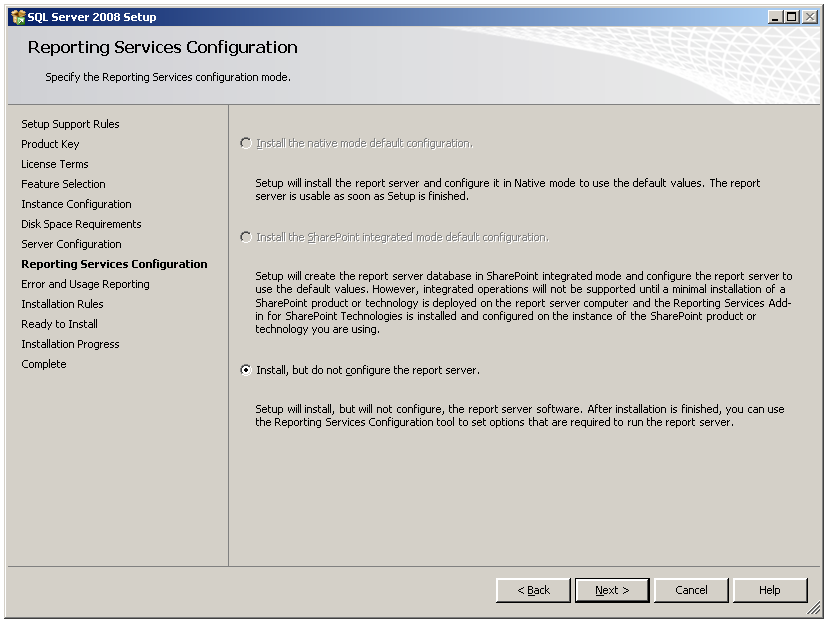
If an additional server is available and will serve as the new SSRS 2008 report server, you can install SSRS 2008 as the default instance or as a named instance on the new server. After the upgrade (and testing) is complete, you can decommission the old report server or reuse it for other purposes.

To begin the upgrade process, start the Setup application for SQL Server 2008.

1. After starting, the Setup application will install a set of prerequisites to the installation of SQL Server 2008 components, run a system configuration check, gather system information, and prompt for typical registration information (user name, company name, and product key).
2. The application will then provide options for selecting components to install. Select the Reporting Services option. When using a single-server upgrade, in cases where the report server previously had client components installed (such as Books Online), select the Workstation Components option as well.
3. The Setup application will then prompt for an instance name for the new SQL Server 2008 components. For a single-server upgrade, select Named Instance,and enter a new instance name; for a two-server upgrade, select Default Instance to install SSRS 2008 as a default instance, or select Named Instanceand enter a new instance name.
4. **Note**: The default selection made by the Setup application is Default Instance. Thus, if you need to use a new instance name, be sure to select Named Instance and provide a new instance name.
5. The Setup application will prompt for service account information. Choose the credentials to use for the Windows service to be created for SSRS 2008 and proceed.
6. Next, the Setup application will display an installation options screen for specifying the type of installation to perform.

When you are installing a default instance of SSRS 2008, the Setup application can create and fully configure the default instance. However, when you are installing a named instance (to support a single-server side-by-side upgrade, for example), the Setup application can install the report server software but cannot configure the new instance. Thus, you must manually configure the new named instance (in this case, by using specific steps to migrate the existing report server’s configuration and content to the new instance). If you are performing a two-server side-by-side upgrade and installing SSRS 2008 as the default instance on the new server, Microsoft recommends that you select the *Install, but do not configure the report server* option. Although the default instance can be configured by the Setup application during installation, you will need to manually reconfigure the configuration by using the steps we will cover in a moment to migrate the existing report server’s configuration and content to the new server.

Figure 14-11 shows the Setup application screen you use to select the type of installation to complete.



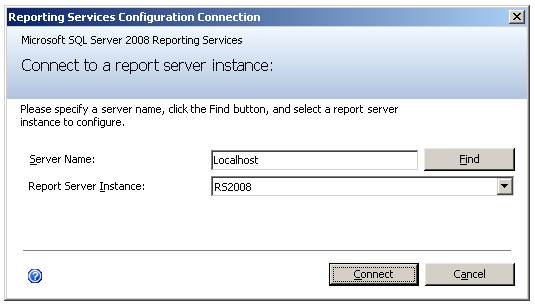
**Figure 14-11:** Install, but do not configure the report server installation option

1. To complete the installation of the new instance, simply proceed through the remaining screens of the Setup application. All the required SSRS 2008 files will be installed, and the new instance will be ready for migration.

#### Configuring the New Instance

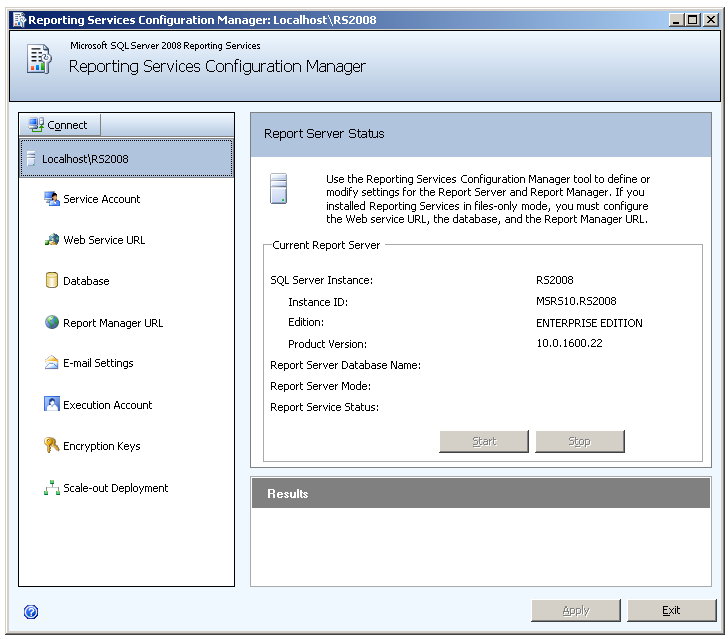
After you have installed the new instance, you should use the new Reporting Services Configuration tool to configure the instance. Launch the tool from the Configuration Tools group within the Microsoft SQL Server 2008 Start menu group.

1. When the configuration tool is first started, enter the name of the server (or “localhost”), and use the Find button to locate the newly installed SSRS 2008 instance. Figure 14-12 shows the screen for finding and connecting to the new instance.



**Figure 14-12:** Connect to a report server instance screen

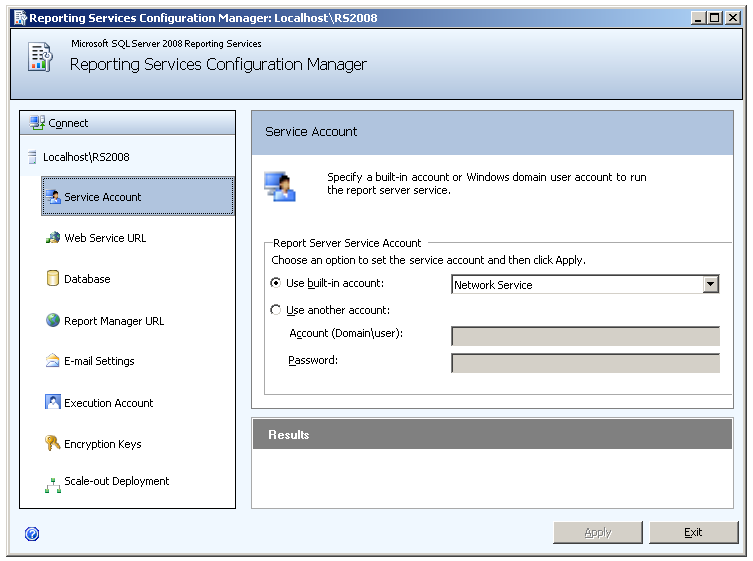
1. Once connected to the new instance, the configuration tool will show the current status of each item requiring configuration for an instance of SSRS 2008. Figure 14-13 shows the status of each area before configuration is complete.



**Figure 14-13:** Initial state of new SSRS 2008 instance

**Note**: If the configuration tool is started right after the installation is complete, the Server Status screen might show that the server is stopped. It can be started before, during, or after the configuration steps are completed.

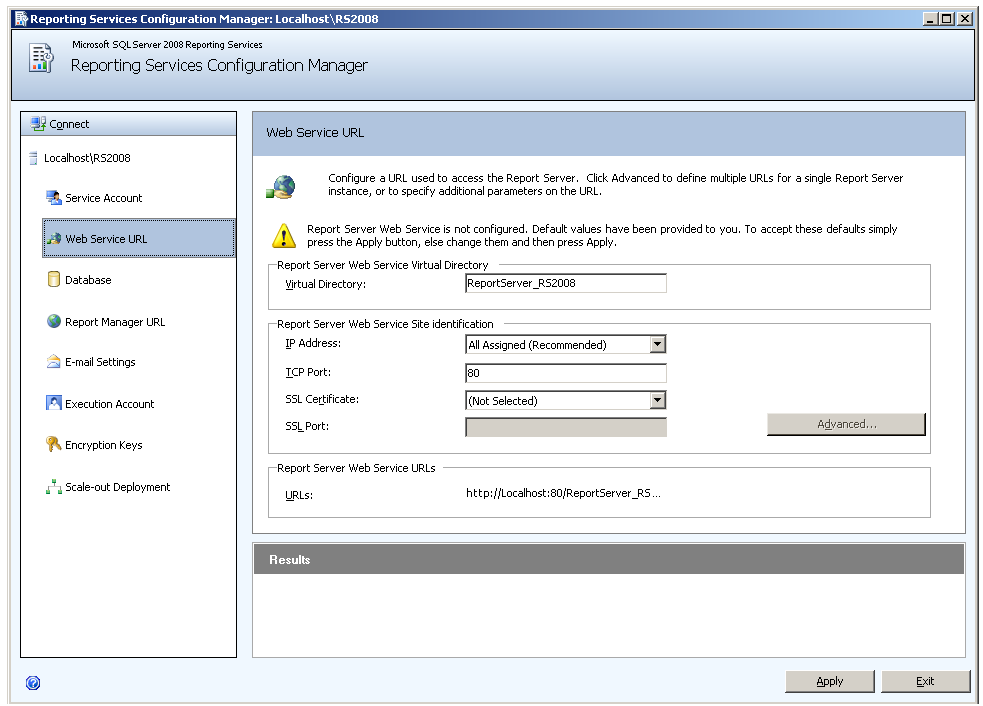
1. To start the configuration process, use the Service Account option to configure the account to run the ReportServer service, as Figure 14-14 shows.



**Figure 14-14:** Service account of new SSRS 2008 instance

1. Use the Web Service URL option to configure a URL to access the report server; you can use the default values provided or specify different values. You can use any names for a Web service URL except virtual directory names already in use. Thus, because the original instance of SSRS 2000 is still installed and running, the virtual directories should not be named ReportServer or Reports (the default names for the virtual directories created when SSRS 2000 is installed).

Figure 14-15 shows the Report Server Web services URL being created, using ReportServer\_RS2008 as the virtual directory name.

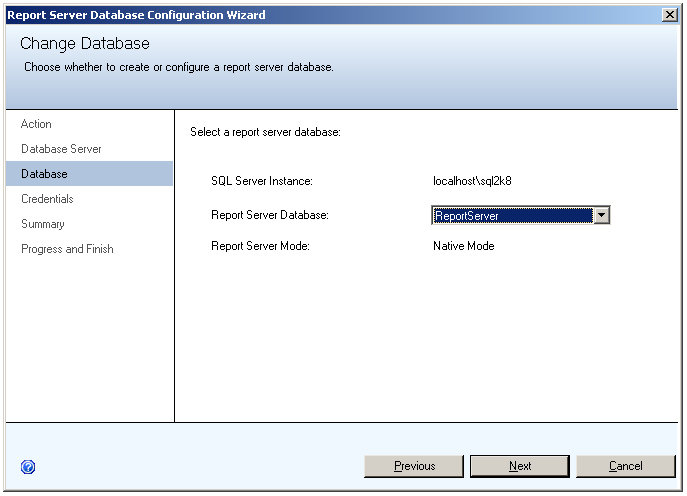


**Figure 14-15:** Web service URL of new SSRS 2008 instance

1. After you have defined the service account and a Web service URL, you must define the new instance’s database setup. You use the configuration tool’s Database option to define the database setup (identifying the server name, database name, and connection credentials) for the new instance.

You can also use this option to upgrade a copy of the report server database used for SSRS 2000 to the new format needed for SSRS 2008. For a migration effort, a backup copy of the report server database used for SSRS 2000 (named ReportServer by default) and the additional database used by SSRS 2000 for temporary database use (named ReportServerTempDB by default) should be restored and then upgraded for SSRS 2008.

* + To start database setup, restore a backup of the two SSRS 2000 report server databases. If you are performing a single-server upgrade, you should restore the databases by using new names to allow the existing instance of SSRS 2000 continued access to its existing databases.
  + When you have restored the databases, use the Database Setup option to connect to and upgrade the restored databases. Click the Connect button to connect to the instance of SQL Server, and use the drop-down list of databases to select the restored report server database (not the restored temporary database). Figure 14-16 shows the Change Database screen after a connection has been established and a restored database selected.



**Figure 14-16:** Configuring database setup

1. When the upgrade process is complete, use the Credentials Type drop-down list to specify how SSRS 2008 should connect to the newly upgraded databases on an ongoing basis. Then, click Apply to have the configuration tool update the database configuration information for the new instance of SSRS 2008.

#### How Schema, Metadata, and Report Server Content Is Updated

The report server database is upgraded in three stages:

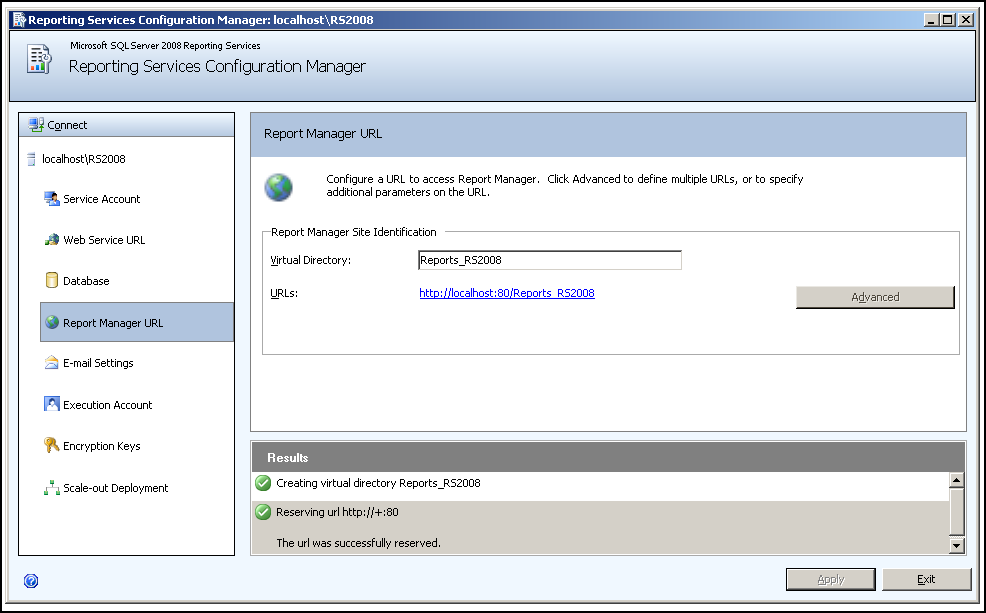
* 1. The schema is upgraded automatically after setup and service startup or when you select a SQL Server 2000 or SQL Server 2005 report server database in the Reporting Services Configuration tool. In addition, the Report Server service checks the database version at startup. If the report server is connected to a database that is an earlier version, the report server will update the database during startup.

Published reports and compiled report snapshots are updated on first use. For more information, see [Upgrading Reports](http://go.microsoft.com/fwlink/?LinkId=134551) in SQL Server 2008 Books Online.

Review [Upgrading a Report Server Database](http://go.microsoft.com/fwlink/?LinkId=134552), in SQL Server 2008 Books Online, for more information.

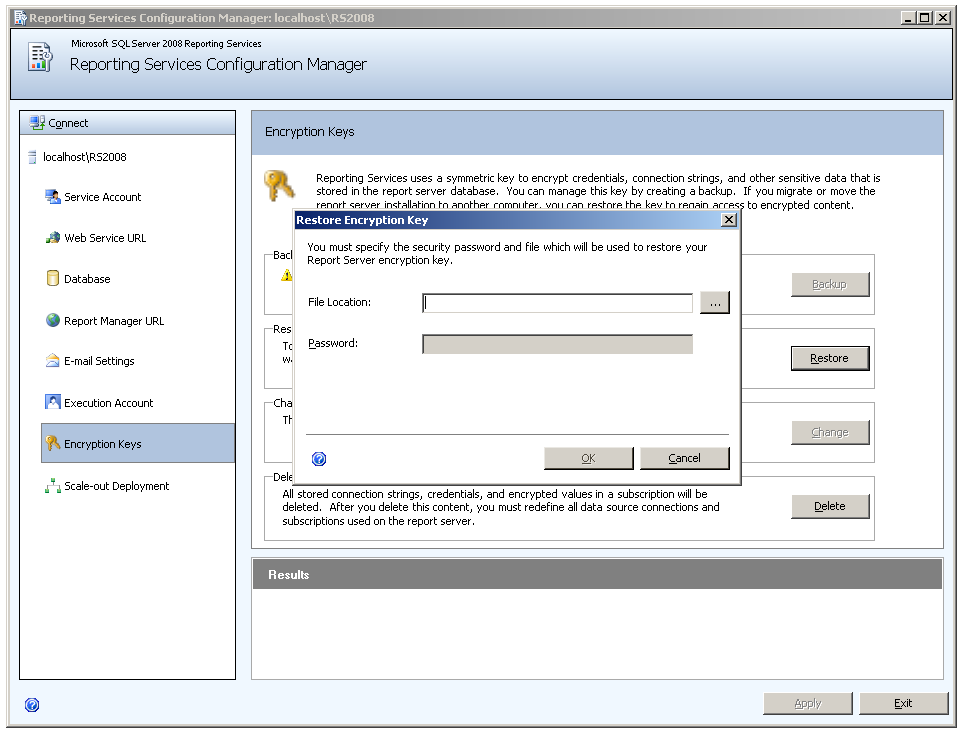
* 1. Use the Report Manager URL option to configure a URL to access Report Manager; you can use the default values provided or specify different values. You can use any names for Report Manager URLs except virtual directory names already in use. Thus, because the original instance of SSRS 2000 is still installed and running, you should not name the virtual directories ReportServer or Reports (the default names for the virtual directories created when SSRS 2000 or SSRS 2005 is installed).

Figure 14-17 shows the Report Manager URL being created, using Reports\_RS2008 as the virtual directory name.



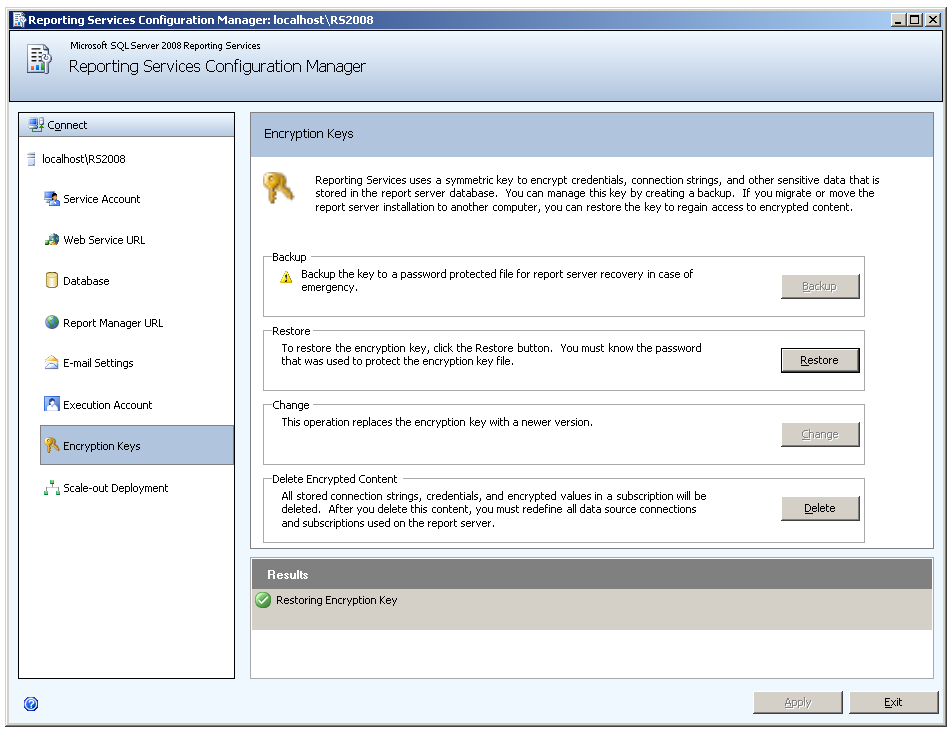
**Figure 14-17:** Report Manager URL setup

1. After the database connection has been established, use the Encryption Keys option to restore the symmetric encryption key extracted and saved as part of the pre-upgrade planning process. Select Restore, and provide the filename and password used with the rskeymgmt utility to extract and save the key from the SSRS 2000 instance. Figure 14-18 shows this option being used to restore a symmetric encryption key.



**Figure 14-18:** Restoring the symmetric encryption key

**Note**: If the symmetric encryption key has not yet been extracted from SSRS 2000, simply do so by using the rskeymgmt utility (found in the <Drive:>*\Program Files\Microsoft SQL Server\80\Tools\Binn\* directoryif running SSRS 2000). If you cannot restore the encryption key for some reason, you will have to use the Delete option to delete existing encrypted content. Note that, in this case, you will have to manually recreate any encrypted content (such as existing data source credentials) after the upgrade. Once the encryption key is restored, Reporting Services Configuration Manager should show a green checkmark in the results area, as Figure 14-19 shows.



**Figure 14-19:** Green checkmark in results area

1. Finally, you can use the configuration tool’s Email Settings and Execution Account options to establish other extended configuration settings for SSRS 2008. Use these options to set these extended options for the new instance.
2. After you use the configuration tool to complete the configuration of the new instance, the new instance should be ready for use. Start the Report Manager interface by using a URL referring to the newly configured virtual directory for the Report Manager application. For example, if the virtual directory is named Reports\_RS2008, you can use the URL http://localhost/Reports\_RS2008 to launch the new version of the Report Manager application. When opened, the report folders and contents (data sources, reports, and other report item files) available within SSRS 2000 should now be present and available within SSRS 2008.

## Upgrading from SQL Server 2005

### In-Place Upgrade

When upgrading an installation of SSRS 2005 in place to SSRS 2008, the upgrade process handles all aspects of the upgrade, automatically updating report server content, report definitions, and component configurations. Note, however, that this upgrade does not automatically handle updates to client workstations and computers that have the Report Designer or management tools installed. You will have to upgrade those workstations and computers after you upgrade the report server.

#### Upgrading via the Setup Application

Upgrading from SSRS 2005 to SSRS 2008 via the Setup application is identical to upgrading from SSRS 2000 via the Setup application. For detailed steps, see the “Upgrading via the Setup Application” section under “Upgrading from SQL Server 2000” earlier in this document.

### Side-by-Side Upgrade

Alternatively, you can upgrade SSRS 2005 installations to SSRS 2008 by using a side-by-side upgrade method. You can perform side-by-side upgrades on a single server (the existing report server) or by using two servers (to take advantage of new hardware, for example).

When you perform the upgrade on a single server, you install a new instance of SSRS 2008 alongside the existing SSRS 2005 installation and then manually move report server content, report definitions, and other configuration information from SSRS 2005 to the new instance.

When you perform the upgrade by using two servers, you install SSRS 2008 on the new server (as the default instance or as a named instance) and then perform the same manual movement of report server content, report designs, and configuration information.

**Note:** Regardless of the upgrade process you use, workstations and computers with the Report Designer or SSRS 2005 management tools installed will have to be upgraded after the report server is upgraded.

#### Installing the New Instance

As with a side-by-side upgrade from SSRS 2000, the first step for a side-by-side upgrade from SSRS 2005 is to install (but not configure) SSRS 2008. The following points should be considered when planning a single-server or a two-server upgrade process:

If an additional server is not available, you can use a single-server upgrade process. During the upgrade process, you must install SSRS 2008 as a named instance. After the upgrade (and testing) is complete, you can rename the named instance to serve as the default instance once you have uninstalled SSRS 2005.

If an additional server is available and will serve as the new SSRS 2008 report server, you can install SSRS 2008 as the default instance or as a named instance on the new server. After the upgrade (and testing) is complete, you can decommission the old report server or reuse it for other purposes.

To begin the upgrade process, start the Setup application for SQL Server 2008 as described in the “Installing the New Instance” section under “Upgrading from SQL Server 2000” earlier in this document and follow the same steps.

#### Configuring the New Instance

After you have installed the new SSRS 2008 instance, you should use the new Reporting Services Configuration tool to configure it. Start the tool from the Configuration Tools group within the Microsoft SQL Server 2008 Start menu group.

The steps for configuring the new instance are the same after an SSRS 2005 upgrade as after an SSRS 2000 upgrade; you can find those steps above in the “Configuring the New Instance” section under “Upgrading from SQL Server 2000.”

## Troubleshooting a Failed Upgrade

If the upgrade process should fail, the first course of action is to review the setup logs created by the Setup application.

Review the Summary.txt file located in the <drive>:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\LOG\ directory. If any error messages are listed, take the required actions to correct the situation and try the upgrade process again.

Each execution of Setup will generate a new time-stamped log folder. For example, if you start the SQL Server Installation Center page, it gets its own time-stamped log folder, and each Setup action invoked from that page gets its own as well, so you will probably see several time-stamped log folders in this directory. The time-stamped log folder name format is YYYMMDD\_hhmmss.

You can find detailed Setup logs at the following location: <drive>:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\LOG\.

When looking for errors in the detail log, search for the following phrases:

* Watson bucket
* Error:
* Exception has been

A typical Setup request goes through three execution phases:

1. Global rules check
2. Component update
3. User-requested action

Each of these phases will generate detail and summary logs, with additional log files being generated as appropriate. Setup is called at least three times per user-requested Setup action.

Typical log files generated are:

* Detail\_GlobalRules.txt
* Detail\_ComponentUpdate.txt
* Detail.txt

The summary log file name format is Summary\_[machine name]\_timestamp\_[execution phase]. The final summary log is copied to %Program Files%\Microsoft SQL Server\100\setup bootstrap\log folder and named Summary.txt for quick reference.

**Note:** Setup will not CAB the log files unless there is an error.

Windows Installer (MSI) actions performed during Setup generate their own log files in the following format: [product feature]\_[cpu]\_[LCID (optional)]\_[attempt #].log. If an MSI execution fails, look in the associated MSI log for “return value 3” only for ENU versions.

Datastore files contain a snapshot of the state of all configuration objects being tracked by the Setup process and are useful for troubleshooting configuration errors. XML file dumps are created for datastore objects for each execution phase. They are saved in their own log subfolder under the time-stamped log folder, as follows:

* Datastore\_GlobalRules
* Datastore\_ComponentUpdate
* Datastore

For more information, see [How to: View SQL Server 2008 Setup Log Files](http://go.microsoft.com/fwlink/?LinkID=133952) in SQL Server 2008 Books Online.

## Post-Upgrade Tasks

After upgrading to SSRS 2008 from either SSRS 2000 or SSRS 2005, it is important to ensure that the upgrade ran successfully and to configure SSRS 2008.

1. To begin, particularly if you performed an in-place upgrade, use the Report Server Configuration Tool to check the configuration of the report server. After the tool is launched, connect to the upgraded instance.
2. Review the configuration settings by selecting each of the items in the left pane of the tool. If any of the settings seem incorrect or are missing, update the settings and save the changes.
3. Ensure that the report server is behaving as expected by running a sample set of the reports deployed to the server. Start Report Manager by using the correct URL (for example, http://locahost/Reports for an upgraded default instance or http://localhost/Reportsnew for a newly installed and configured named instance). At a minimum, you should select and execute reports to verify that the following report server features and capabilities (if used) are working correctly:

* **Standard and custom data extensions.** You should execute reports against all defined data sources (using standard data providers or custom data extensions) to ensure that each is working as expected.
* **Security credentials.** Run reports that rely on each of the security credential options that can be used for connecting to a data source: credentials supplied by the user running the report, credentials stored securely in the report server, and Windows integrated security.
* **Subscriptions.** You should review report subscriptions to ensure that their settings are still applicable, and you should test each to verify that it completes successfully.
* **Custom rendering and delivery extensions**. You should fully test any custom rendering and delivery extensions to ensure that each is working correctly. Remember, you must recompile all custom extensions created for SSRS 2000 or SSRS 2005 to use the Common Language Runtime (CLR) provided with Visual Studio 2008.
* **Custom report assemblies.** If any reports include references to custom assemblies, you should test the reports to ensure the custom assemblies continue to function as designed. As just noted, you must recompile all custom assemblies created for reports within SSRS 2000 or SSRS 2005 to use the Visual Studio 2008 CLR.

1. Finally, SQL Server 2005 and 2008 Reporting Services comes with a new ad hoc reporting tool called Report Builder. If you want to use this new feature, you need to change the existing security role definitions to provide end-user access to Report Builder. Consider updating the existing role definitions as Table 14-5 shows.

Table 14-5: Role Updates After the SSRS Upgrade

| **Existing Role Definition** | **Suggested Changes** |
| --- | --- |
| Browser | Add View Models to grant permission to view published Report Builder models. |
| Content Manager | Add Manage Models, View Models, and Consume Reports to grant full permission over models and to provide the ability to create and modify reports in Report Builder. |
| Publisher | Add Manage Models to grant permission to create, view, and delete Report Builder models. |
| System Administrator | Add Execute Report Definitions to run reports using Report Builder. |
| System User | Add Execute Report Definitions to run reports using Report Builder. |

Moving Reports Between SSRS 2000 and SSRS 2005, or SSRS 2005 and SSRS 2008

When you upgrade an SSRS 2000 or SSRS 2005 instance to SSRS 2008 using the procedures this chapter discusses, all the report server content is moved to the new instance and upgraded. In some unique cases, it might be more appropriate to migrate individual reports to a new SSRS 2008 instance in a more controlled fashion. For example, if a single report server supports a varied group of users, with reports developed by different development groups, a staged move of reports and users to a new instance of SSRS might be a suitable course of action. Additionally, if a set of complex reports requires additional testing efforts, migrating each report individually could be beneficial.

After you have installed a new instance of SSRS 2008, you can migrate reports by using one of the following three options:

**Manually move existing files deployed to the SSRS 2000 or SSRS 2005 instance to the new SSRS 2008 instance.** Using the Report Manager application for SSRS 2000 or SSRS 2005, you can save files within a given folder to the file system on a one-by-one basis. You do this by using the Edit option when viewing the properties of a given report. The Edit option returns an RDL file for each report, which you can save and then upload to the new instance using Report Manager for SSRS 2008. Note that you need to manually recreate within the new instance data sources used by each report, and you need to configure uploaded reports to use the new data sources (because the uploaded reports will not automatically see new data sources even if they have the same name as their counterparts within the SSRS 2000 or SSRS 2005 instance).

You will also need to manually upload other files (such as images) to the new instance, placing them within the same folder structure and using the same names as their counterparts within SSRS 2000 or SSRS 2005. As RDL files are uploaded and configured, you can test each report individually to ensure that it works correctly. You can then correct any deficiencies within the RDL file by using the new Report Designer tools within BI Development Studio.

**Move reports, data sources, and resources by using the SSRS script host (rs.exe).** By using the Web services APIs, you can extract all the report definitions from the SSRS 2000 or SSRS 2005 instance to the local file system and then republish them to the SSRS 2008 instance.

**Use BIDS to deploy existing report projects to the new SSRS 2008 instance.** Using BI Development Studio, you can open and deploy report projects that were originally created through Report Designer for SSRS 2000 or SSRS 2005 to the new SSRS 2008 instance. When using BI Development Studio, you can deploy data sources (as well as other files included in the report project) along with the reports. This approach might provide a simpler and more efficient way of deploying groups of reports to the new instance. As reports are deployed, you can test each report to make sure it works as designed. And you can correct any deficiencies within BI Development Studio, redeploying fixed reports as needed.

Note that deploying individual reports to a new instance of SSRS 2008 will not automatically migrate the metadata related to any given report. For example, report history and execution settings, parameter defaults, subscription definitions, and security settings defined for the report within SSRS 2000 or SSRS 2005 will not be automatically recreated within SSRS 2008. You need to manually configure all these settings after you deploy the report to the new report server. Before proceeding with this approach to moving reports from SSRS 2000 or SSRS 2005, ensure that all the additional settings for each report are well known and documented so that you can recreate the settings once each report is deployed to SSRS 2008.

### Deploying Custom Extensions and Assemblies

If your installation includes custom report items, assemblies, or extensions, you must redeploy the custom components. If you are not using custom components, you can skip this section.

If you deployed and used any custom extensions or custom assemblies for reports with SSRS 2000 or SSRS 2005, you need to redeploy the extensions or assemblies for use with SSRS 2008, as follows:

1. You need to recompile each extension or assembly by using Visual Studio 2008. This action ensures that each extension or assembly uses the CLR compatible with SSRS 2008.
2. After you have recompiled each extension or assembly, you need to deploy it for use with SSRS 2008. This typically involves putting a copy of the compiled extension or assembly file within the correct directory under the SSRS 2008 installation directory and updating one or more configuration files. For details, see [How to: Deploy a Custom Report Item](http://go.microsoft.com/fwlink/?LinkId=134553) in SQL Server 2008 Books Online.

To redeploy the custom components, follow these steps:

1. Determine whether the assemblies are supported or need recompilation:

* You must recompile custom authentication extensions created for the SQL Server 2000 release.
* You must rewrite custom rendering extensions for SSRS 2008 by using the ROM.
* HTML 3.2 and HTML OWC renderers are not supported in SSRS 2008.
* Other custom assemblies should not require recompilation.

1. Move the assemblies to the new report server and Report Manager \bin folders. In SQL Server 2008, the report server binaries are located in \Program files\Microsoft SQL Server\MSRS10.MSSQLSERVER\Reporting Services\ReportServer\bin for the default SSRS 2008 instance.
2. Modify the configuration files to add entries for your custom component. The entries will vary depending on the kind of assembly you are using. For instructions about where to place files and add configuration entries, see the following SQL Server 2008 Books Online topics:
   * [Deploying a Custom Assembly](http://go.microsoft.com/fwlink/?LinkId=134554)
   * [How to: Deploy a Custom Report Item](http://go.microsoft.com/fwlink/?LinkID=134553)
   * [Deploying a Data Processing Extension](http://go.microsoft.com/fwlink/?LinkId=134556)
   * [Deploying a Delivery Extension](http://go.microsoft.com/fwlink/?LinkId=134557)
   * [Deploying a Rendering Extension](http://go.microsoft.com/fwlink/?LinkId=134558)
   * [Implementing a Security Extension](http://go.microsoft.com/fwlink/?LinkId=134559)

### Verifying Configuration Files

In some cases, configuration changes made within SSRS 2000 or SSRS 2005, particularly those made manually within the configuration files, are not created automatically by the new SSRS 2008 configuration tool. Thus, it is a good idea to compare each of the configuration files between the old and new instances, looking for any configuration differences that might have been made manually for SSRS 2000 or SSRS 2005. You should compare each SSRS 2000 or SSRS 2005 configuration file that you saved as part of the pre-upgrade planning process to its new counterpart.

A default installation of SSRS 2000 uses the *<SQL Install Dir>*\MSSQL\Reporting Services directory, storing configuration information under the ReportServer and ReportManager subdirectories. A default installation of SSRS 2005 or SSRS 2008 uses the *<SQL Install Dir*>\MSSQL.*<N>*\Reporting Services directory, where *<N>* is a number indicating the order of installation of SQL Server 2008 components. For example, if Reporting Services is the first SQL Server 2008 component installed, the directory will be *<SQL Install Dir*>\MSSQL.1\Reporting Services. Configuration information is then stored under the same ReportServer and ReportManager subdirectories. The pre-upgrade planning section above lists the files to check and compare.

### Uninstalling SSRS 2000 or SSRS 2005

If you performed a side-by-side upgrade on a single server, you can use the new SSRS 2008 instance as configured alongside your SSRS 2000 or SSRS 2005 instance. However, at some point, you should uninstall your previous SSRS instance. At that point, you can rename the virtual directories you created for SSRS 2008 to use the names originally configured for SSRS 2000 or SSRS 2005. End users and applications that reference the report server can then continue to use the original URLs and connection information as opposed to the virtual directory names you assigned to SSRS 2008 for upgrade purposes.

1. You can uninstall SSRS 2000 or SSRS 2005 by using the Add or Remove Programs option within Control Panel. The report server software (and other associated files installed with SSRS 2000 or SSRS 2005) will be removed from the server. This action should not affect SSRS 2008 or any other installed software.
2. After you have removed SSRS 2000 or SSRS 2005 from the server, you can delete the original report server databases from SQL Server. Ensure that you delete the databases associated with your old SSRS version—not those restored and upgraded for use with SSRS 2008.  
   **Note**: You should retain good backups of these databases for a period of time just in case you need to resurrect the SSRS 2000 or SSRS 2005 implementation.
3. After you have removed SSRS 2000 or SSRS 2005, modify the virtual directory names assigned to SSRS 2008 to use the names assigned to the virtual directories for SSRS 2000 or SSRS 2005. To do this, start the Reporting Services Configuration tool (as described in the section about configuring the new instance of SSRS 2008) and connect to the new instance.
4. Use the Report Server Virtual Directory option to create a new virtual directory for the report server, using the name of the SSRS 2000 or SSRS 2005 report server virtual directory. For example, if SSRS 2000 or SSRS 2005 used the default name of ReportServer for the report server virtual directory, create a new virtual directory with that name for the SSRS 2008 report server.
5. Use the Report Manager Virtual Directory option to create a new virtual directory for the Report Manager application. For example, if SSRS 2000 or SSRS 2005 used the default name of Reports for the Report Manager virtual directory, create a new virtual directory with that name for the Report Manager application.

These steps reconfigure Reporting Services 2008 to use the virtual directory names originally associated with SSRS 2000 and SSRS 2005. For more information, see [How to: Migrate a Reporting Services Installation](http://go.microsoft.com/fwlink/?LinkID=134498) in SQL Server 2008 Books Online.

These steps reconfigure SSRS 2008 to use the virtual directory names originally associated with SSRS 2000 or SSRS 2005. When this process is completed, you can delete the virtual directories you created when configuring the new instance of SSRS 2008 by using the IIS Manager application.

## Conclusion

The key to a successful SSRS upgrade is a detailed, well-thought-out upgrade plan, including a review of possible upgrade issues and a rollback strategy in case of a failed upgrade. In planning for a rollback, you should include at least backups of the following elements:

* Databases, applications, and configuration files
* Reporting Services encryption key
* Customized IIS configuration files
* Customized extensions and assemblies folder

Use Upgrade Advisor to help discover blocking issues related to SSRS. And determine the appropriate upgrade method for your organization and configuration. This chapter discusses several advantages and disadvantages of using each method for upgrading SSRS. For example, the side-by-side method is easy to roll back because your original instance remains intact, whereas an in-place upgrade might be faster, but you would have to restore the previous instance in case of a failed upgrade.

By following the preparation guidance and upgrade steps in this chapter, you should have a smooth transition to SSRS 2008.

## Additional Resources

For an up-to-date collection of additional references for upgrading Transact-SQL queries to SQL Server 2008, see the [[[Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkID=132510)](http://go.microsoft.com/fwlink/?LinkID=132510)](http://msdnstage.redmond.corp.microsoft.com/en-us/library/cc936623.aspx).

[SQL Server Web site](http://go.microsoft.com/fwlink/?LinkId=132512)

[SQL Server Developer Center](http://go.microsoft.com/fwlink/?LinkId=132514)

[SQL Server Reporting Services Web site](http://go.microsoft.com/fwlink/?LinkID=133955)

# Other Microsoft Applications and Platforms

## Introduction

SQL Server serves as the data server at the back end of lots of Microsoft applications. When you upgrade these applications or the back-end server, you have to ensure continued effective and efficient interaction between the layers during and after the upgrade or ensure that the services that are required by the application layer can be reinstated if there is an adverse effect on either. In this chapter, we investigate the effect of upgrading applications that rely on a functionally effective data service from SQL Server.

This chapter covers the following applications:

* + Microsoft Small Business Server 2008
  + Microsoft Office Communications Server 2007
  + Microsoft Office SharePoint Server 2007
  + Microsoft System Center: Data Protection Manager 2007
  + Microsoft Dynamics

For a full list of Microsoft products that use SQL Server technology and that might be affected by an upgrade, see the [Microsoft Products Web site](http://go.microsoft.com/fwlink/?LinkID=133957).

## Microsoft Windows Small Business Server 2008

Microsoft Windows Small Business Server 2008 (Windows SBS 2008) is scheduled for release on November 12, 2008. Windows SBS 2008 is a prepackaged small-business solution that contains a customized version of the Windows operating system. This includes services and features that might otherwise be sold as separate applications. In addition to Windows SBS 2008, Microsoft currently also supports Windows SBS 2003 and SBS 2000.

Windows SBS 2008 includes a preconfigured version of the Windows Server 2008 operating system, designed for up to 75 users. For more information about different Windows SBS 2008 editions, see the Windows SBS 2008 feature comparison table at [Compare Features](http://go.microsoft.com/fwlink/?LinkId=133960).

### Preparing to Migrate to Windows SBS 2008

Windows SBS supports a side-by-side migration method, in which you move settings and data from the legacy server or servers to a new set of servers.

**Note:** You should always back up your database data before you try a Windows SBS migration. For information about backup tools, see the "Data Protection Manager" section later in this chapter.

Some high-level steps for migrating to this new environment are covered in the "Migrating to Windows SBS 2008" section in this chapter. That section provides links to more details for each migration step.

You can migrate from SBS 2000 or Windows SBS 2003 to Windows SBS 2008, but you must use two separate servers. After you transfer the Windows SBS functional elements between the two servers, you can start replacing the old server.

#### Migration Preparation Tool

Windows SBS 2008 provides a Migration Preparation Tool to help you with the networking component of a migration from Windows SBS 2003. For more information, see [Run the Migration Preparation Tool](http://go.microsoft.com/fwlink/?LinkId=133963).

#### Windows SBS 2008 and SQL Server 2008

A copy of SQL Server 2008 Standard for Small Business is supplied with Windows SBS 2008 Premium. This edition functions only if you install it in a Windows SBS 2008 domain.

#### Preparing Your Windows SBS Source Server for Migration

You must make sure that the source server and network are ready for migration. The TechNet guide [Migrate to Windows Small Business Server 2008 from Windows Small Business Server 2003](http://go.microsoft.com/fwlink/?LinkId=133966) helps you with the following steps:

* Backing up the source server
* Evaluating the source server’s system health
* Installing the most recent service packs and fixes
* Verifying the network configuration
* Raising the functional level of the Active Directory Domain Services (AD DS) domain and forest

You must also run the Migration Preparation Tool on the source server. It updates the AD DS schema, installs an update that extends the time limit for the migration, and configures Microsoft Exchange Server to support migration.

In addition, see [Prepare the Source Server for migration](http://go.microsoft.com/fwlink/?LinkId=133969), which covers the steps that you have to take to make sure that the settings and data on the source server migrate successfully to the destination server.

#### Backup and Rollback

You should always back up your SQL Server data before a migration. Because Windows SBS is a complete environment migration, as of this writing, the Windows SBS team has no automated support for rolling back a Windows SBS 2008 migration. However, make sure that you carefully watch the work being done by the Data Protection Manager (DPM) team that could fill this gap (for more information, see the "Data Protection Manager" section later in this chapter).

For information about rolling back SQL Server data, see the relevant chapters in this guide, including

* Chapter 1, "Upgrade Planning and Deployment"
* Chapter 2, "Relational Engine"
* Chapter 11, "Analysis Services"

### Migrating to Windows SBS 2008

The general migration tasks for moving to Windows SBS 2008 are as follows:

1. **Create a migration answer file**. Windows SBS 2008 Setup uses an answer file to automate the installation and to run Setup in migration mode. [Create a migration answer file](http://go.microsoft.com/fwlink/?LinkId=133973) introduces you to the migration answer file and guides you through how to use the Answer File Tool to create the migration answer file.
2. **Install Windows SBS 2008 in Migration Mode.** [Install Windows Small Business Server 2008 in Migration Mode](http://go.microsoft.com/fwlink/?LinkId=133976) explains how to use the migration answer file to install Windows SBS 2008 on the destination server in migration mode.
3. **Migrate settings and data to the destination server.** The MigrationWizard helps you migrate settings and data from the source server to Windows SBS 2008. [Migrate settings and data to the Destination Server](http://go.microsoft.com/fwlink/?LinkId=133979) explains how to use the Migration Wizard and provides information about the settings and data that you can migrate.
4. **Demote and remove the source server from the network.** After you have installed Windows SBS 2008 and have successfully migrated all the settings and data, you must demote and physically remove the source server from the network. For detailed instructions, see [Demote and remove the Source Server from the network](http://go.microsoft.com/fwlink/?LinkId=133981).
5. **Delete the old Folder Redirection Group Policy object.** This is the final task to rehome the redirected folders to the destination server. Perform this task only if you had folder redirection enabled on the source server. For instructions, see [Delete the old Folder Redirection Group Policy object](http://go.microsoft.com/fwlink/?LinkId=133984).

### Post-Migration Tasks

After you finish migrating all settings and data to Windows SBS 2008, you might want to map permitted computers to user accounts, enable folder redirection, configure POP3 connectors, or update mailbox quotas on your new server. For more information, see [Optional post-migration tasks](http://go.microsoft.com/fwlink/?LinkId=133992).

## Microsoft Office Communications Server

The next release of Office Communications Server (OCS)—OCS 2007 Release 2 (R2)—will support SQL Server 2008 in addition to SQL Server 2005. The expected release date is 2009. This release will fully support SQL Server 2008 as a data server.

As of this writing, OCS is currently at release OCS 2007. The earlier version was known as Live Communications Server (LCS) 2005. Neither OCS 2007 nor LCS 2005 supports SQL Server 2008.

**Note:** Although OCS 2007 R2 will support both the 32-bit and 64-bit versions of SQL Server 2008, SQL Server 64-bit is preferred. OCS 2007 R2 will support the x64 versions of SQL Server 2008, not the IA-64 Itanium versions. In addition, when you use a 64-bit version of the Windows operating system, you will also be required to use the 64-bit version of SQL Server 2008. OCS 2007 R2 will also support SQL Server 2005 Service Pack 2 (SP2).

For more information about database support for OCS 2007, see [Microsoft Office Communications Server 2007: Database Topologies](http://go.microsoft.com/fwlink/?LinkId=133994).

### Preparing to Upgrade to OCS 2007 R2

Upgrading to OCS 2007 R2 requires you to consider the Windows operating system, OCS 2007, and SQL Server.

#### Upgrading the Operating System

As of this writing, an in-place upgrade of Windows with OCS installed is not supported. OCS 2007 R2 will support Windows Server 2008 x64 and Windows Server 2003 x64 only. However, you must install OCS 2007 R2 on a new Windows Server 2008 server to run OCS 2007 R2 on Windows Server 2008 system.

#### Upgrading to OCS R2

There are two main scenarios for upgrading/migrating from OCS 2007 to OCS 2007 R2, as follows. Data migration is not performed by moving data directly between OCS SQL Server back-end servers, but instead is performed at the OCS level only.

* **Database export/uninstall/reinstall/database import scenario for upgrades from OCS 2007 to OCS 2007 R2.** This method backs up user data (Contact lists per user); performs a full uninstall, operating system upgrade, and reinstall, and then imports the previously exported data. Currently, you can also move from a SQL Server 2005 SP2 OCS back-end database to a SQL Server 2008 one.
* **OCS side-by-side migration scenario.** This method installs a deployment of OCS 2007 R2 into an existing OCS 2007 or LCS 2005 SP1 deployment on a separate set of servers, with additional back-end SQL Server database instance(s). For the newly or additionally deployed instance of SQL Server, you can decide to use SQL Server 2005 SP2 or SQL Server 2008. The migration is performed by using the OCS "Move user" functionality.

#### Upgrade Tool for OCS 2007

When you move from OCS 2007 to OCS 2007 R2, you can save the user data to the operating system by using the DPimpexp.exe tool. After you have uninstalled the original release and installed the new release, you can use the same tool to repopulate the new release with the extracted data. This tool is data server-agnostic. Therefore, it will work with either SQL Server 2005 or SQL Server 2008 as data servers.

### Upgrading OCS 2007 R2 to Use SQL Server 2008

You can upgrade the SQL Server version that OCS 2007 R2 uses from SQL Server 2005 to SQL Server 2008 by using either an in-place or side-by-side upgrade.

#### In-Place Upgrade

You can update the OCS instance data by doing an in-place upgrade from SQL Server 2005 to SQL Server 2008. Microsoft expects that OCS 2007 R2 deployments using SQL Server 2005 and migrating to SQL Server 2008 will probably use the in-place upgrade method for their Database Engine, with some downtime expected for OCS upgrading. Although this method requires some downtime, which could affect the business and might not work for enterprise users, Microsoft considers an in-place upgrade the most common approach.

#### Side-by-Side Upgrade

Alternatively, especially if you cannot afford the downtime, you can do a full side-by-side upgrade of OCS 2007 R2 running SQL Server 2005 to OCS 2007 R2 running SQL Server 2008. In this scenario, you implement two OCS servers together with their applications and data servers intact. One of the servers has SQL Server 2005 installed, and the other server has SQL Server 2008 installed.

Then you use a method called "Move User," where individual user data is moved from the OCS data server of one instance to the OCS data server of the other. No SQL Server tools are used to transfer the user data. For example, a first instance of the OCS service could be using SQL Server 2005 as the data server while the second instance of the OCS service is using a SQL Server 2008 data server for its functionality. For more information, see the "Database topologies" section of the [Office Communications Server 2007 Document: Supportability Guide](http://go.microsoft.com/fwlink/?LinkId=133995), and the [Office Communications Server](http://go.microsoft.com/fwlink/?LinkId=133996).

This is the typical migration scenario for enterprise customers. This move has no downtime and enables controlled migration of users—controlled to the extent that if something goes wrong, the original system is still intact for rollback purposes. This kind of staged migration also helps with the client upgrade rollout by minimizing the effect of the upgrade on the business.

**Note:** OCS requires multiple instances of SQL Server that serve as the main SQL Server back end, the archiving database, and the Monitoring Server database.

### Rollback Options and Tools

The side-by-side upgrade option helps provide the safety of having your legacy system available in case a rollback is needed. As of this writing, there are no specific tools available from the OCS product team, but see the "Data Protection Manager" section later in this chapter for support provided by DPM.

## Microsoft Office SharePoint Server 2007

Microsoft Office SharePoint Server (MOSS) 2007 is one of the most widely-used SQL Server applications. As of this writing, MOSS 2007 supports SQL Server 2008 in addition to SQL Server 2005 and SQL Server 2000 as back-end data servers.

For more information about how to use SQL Server 2008 with Windows SharePoint Services, see [Determine hardware and software requirements (Windows SharePoint Services)](http://go.microsoft.com/fwlink/?LinkID=79264).

**Note:** You must install Windows SharePoint Services 3.0 SP1 or later versions before you install SQL Server 2008 for use with Windows SharePoint Services.

For information about how to use SQL Server 2008 with MOSS, see [Determine hardware and software requirements (Office SharePoint Server)](http://go.microsoft.com/fwlink/?LinkID=79263).

**Note:** You must install both Windows SharePoint Services 3.0 SP1 or later versions and MOSS 2007 SP1 or later versions before you install SQL Server 2008 for use with MOSS.

### Preparing to Upgrade MOSS 2007 to SQL Server 2008

MOSS 2007 currently has no requirements for SQL Server that would bind it to releases earlier than SQL Server 2008.

**Note**: The MOSS 2007 application functions independently of the SQL Server data server. However, to use all the features of MOSS 2007 with the 2007 Microsoft Office system, you must be using at least SQL Server 2000 SP4 or SQL Server 2005.

#### Upgrading to MOSS 2007

SharePoint is a complex application, and upgrading to MOSS 2007 requires planning. For more information, see [Migration and Upgrade Resource Center for SharePoint Server 2007](http://go.microsoft.com/fwlink/?LinkId=133997).

#### Upgrading Windows with MOSS 2007

You can upgrade Windows with MOSS 2007 with minimal effect. The only aspects of the operating system that MOSS 2007 uses are ASP.NET and IIS. As long as these are backwardly compatible after the upgrade, there will be no effect.

For more information about the changes to ASP.NET and IIS because of upgrading Windows, see the relevant documentation and make sure that backward compatibility is guaranteed. For more information, see [Determine hardware and software requirements (Office SharePoint Server)](http://go.microsoft.com/fwlink/?LinkID=79263).

#### Upgrade Tools

There are no tools within MOSS to make it easier to upgrade from an earlier version of SQL Server to the latest version. However, you can find details about the steps that you must follow to move databases between different versions of MOSS in the TechNet article [Migrate databases](http://go.microsoft.com/fwlink/?LinkId=133998).

#### Backup and Rollback

You should always back up your SQL Server and SharePoint data before an upgrade. DPM 2007, which we cover later in this chapter, provides extensive support for backing up all files and databases of a MOSS 2007 server farm.

If you are upgrading only to SQL Server 2008, make sure that you back up your data first. If it is possible, test the upgrade on a test system before you try the upgrade in production. (See the other chapters in this guide for specific, detailed information about how to upgrade each SQL Server component to SQL Server 2008.)

#### Upgrading to SQL Server 2008

As of this writing, the latest features of MOSS 2007 that function together with Microsoft Office 2007, are available only through data server features released with SQL Server 2008. Therefore, if you want any of these features, you must also upgrade the data server.

Upgrading the MOSS 2007 database server to SQL Server 2008 does not affect the MOSS 2007 application.

**Note: The one SQL Server component that SharePoint cannot use is replication.** There are no other restrictions in the services that the data server can provide.

## Microsoft System Center 2007

The Microsoft System Center 2007 family of management products helps IT professionals manage their Windows Server infrastructure. The tools are especially useful in midsize to large data environments. For comprehensive information about System Center, see the [Microsoft System Center Web site](http://go.microsoft.com/fwlink/?LinkId=133999).

Two System Center 2007 products are especially relevant for SQL Server 2008: Microsoft Operations Manager (MOM) and Data Protection Manager (DPM).

### Operations Manager: Management Pack for SQL Server 2008

MOM 2007 works seamlessly with Microsoft software and applications, including SQL Server 2008, which enables more control of the IT environment. MOM supports monitoring with its targeted Management Packs.

SQL Server Management Packs provide proactive and reactive monitoring of SQL Server 2008 in addition to SQL Server 2005 and SQL Server 2000. You can proactively manage your instances of SQL Server and discover issues before they become critical. You can use the Management Pack for the particular SQL Server version that you are running to increase the security, availability, and performance of your SQL Server infrastructure.

Availability and configuration monitoring, performance data collection, and default thresholds are built for enterprise-level monitoring. Both local and remote connectivity checks help ensure database availability. The MOM Management Pack for SQL Server 2008 includes the following features:

* Monitoring the state of different SQL Server 2008 services such as SQL Server, SQL Server Agent, Report Server, and Notification Services
* Monitoring the state of databases
* Monitoring the available space in databases, configurable by % or MB
* Making sure that databases are configured correctly
* Making sure that clients can connect to SQL Server
* Monitoring blocked processes
* Watching for failed SQL Server Agent jobs and jobs taking an too much time to execute
* Monitoring the health of replication and alerting on failures
* Monitoring the state of database mirroring

You can download the Management Pack for SQL Server 2008 at [Microsoft SQL Server Management Pack for Microsoft Operations Manager 2005](http://go.microsoft.com/fwlink/?LinkId=134000).

### Data Protection Manager

Data Protection Manager (DPM) 2007 sets a new standard for Windows backup and recovery, delivering continuous data protection for Microsoft application and file servers to a seamlessly integrated secondary disk and tape solution on the DPM server. DPM enables rapid and reliable recovery through advanced technology for enterprises of all sizes.

DPM provides a rich feature set for protecting SQL Server data, helping you guarantee the ability to recover databases if there is server or database loss. DPM 2007 uses the SQL Server Volume Shadow Copy Service Writer to provide block-level synchronization for database backups, in combination with transaction log backups. After an initial baseline copy of database data is taken, the following two parallel processes enable continuous data protection with integrity:

* Transaction logs are continuously synchronized to the DPM 2007 server, as frequently as every 15 minutes.
* An "express full" backup uses the SQL Server Volume Shadow Copy Service Writer to discover files that need to be backed up for a selected database. Also, the DPM volume filter tracks which blocks have changed in the whole production database and sends just the updated blocks or fragments for backing up database data. By using the SQL Server writer and the Volume Shadow Copy Service infrastructure, DPM guarantees that a complete and consistent image of the data files is backed up onto the DPM server or appliance. DPM 2007 maintains up to 512 shadow copies of the full SQL Server database(s) and associated transaction logs by storing only the differences between any two images.

DPM uses an instance of SQL Server 2005 to support configuration and system data.

You can find more information about DPM’s features and functionality on the [Microsoft System Center Data Protection Manager Web site](http://go.microsoft.com/fwlink/?LinkId=134001).

**Note:** As of this writing, DPM 2007 does not support SQL Server 2008 as its data repository, the specialized database server that DPM 2007 uses to track its operations. DPM 2007 relies on SQL Server 2005 support for the Windows Management Interface (WMI) entries and resource placement in the operating system. In addition, DPM 2007 requires features of SQL Server 2005 Reporting Services.

#### DPM 2007 and Upgrading to SQL Server 2008

When you are upgrading to SQL Server 2008 from either SQL Server 2000 or SQL Server 2005, you can use DPM 2007 during the upgrade process to protect your SQL Server 2000 and SQL Server 2005 data. Used with an in-place upgrade, DPM 2007 will give you a secure point to which you can roll back to a SQL Server 2000 or SQL Server 2005 server in case the upgrade is unsuccessful.

#### DPM 2007 and Data Protection: SQL Server 2008 Databases

If you want to use DPM 2007 to protect SQL Server 2008 databases, make sure that you have upgraded to hotfix KB949779:

* If you are using an x86-based operating system, download the [System Center Data Protection Manager 2007 Feature Pack (x86)](http://go.microsoft.com/fwlink/?LinkId=134002).
* If you are using an x64-based operating system, download [System Center Data Protection Manager 2007 Feature Pack (x64)](http://go.microsoft.com/fwlink/?LinkId=134003).

To read the DPM program manager’s blog, that introduces these fixes and links to more details about issues that might arise from applying this rollup, see [System Center Data Protection Manager 2007 Feature Pack (x64)](http://go.microsoft.com/fwlink/?LinkId=134003). For a valuable Webcast about SQL Server and DMP 2007, see [How to protect SQL SERVER with DPM 2007](http://go.microsoft.com/fwlink/?LinkId=134004).

As of this writing, seamless backup of mirrored configurations is not available. However, this functionality will be provided in DPM 2007 SP1 (expected for release at the end of 2008).

## Microsoft Dynamics

The Microsoft Dynamics products consist of a set of integrated solutions for Financials, Supply Chain, and Customer Relationship Management. The products include the following: Dynamics AX, Dynamics CRM, Dynamics GP, Dynamics NAV, Dynamics SL, Dynamics Retail Management System, and other applications.

Each current Dynamics product supports SQL Server 2008 but has very specific requirements for Windows versions, SQL Server version, product service/feature packs, and so on.

Generally, you should upgrade a Dynamics application’s database server to SQL Server 2008 only after you follow specific guidance from your Dynamics Technical Account Manager and by reviewing information found on the different Dynamics support Web sites. If you are a registered Dynamics user, you can find this information at [https://mbs.microsoft.com/partnersource/products](http://go.microsoft.com/fwlink/?LinkId=134005).

## Conclusion

As with any upgrade, planning is important for moving to any of these latest product versions. Even if there is no specific upgrade tool to help with the upgrade of some of these applications, you can turn to the Microsoft System Center products that can help in all environment, application, and data server upgrades. Before you start, you can use the System Center DPM to create differential backups during the upgrade process in addition to a base backup.

It is also important that the performance of the new system at least equals the performance of the existing system. MOM can help in regression testing the new environment post-upgrade. This gives you the information your organization has to have to decide whether to go forward with the upgrade or roll back, depending on the outcome of the tests.

## Additional References

For an up-to-date collection of additional references for upgrading any of these Microsoft applications, especially in association with SQL Server, see the [Microsoft SQL Server 2008 Upgrade Resources page](http://go.microsoft.com/fwlink/?LinkId=132510).

# Appendix 1: Version and Edition Upgrade Paths

Table A-1, taken from [Version and Edition Upgrades](http://go.microsoft.com/fwlink/?LinkID=132447) in SQL Server 2008 Books Online, shows the paths that SQL Server 2008 Setup will allow when directly upgrading a SQL Server 2000 or SQL Server 2005 instance by using the in-place upgrade method.

**Table A-1: Supported Paths for an In-Place Upgrade from SQL Server 2000 or SQL Server 2005 to SQL Server 2008**

| **Upgrade from** | **Supported Upgrade Paths** |
| --- | --- |
| SQL Server 2000 (32-Bit) MSDE SP41 | SQL Server 2008 Express |
| SQL Server 2000 (32-Bit) Standard SP41,4 | SQL Server 2008 Standard  SQL Server 2008 Enterprise |
| SQL Server 2000 (32-Bit) Developer SP41,4 | No upgrade support |
| SQL Server 2000 (32-Bit) Enterprise SP41,4 | SQL Server 2008 Enterprise |
| SQL Server 2000 Enterprise Evaluation (32-bit, IA64, X64)4,5 | No upgrade support |
| SQL Server 2000 (64-Bit) IA64 Developer SP43,4,5 | SQL Server 2008 (64-Bit) IA64 Developer |
| SQL Server 2000 (64-Bit) IA64 Enterprise SP43,4,5 | SQL Server 2008 (64-Bit) IA64 Enterprise |
| SQL Server 2000 (32-Bit) Personal SP42 | No upgrade support |
| SQL Server 2005 (32-Bit) Express | SQL Server 2008 Express  SQL Server 2008 Express Advanced |
| SQL Server 2005 (32-Bit) Express Advanced1 | SQL Server 2008 Express Advanced |
| SQL Server 2005 (32-Bit) Workgroup1 | No upgrade support |
| SQL Server 2005 (32-Bit) Standard1 | SQL Server 2008 Standard  SQL Server 2008 Enterprise |
| SQL Server 2005 (32-Bit) Developer1 | No upgrade support |
| SQL Server 2005 (32-Bit) Enterprise1 | SQL Server 2008 Enterprise |
| SQL Server 2005 Enterprise Evaluation (32-bit, IA64, X64) | No upgrade support |
| SQL Server 2005 IA64 (64-bit) Developer | No upgrade support |
| SQL Server 2005 IA64 (64-bit) Standard | SQL Server 2008 IA64 (64-bit) Enterprise |
| SQL Server 2005 IA64 (64-bit) Enterprise | SQL Server 2008 IA64 (64-bit) Enterprise |
| SQL Server 2005 X64 (64-bit) Developer | SQL Server 2008 X64 (64-bit) Developer |
| SQL Server 2005 X64 (64-bit) Standard | SQL Server 2008 X64 (64-bit) Standard  SQL Server 2008 X64 (64-bit) Enterprise |
| SQL Server 2005 X64 (64-bit) Enterprise | SQL Server 2008 X64 (64-bit) Enterprise |
| SQL Server 2008 Express1 | SQL Server 2008 Express  SQL Server 2008 Express Advanced  SQL Server 2008 Standard  SQL Server 2008 Enterprise |
| SQL Server 2008 Express Advanced1 | SQL Server 2008 Express Advanced  SQL Server 2008 Standard  SQL Server 2008 Enterprise |
| SQL Server 2008 Express x64 (64-bit) | SQL Server 2008 Express x64 (64-bit)  SQL Server 2008 Express Advanced x64 (64-bit) |
| SQL Server 2008 Express Advanced x64 (64-bit) | SQL Server 2008 Express Advanced x64 (64-bit)  SQL Server 2008 Standard x64 (64-bit)  SQL Server 2008 Enterprise x64 (64-bit) |
| SQL Server 2008 Workgroup1 | SQL Server 2008 Workgroup  SQL Server 2008 Standard  SQL Server 2008 Enterprise |
| SQL Server 2008 Web1 | SQL Server 2008 Web |
| SQL Server 2008 Standard1,2 | SQL Server 2008 Standard  SQL Server 2008 Enterprise |
| SQL Server 2008 Developer1,2 | SQL Server 2008 Workgroup  SQL Server 2008 Standard  SQL Server 2008 Developer  SQL Server 2008 Enterprise |
| SQL Server 2008 Enterprise1,2 | SQL Server 2008 Enterprise |
| SQL Server 2008 Enterprise Evaluation2 | SQL Server 2008 Standard  SQL Server 2008 Enterprise  SQL Server 2008 Enterprise Evaluation |
| SQL Server 2008 IA64 (64-bit) Enterprise Evaluation2 | SQL Server 2008 IA64 (64-bit) Enterprise  SQL Server 2008 IA64 (64-bit) Enterprise Evaluation |
| SQL Server 2008 x64 (64-bit) Enterprise Evaluation2 | SQL Server 2008 x64 (64-bit) Standard  SQL Server 2008 x64 (64-bit) Enterprise  SQL Server 2008 x64 (64-bit) Enterprise Evaluation |
| SQL Server 2008 IA64 (64-bit) Developer2 | SQL Server 2008 IA64 (64-bit) Standard  SQL Server 2008 IA64 (64-bit) Developer  SQL Server 2008 IA64 (64-bit) Enterprise |
| SQL Server 2008 IA64 (64-bit) Standard2 | SQL Server 2008 IA64 (64-bit) Enterprise |
| SQL Server 2008 x64 (64-bit) Standard2 | SQL Server 2008 IA64 (64-bit) Standard  SQL Server 2008 IA64 (64-bit) Enterprise |
| SQL Server 2008 IA64 (64-bit) Enterprise2 | SQL Server 2008 IA64 (64-bit) Enterprise |
| SQL Server 2008 x64 (64-bit) Enterprise2 | SQL Server 2008 x64 (64-bit) Enterprise |

**Table Footnotes:**

1This SQL Server edition can be upgraded to SQL Server 2008 on the 32-bit subsystem (WOW64) of a 64-bit server.

2Changing the edition of a SQL Server 2008 failover cluster is limited. The following edition downgrade scenarios are not supported for SQL Server 2008 failover clusters:

* SQL Server 2008 Enterprise to SQL Server 2008 Developer, Standard, or Enterprise Evaluation
* SQL Server 2008 Developer to SQL Server 2008 Standard or Enterprise Evaluation
* SQL Server 2008 Standard to SQL Server 2008 Enterprise Evaluation
* SQL Server 2008 Enterprise Evaluation to SQL Server 2008 Standard

3Upgrade of SQL Server 2000 (64-Bit) IA64 failover clusters are not supported.

4Upgrade of SQL Server 2000 Analysis Services to SQL Server 2008 is not supported on failover clusters.

5Upgrade of SQL Server 2000 (64-bit) will not install SQL Server 2008 Management and Development Tools. To install the management and Development tools, you must rerun Setup after the upgrade is complete.

Appendix 2: Upgrade Planning Deployment and Tasks Checklist

| **Decision** | **Factors** | **Remarks** |
| --- | --- | --- |
| **Preparing to Upgrade** | | |
| Decide to upgrade to SQL Server 2008 | Identify the business reasons for upgrading to SQL Server 2008. |  |
| Choose SQL Server 2008 enhancements to implement | Select required and desired SQL Server 2008 features and enhancements for current and future development in the following categories:  Relational Database Features  Database Engine  High Availability  Security and Auditing.  Business Intelligence Features  Analysis Services  Data Mining  Integration Services  Reporting Services. |  |
| Determine instances of SQL Server to upgrade | Classify upgradable instances of SQL Server according to level of criticality:  High level: mission critical  Medium level  Low level.  Also classify according to whether instances are:  Default instance  Named instance  Virtual Machine. |  |
| Backward Compatibility and  Upgrade Tools | Gain familiarity with and select the appropriate upgrade tools for use in upgrade planning:  SQL Server 2008 Upgrade Advisor  SQL Server 2008 Upgrade Assistant  SQL Server 2000 and 2005 Best Practices Analyzer  SQL Server Profiler  Analysis Services Migration Wizard  DTS Package Wizard;  Run the SQL Server 2008 Upgrade Advisor to determine potential blocking issues :  Deprecated features  Discontinued features  Breaking changes  Behavior changes;  Run the SQL Server 2008 Upgrade Assistant to gather baseline performance data;  Run the SQL Server 2000 or 2005 Best Practices Analyzer to detect incorrect settings or other issues on legacy servers. |  |
| Ensure servers meet SQL Server 2008 requirements | Ensure that servers meet minimum requirements for SQL Server 2008, including:  Processor, memory, and disk space  Windows Server 2003 or higher  SQL Server 2000 SP4 (in-place upgrade)  SQL Server 2005 SP2 or later (on Windows Server 2008, in-place upgrade). |  |
| Decide on CPU platform: 64-bit or 32-bit | Determine what CPU platform to use for SQL Server servers, and ensure that database servers meet requirements. |  |
| Determine upgrade paths for editions | For each server, determine the proper SQL Server 2008 edition to upgrade to:  Enterprise  Standard  Workgroup  Developer  Web  SQL Server Express  Ensure that the legacy instances of SQL Server will have allowed upgrade paths for in-place upgrades. |  |
| Determine application connectivity requirements | Determine whether applications require any of the following:  Upgrade to support SQL Server 2008.  Change connectivity settings.  Change authentication mode.  Take measures to prevent SQL Injection. |  |
| Determine Windows upgrades | For each database server, determine whether it requires a Windows upgrade.  Note the restrictions on Windows Server versions for SQL Server:  SQL Server 2008 requires Windows Server 2003 or later  SQL Server 2000 is not supported on Windows Server 2008. |  |
| Select the appropriate upgrade strategy | For each server or database, choose the optimal upgrade strategy:  In-place upgrade  Side-by-side upgrade  Same physical server or VM  Separate physical server or VM. |  |
| **Developing an Upgrade Plan** | | |
| Upgrade as an IT Project | Follow standard IT project practices for developing the upgrade project. Identify:  Team members  Stakeholders  Application/business owners  Follow standard IT project procedures for the upgrade. |  |
| Update DBA skills to SQL Server 2008 | Ensure that DBA team members have SQL Server 2008 skills for implementing and potentially troubleshooting the upgrade. |  |
| Document the upgrade plan | Ensure that critical decisions and steps are documented and known to those involved in the upgrade. |  |
| Include other upgrade knowledge | Gather and consider lessons from past upgrades. |  |
| Minimize upgrade variables | Keep the project focused as much as possible on upgrading.  Avoid extending the project scope to application enhancements or fixes not related to upgrading. |  |
| Identify pre-upgrade tasks | Identify tasks that might be accomplished before the upgrade and without downtime.  For example, determine where it is possible to pre-install:  Microsoft Windows Installer (MSI) 4.5  .NET Framework 3.5 SP1  SQL Server Native Client 10.0. |  |
| Establish performance baselines | Use tools such as the SQL Server 2008 Upgrade Assistant and SQL Server Profiler to gather data indicating typical performance measurements.  Ensure that the broadest possible sets of commands are captured in traces. |  |
| Estimate required downtime | Allow sufficient downtime so that the upgrade process and testing can be completed successfully.  Allow time for rollback if unexpected issues arise. |  |
| Develop upgrade checklists | Detail the steps required for taking the systems offline for a period of time and bringing them back online.  Detail the steps to take during the upgrade processes. |  |
| Develop an upgrade test plan | Build a test environment.  Determine test procedures for each individual upgrade or upgrade type.  Test the upgrade checklists and procedures and revise as results indicate.  Be familiar with upgrade troubleshooting techniques. |  |
| Identify backup and restore operations | Plan for backing up the targeted legacy databases.  Verify the backups.  Plan for restoring the backup files if needed.  Test all backup procedures. |  |
| Determine acceptance and rollback steps | Identify how the organization will accept the upgrade, and how it will make the "go/no-go" decision:  Verify tests to ensure applications using the upgraded database servers will run as expected and required;  If available, enlist the support of the QA team to develop appropriate acceptance tests;  Determine exactly when and how a rollback to the legacy SQL Server might be required;  Test the rollback plan. |  |
| **Post-Upgrade Tasks** | | |
| Integrate the upgraded server | Remaining tasks might include the following:  Update statistics  Rebuild cubes  Reconfigure log shipping  Database mirroring  Test a failover cluster  Verify that SQL Server Agent jobs run correctly. |  |
| Decommission servers | After a suitable time period, after full acceptance of upgrades, decommission servers that are no longer needed for rollback or running in parallel. |  |
| Prepare for the next upgrade | Collect knowledge and experience from the upgrade project and store it so that lessons learned can be used by future upgrade projects. |  |